UNIVERSIDADE SAGRADO CORAÇÃO

GABRIEL SALLES ROUSSEAU GUEDES

CRIAÇÃO DE APLICATIVO MÓVEL PARA AUXILIAR NA CONTAGEM DE LEUCÓCITOS

BAURU
2016

GABRIEL SALLES ROUSSEAU GUEDES

CRIAÇÃO DE APLICATIVO MÓVEL PARA AUXILIAR NA CONTAGEM DE LEUCÓCITOS

Trabalho de conclusão de curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas da Universidade do Sagrado Coração como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Elvio Gilberto da Silva.

BAURU
2016

Guedes, Gabriel Salles Rousseau

G924c

Criação de Aplicativo Móvel para Auxiliar na Contagem de Leucócitos / Gabriel Salles Rousseau Guedes. -- 2016.

58f. : il.

Orientador: Prof. Dr. Elvio Gilberto da Silva.

Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) – Universidade do Sagrado Coração – Bauru – SP.

1. Aplicativo mobile. 2. Leucograma. 3. Android. I. Silva, Elvio Gilberto da. II. Título.

GABRIEL SALLES ROUSSEAU GUEDES

CRIAÇÃO DE APLICATIVO MÓVEL PARA AUXILIAR NA CONTAGEM DE LEUCÓCITOS

Trabalho de conclusão de curso apresentado ao Centro de Ciências Exatas e Sociais Aplicadas da Universidade do Sagrado Coração como parte dos requisitos para obtenção do título de bacharel em Ciência da Computação, sob orientação do Prof. Dr. Elvio Gilberto da Silva.

Banca examinadora:

_____
Prof. Dr. Elvio Gilberto da Silva
Universidade do Sagrado Coração

_____
Prof. M.e. Patrick Pedreira Silva
Universidade do Sagrado Coração

_____
Prof. M.e Daniela Barbosa Nicolielo
Universidade do Sagrado Coração

Bauru, 26 de novembro de 2016.

AGRADECIMENTOS

Agradeço ao meu pai Roger Guedes e minha mãe Graziella Salles Rousseau Guedes que me apoiaram em toda essa minha jornada acadêmica, a Danielle Líbano pela paciência nos dias dedicados ao TCC, ao meu orientador Prof. Dr. Elvio Gilberto da Silva que tanto me ajudou e incentivou neste trabalho.

RESUMO

O projeto consiste na criação de um aplicativo mobile Android em Java para auxiliar no ensino de contagem de leucócitos em ambientes acadêmicos e diminuir os custos com a aquisição de equipamentos, tanto nas universidades como em clínicas. O usuário da aplicação consegue cadastrar diversos pacientes, visualizar o resultado e exportar facilmente estes. É feito o uso do banco de dados SQLite para que nenhum dado seja perdido ao fechar a aplicação. Expandindo ainda mais as possibilidades de um equipamento comum, cada vez que o usuário seleciona uma célula, é aberta uma imagem representando a mesma, para que assim o estudante possa conferir se selecionou a célula correta.

Palavras-chave: Aplicativo mobile. Leucograma. Hemograma. Android

ABSTRACT

The project consists on the creation of a Java mobile Android application to aid in the teaching of leukocyte counting in academic environments and decrease the costs of acquiring equipment, both in universities and clinics. The application user can register several patients, view the result and easily export these. The SQLite database is used so that no data is lost when closing the application. By expanding even more the possibilities of a common equipment, each time the user selects a cell, an image representing the itself is opened, so that the student can check if he has selected the correct cell

Keywords: Mobile application. Leukogram. Blood count. Android

SUMÁRIO

1 INTRODUÇÃO

O sangue é um tecido fluido do meio interno que circula rapidamente dentro de um sistema fechado de vasos denominado sistema circulatório. É constituído por uma parte líquida no qual existem células em suspensão, moléculas e íons dissolvidos em água, apresentando propriedades das soluções coloidais. Ele é constituído de duas frações combinadas, sendo 55% para o plasma e 45% para as células. A porção acelular ou plasma é constituído de 91,5% de água que serve de solvente das substâncias orgânicas e minerais e ainda de veículo para as células, moléculas e íons. Os restantes 8% são formados por proteínas, sais e outros constituintes orgânicos em dissolução (CARVALHO, 2008).

A porção celular apresenta três tipos de células em suspensão no plasma:

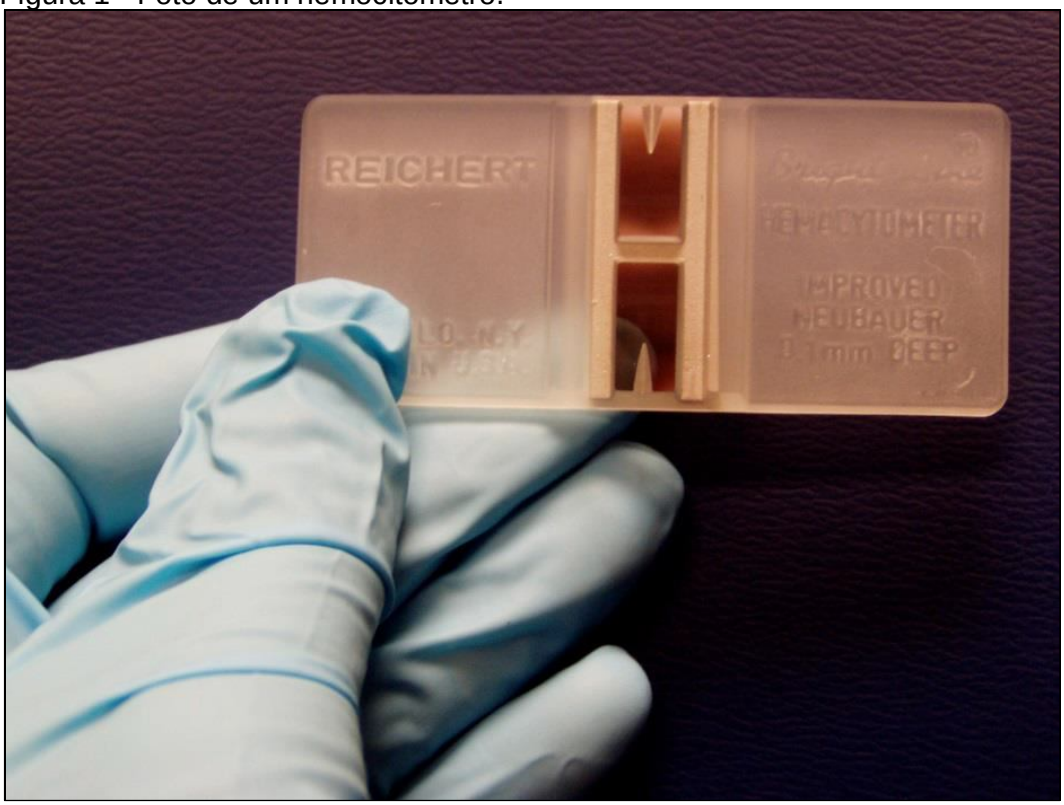
a) Glóbulos vermelhos, hemácias ou eritrócitos;
b) Glóbulos brancos ou leucócitos;
c) Plaquetas ou trombócitos.

Na clínica laboratorial, um dos processos fundamentais em vários exames é a contagem de células sanguíneas, tanto para o diagnóstico médico quanto para pesquisa biológica. O método de contagem de células sanguíneas pode ser manual, geralmente utilizando um hemocitômetro, como é mostrado na Figura 1, o que implica em um trabalho intensivo e demorado. Além da demora, o processo manual está sujeito a falhas, impactando nos resultados da contagem.

A contagem de células é expressa em concentrações — células por unidade de volume de sangue. A unidade de volume é expressa em milímetros cúbicos (mm^3) em decorrência das dimensões lineares da câmara do hemocitômetro (contador de células): 1 mm^3 = 1,00003 µL.

Qualquer procedimento de contagem celular deve passar por três etapas: diluição do sangue, coleta da suspensão diluída em um volume específico, e por fim, contagem das células contidas neste volume (CARVALHO, 2008).

Figura 1 - Foto de um hemocitômetro.



Fonte: HEMOCYTOMETER, 2014.

O hemocitômetro é um dispositivo de cristal espesso com o tamanho de uma lâmina de vidro (30 mm x 70 mm x 4 mm), dividido em 3 partes. (BASTIDAS, 2013).

Na parte central da lâmina, existem várias linhas perpendiculares com marcações em quadrantes. Ao analisar em microscópico, com a objetiva de imersão, pode-se perceber que existem três tipos de quadrantes, que juntos formam um quadrado maior (CARVALHO, 2008). Esses quadrantes são utilizados para fazer as contagens da concentração de células em um determinado volume de fluido (CARVALHO, 2008).

Existem equipamentos à venda no mercado nacional que automatizam a contagem, mas como são importados, têm alto custo, além de não serem tão precisos.

Tendo em vista essa problemática, este trabalho propõe desenvolver um sistema para auxiliar na contagem de células sanguíneas a partir do

desenvolvimento de um aplicativo móvel de forma que reduza o custo na obtenção e manutenção de aparelhos.

O hemograma é o nome dado ao conjunto de avaliações das células do sangue que, reunido aos dados clínicos, permite conclusões diagnósticas e prognósticas de grande número de patologias. A introdução do hemograma na prática médica ocorreu em 1925 por meio de critérios estabelecidos pelo médico e farmacêutico alemão V. Schilling. (JUNQUEIRA, 2008)

Entre todos os exames laboratoriais atualmente solicitados por médicos de todas as especialidades, o hemograma é o mais requerido. Por essa razão reveste-se de grande importância no conjunto de dados que devem ser considerados para o diagnóstico médico, não se admitindo erros ou conclusões duvidosas. (JUNQUEIRA, 2008)

O hemograma é composto por três determinações básicas que incluem as avaliações dos eritrócitos (ou série vermelha), dos leucócitos (ou série branca) e das plaquetas (ou série plaquetária). (BAIN, 2007)

Normalmente o processo de contagem de células sanguíneas é feito de forma manual ou utilizando equipamentos de alto custo. Além disso, algumas vezes os exames são realizados em lugares remotos, então uma solução portátil torna-se interessante.

No curso de Biomedicina da Universidade do Sagrado Coração – USC, nas disciplinas práticas que envolvem a contagem de células, este procedimento é feito em um equipamento simples, porém de custo relativamente alto, e suas peças muitas vezes acabam se danificando, além de que, algumas vezes não existe o número de equipamentos necessários para atender a todos os alunos da turma durante as aulas práticas.

O desenvolvimento de uma aplicativo móvel para essa finalidade implica em uma redução de custos para instituições de ensino superior, e ainda proporcionaria a possibilidade de oferecer mais recursos, tais como, a visualização de imagens para que os alunos fizessem a conferência da célula avaliada, além de, sugestões acerca de um possível diagnóstico do paciente baseado na contagem das células. O protótipo aqui proposto tem como objetivo auxiliar o usuário na contagem de células sanguíneas, diminuindo o tempo e o custo das análises bioquímicas.

2 OBJETIVOS

Apresenta-se a seguir o objetivo geral e específicos desta pesquisa.

2.1 OBJETIVO GERAL

Desenvolver um aplicativo móvel para fins acadêmicos que possibilite a contagem de leucócitos.

2.2 OBJETIVOS ESPECÍFICOS

a) Realizar um levantamento bibliográfico dos métodos a serem utilizados.

b) Modelar e implementar um protótipo.

3 COMPOSIÇÃO DO SANGUE

O sangue é um tecido conjuntivo líquido formado por células e plasma, parte líquida. As células são divididas em três grupos: hemácias, leucócitos e plaquetas. (JUNQUEIRA, 2008)

3.1 HEMÁCIAS

Hemácias, também conhecidas como eritrócitos ou glóbulos vermelhos, são células anucleadas responsáveis pelo transporte de oxigênio para as células e gás carbônico para os pulmões. (JUNQUEIRA, 2008)

3.2 LEUCÓCITOS

Leucócitos, ou glóbulos brancos, são as células responsáveis pela defesa do organismo e são uma das primeiras barreiras contra infecções (JUNQUEIRA, 2008)

Segundo o autor supracitado, há diferentes leucócitos com diferentes funções:

a) Neutrófilo: eficiente no combate a bactérias e fungos.
b) Eosinófilo: eficiente no combate a parasitas, modulação do processo inflamatório, participação em reações alérgicas, ação antiviral.
c) Basófilo: Liberação de histamina e outros mediadores da inflamação, participação em reações alérgicas.
d) Monócito: combate a protozoários, certas bactérias, vírus, células senescentes e apresentação de antígenos para linfócitos.
e) Linfócito: destruição de células tumorais e de células infectadas por vírus.

3.3 PLAQUETAS

Plaquetas são células anucleadas cuja função é participar da coagulação do sangue e o reparo das paredes dos vasos sanguíneos. (FAILACE, 2009)

4 HEMOGRAMA

“O hemograma é o exame que avalia quantitativa e qualitativamente os elementos celulares do sangue. É o exame complementar mais requerido nas consultas, fazendo parte de todas as revisões de saúde.” (FAILACE, 2009, p. 21).

Dentre os exames que podem ser realizados com sangue coletado, há uma preferência em realizar o hemograma (48% das vezes), pois permite identificar e acompanhar diversos problemas de saúde, como doenças infecciosas, doenças crônicas, emergências médicas, cirúrgicas e traumatológicas, também sendo útil no acompanhamento de quimioterapia e a radioterapia (FAILACE, 2009).

O hemograma pode ser dividido em três partes: eritrograma, leucograma e plaquetograma.

4.1 ERITOGRAMA

Eritrograma inclui a contagem das hemácias (também chamadas de eritrócitos ou glóbulos vermelhos), a dosagem de hemoglobina, contagem de reticulócitos, entre outros. (BAIN, 2007)

4.2 LEUCOGRAMA

Leucograma é a porção do hemograma que se limita a contagem de leucócitos (glóbulos brancos), a morfologia destes e a fórmula leucocitária. Diferentes grupos possuem contagens diferentes, por exemplo, mulheres possuem 3-4% a mais leucócitos do que os homens. A contagem média é de 6.540/μL em homens brancos e de 6.760/μL em mulheres brancas. Na população negra é de 5.740/μL em homens e de 6.140/μL em mulheres. Tanto a obesidade quanto o fumo aumentam, aproximadamente, em 1.000 leucócitos/μL. Outro causa que resulta na variação da contagem é o horário da coleta: se matinal, em jejum, será influenciado pela quantidade de horas de sono e as atividades na noite anterior, resultando numa quantidade de leucócitos aproximadamente 5% menor do que numa coleta vespertina (FAILACE, 2009).

Apesar de haver estas variações nas contagens de populações, em indivíduos, analisados isoladamente, não é esta a realidade: se a coleta for realizada sempre nas mesmas condições e horário, será perceptível um padrão pessoal. Por este motivo é importante sempre armazenar os leucrogramas, possibilitando, num exame novo, comparar com os resultados de exames anteriores. Se houver uma variação significativa na contagem do paciente, com o aumento destas células são denominadas de leucocitose (quando estão a cima do valor de referência) e leucocitopenia/leucopenia (quando estão abaixo do valor de referência) (FAILACE, 2009);

4.2.1 Fórmula leucocitária

A Fórmula Leucocitária foi criada por Paul Ehrlich, em 1879 (FAILACE, 2009, P. 242) e constitui em contar, de uma amostra, no mínimo 100 leucócitos e então representar seus diferentes tipos em porcentagem (BAIN, 2007).

Como o os diferentes tipos de leucócitos não se distribuem uniformemente sobre a lâmina, um resultado obtido através da contagem, utilizando um microscópio, é pouco reprodutível, porém é suficientemente exato (BAIN, 2007).

4.2.2 Valores de referência

Para Angulo, os valores de referência para o leucograma de um adulto caucasiano podem ser visualizadas na Tabela 1.

Tabela 1 – Tabela de referência.

	%	**/µL**
Neutrófilos	40 – 70	1600 – 7000
Bastonetes (neutrófilos jovens)	1 – 4	40 – 400
Linfócitos	18 – 48	1000 – 4500
Monócitos	3 – 10	200 – 1000
Eosinófilos	1 – 6	100 – 600
Basófilos	0 – 3	0 – 200

Fonte: ANGULO, 2016

Para Failace (2009), a tabela para o mesmo grupo é a apresentada na Tabela 2.

Tabela 2 – Tabela de referência.

	%	/µL
Leucócitos	–	3600 – 11000
Neutrófilos*	40 – 70	1500 – 6800
Linfócitos	20 – 50	1000 – 3800
Monócitos	2 – 10	100 – 800
Eosinófilos	1 – 7	50 – 400
Basófilos	0 – 3	0 – 200
*Neutrófilos totais. Bastonetes = 0 – 6 %		

Fonte: FAILACE, 2009

4.2.3 Alterações na contagem de leucócitos

Segundo Bain (1998), alterações na contagem de leucócitos recebem as seguintes as denominações conforme apresentadas no Tabela 3.

Tabela 3 – Alterações na contagem.

Leucocitose	Aumento da contagem de Leucócitos
Neutrofilia ou neutrocitose	Aumento da contagem de Neutrófilo
Linfocitose	Aumento da contagem de Linfócito
Monocitose	Aumento da contagem de Monócito
Eosinofilia ou eosinocitose	Aumento da contagem de Eosinófilo
Basofilia ou basocitose	Aumento da contagem de Basófilo
Leucopenia ou leucocitopenia	Diminuição da contagem de Leucócitos
Neutropenia ou neutrocitopenia	Diminuição da contagem de Neutrófilo
Linfopenia ou linfocitopenia	Diminuição da contagem de Linfócito
Monocitopenia	Diminuição da contagem de Monócito
Eosinopenia ou eosinocitopenia	Diminuição da contagem de Eosinófilo
Basopenia ou basocitopenia	Diminuição da contagem de Basófilo

Fonte: BAIN, 1998

O Tabela 4 ilustra as possíveis alterações na contagem e suas respectivas possíveis causas segundo Bain (1998; 2007).

Tabela 4 – Possíveis causas de alterações.

Alteração	Possíveis causas
Neutrofilia	• Tabagismo • Exercício físico, convulsões, injeção de adrenalina • Período neo-natal, gravidez, pós-parto • Alta dose de corticóide ou de lítio • Administração de citoquinas • Muitas infecções bacterianas • Algumas infecções virais • Algumas infecções fúngicas • Algumas parasitoses • Infarto tecidual (infarto o miocárdio) • Dano tecidual (queimadura, cirurgia, trauma) • Inflamação aguda • Hemorragia ou hipóxia agudas • Leucemia • Doenças mieloproliferativas crônicas • Envenenamento, como picada de cobra, abelha e ofídio
Linfocitose	• Exercício físico • Tabagismo • Administração de adrenalina • Reação ao estresse físico (após trauma, crise drepanocítica, infarto do miocárdio, parada cardíaca) • Infecções virais • Malária • Certas infecções bacterianas como rickettsias, coqueluche, sífilis • Infecções bacterianas em lactantes e crianças pequenas
Monocitose	• Infecção crônica • Condições inflamatórias crônicas • Malária • Hemodiálise a longo prazo
Eosinofilia	• Doenças alérgicas • Asma • Hipersensibilidade medicamentosa

	• Infecção parasitária • Doenças cutâneas como pênfigo, penfigóide bolhoso, herpes gestacional, foliculite pustular eosinofillica
Basofilia	• Mixedema (hipertireoidismo) • Colite ulcerativa • Estados de hipersensibilidade • Administração de estrógeno • Hiperlipidemia • Síndrome hipereosinofillica idiopática • Administração de IL-3 • Desordens leucêmicas e mieloproliferativas como leucemia granulocítica crônicas, outras leucemias mielóides crônicas, policitemia rubra vera, trombocitemia essencial, mielofibrose idiopática, mastocitose sistêmica, leucemia basofilica
Neutropenia	• Infecções virais inclusive pelo HIV em estágio avançado • Infecções bacterianas • Infecções por protozoários • Infecções fúngicas • Alcoolismo • Deficiência de cobre • Hipercarotenemia • Drogas, como os agentes alquilantes e outros antineoplásicos e drogas correlatas, reações idiossincrásica a drogas • Irradiação • Substituição da medula óssea • Hemodiálise e leucoferese de filtração (no início dos procedimentos) • Doença Rh do recém-nascido • Bebês com mães hipertensas • Asfixia do recém-nascido • Oxigenação por membrana extracorpórea em neonatos • Diversas desordens hereditárias
Linfopenia	• Algumas raras síndromes de imunodeficiência congênita • Estresse agudo, incluindo trauma, cirurgia, queimaduras, infecção aguda, insuficiência hepática fulminante • Insuficiência renal aguda e crônicas • Síndrome de Cushing e administração de corticoides ou ACTH

	• Alguns linfomas • AIDS • Tratamento com clozapina • Vacinação contra *influenza* • Irradiação • Alcoolismo • Anorexia nervosa • Anemia ferropênica • Doença do enxerto *versus* hospedeiro • Administração de “óleo de Lorenzo”
Monocitopenia	• Administração de corticosteroides • Infecções agudas associadas a endotoxemia • Leucemia de células cabeludas
Eosinopenia	• Estresse agudo, incluindo trauma, cirurgia, queimaduras, convulsões epileptiformes, infecções agudas, inflamação aguda, infarto do miocárdio, anóxia e exposição ao frio • Síndrome de Cushing • Drogas, incluindo os corticosteroides, o ACTH, a adrenalina e outros β-agonistas, a histamina e a aminofilina • Hemodiálise (durante o procedimento)
Basopenia	• Estresse agudo, incluindo infecção e hemorragia • Síndrome de Cushing e administração de ACTH • Anafilaxia, urticária aguda e outras reações alérgicas agudas • Hipertireoidismo • Administração de progesterona

Fonte: BAIN, 1998 e 2007

4.3 PLAQUETOGRAMA

Plaquetograma é a contagem e medição das plaquetas dentro do hemograma. Esta contagem é necessária que um técnico apresente vasta experiência e um microscópio de alta qualidade, além de demandar muito tempo. Por este motivo, exceto em casos excepcionais, esta contagem só é realizada com contadores eletrônicos (FAILACE, 2009).

5 SOFTWARE

Segundo Sommerville (2010), software não é somente um programa (instruções) de computador, mas também toda a documentação associada a ele e as configurações necessárias para o seu funcionamento.

> Um sistema de software consiste, geralmente, de um conjunto de programas separados; arquivos de configuração, que são utilizados para configurar esses programas; documentação do sistema, que descreve a estrutura do sistema; a documentação do usuário, que explica como usar o sistema; e sites Web por meio dos quais os usuários obtêm informações recentes sobre o produto.” (SOMMERVILLE, 2010, p. 103).

5.1 TIPOS DE SOFTWARE

Para PRESSMAN (2011), existem 7 tipos de software e estes estão apresentados a seguir:

a) **Software de sistema**: são os programas que proporcionam o funcionamento básico do computador e auxiliam outros programas. Softwares de sistema são caracterizados por grande interação com o hardware do computador, uso intenso por múltiplos usuários, compartilhamento de recursos, gestão de processos, estrutura de dados complexas. Alguns exemplos deste tipo de software seriam compiladores, utilitários de gerenciamento de arquivos, drivers, software de rede, componentes do sistema operacional, processadores de telecomunicação.
b) **Software de aplicação**: programas feito sob medida para atender necessidades específicas. Estes são responsáveis por processar dados comerciais que auxiliam nas tomadas de decisões. O software de aplicação também é responsável por controlar funções de negócio em tempo real, como transações comerciais e processos de fabricação em tempo real.
c) **Software científico/de engenharia**: caracterizado por utilizar um pesado processamento numérico. É utilizado em diversas áreas de

pesquisa, como astronomia, vulcanologia, análise de tensões em indústrias, dinâmica orbital de ônibus espacial, biologia molecular.

d) **Software embutido**: é um software criado para um hardware específico, não tenso utilidade fora dele. Alguns exemplos de software embutido: controle de painel de micro-ondas, sistema de freios, controle de nível de combustível de um veículo.

e) **Software para linha de produtos**: softwares projetados para atender necessidades de um grande número de clientes diferentes. Podem atender nichos (software para controle de estoque, editores profissionais de vídeo) ou mercados de consumo de massa (processamento de texto, planilhas eletrônicas, computação gráfica, multimídia, gerenciamento de banco de dados, aplicações financeiras pessoais e comerciais).

f) **Aplicações Web**: também chamadas de “WebApps”, esta categoria de softwares se faz necessário o uso de uma rede (internet, por exemplo). Geralmente acessadas por navegadores, atualmente elas não apenas entregam ao usuário apenas hipertexto, mas também estão integradas a banco de dados corporativos.

g) **Software de inteligência artificial**: utiliza algoritmos não numéricos para atender questões complexas que não são resolvíveis por análise direta. Este tipo de software atende diversas áreas, como a robótica, sistemas especialistas, redes neurais artificiais, reconhecimento de padrões (de imagem e som), prova de teoremas e jogos.

Embora Wazlawick (2013) faça uma separação de tipos de softwares bem semelhante ao de Pressman (2011), para ele há mais um tipo de software: Jogos. Neste incluem diferentes tipos de jogos, como os que a quantidade de processamento não é uma grande preocupação (Campo Minado, Paciência), como também os que exigem muito dos recursos do computador, incluindo processador, placa gráfica e memória RAM.

6 ENGENHARIA DE SOFTWARE

De acordo com o IEEE, a engenharia de software é “a aplicação de uma abordagem sistemática, disciplinada e quantificável no desenvolvimento, operação e manutenção de software; ou seja, a aplicação de engenharia ao software.” (ISO/IEC/IEEE 24765:2010).

A função do engenheiro de software inclui estudar, criar e otimizar os processos de trabalho para o desenvolvimento de software (WAZLAWICK, 2013), isto engloba todos os aspectos da produção, desde os estágios iniciais do sistema até a manutenção (SOMMERVILLE, 2010).

6.1 UML

A UML (Unified Modeling Language, linguagem de modelagem unificada) é uma linguagem gráfica que possui a finalidade de organizar e representar a estrutura de um software. BOOCH, um dos seus criadores, define a UML como *“(…) uma linguagem visual de modelagem que é usada para especificar, visualizar, construir e documentar os artefatos de um sistema de software. Ela é usada para compreender, desenhar, buscar, configurar, manter e controlar a informação sobre o sistema. Ela é destinada para o uso em todos os métodos de desenvolvimento, fases do ciclo de vida* (do software) *(…)”* (THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL, 2010).

Ela foi apresentada em 1997 e sua criação utilizou muito das informações fornecidas pela comunidade de desenvolvimento e das notações comumente utilizadas pela indústria de software (PRESSMAN, 2011).

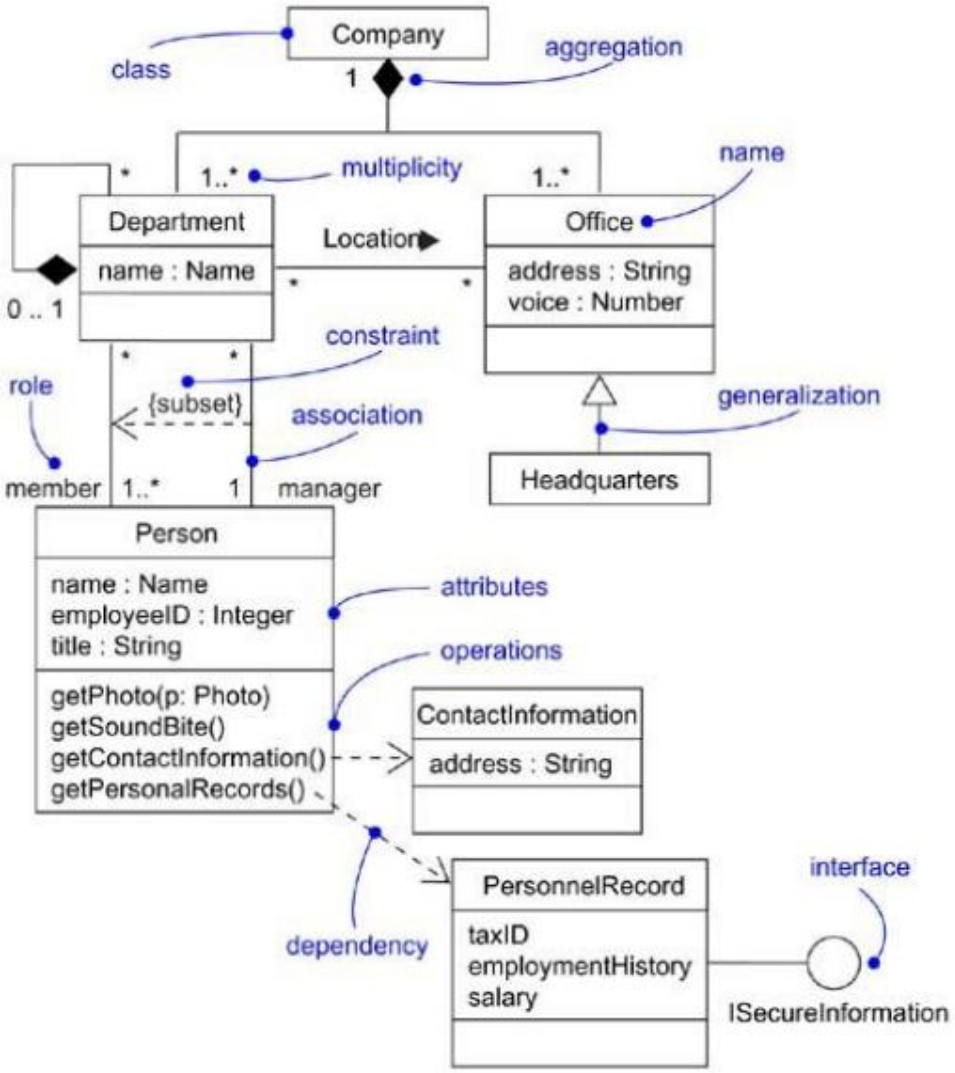
Para a modelagem, a UML 2 fornece 13 diagramas distintos, incluindo os seguintes: *diagrama de classe, diagrama de atividade, diagrama de caso de uso, diagrama de estado, diagrama de comunicação, diagrama de sequencia e diagrama de distribuição* (PRESSMAN, 2011).

6.1.1. Diagrama de classe

O Diagrama de Classe (representado na figura 2) tem como objetivo representar as classes de um software, seus métodos, variáveis e interações com outras classes. Este modelo fornece uma visão estrutural do sistema, não demonstrando a dinamicidade da comunicação entre os objetos das classes (PRESSMAN, 2011).

As classes são representadas por caixas. Estas caixas são divididas horizontalmente, aonde a seção superior contém o nome da classe. A seção intermediaria lista os atributos da classe, ou seja, dados que esta classe armazena (endereço) ou calcula (horário atual). A ultima seção do diagrama contém as operações ou comportamentos da classe, equivalente aos métodos de uma classe na programação orientada a objeto (PRESSMAN, 2011).

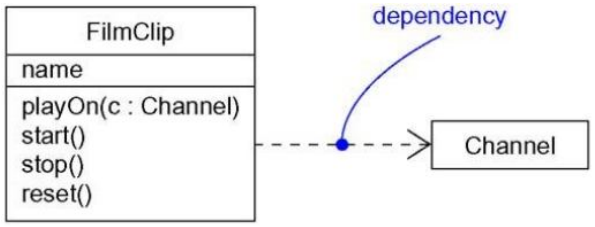
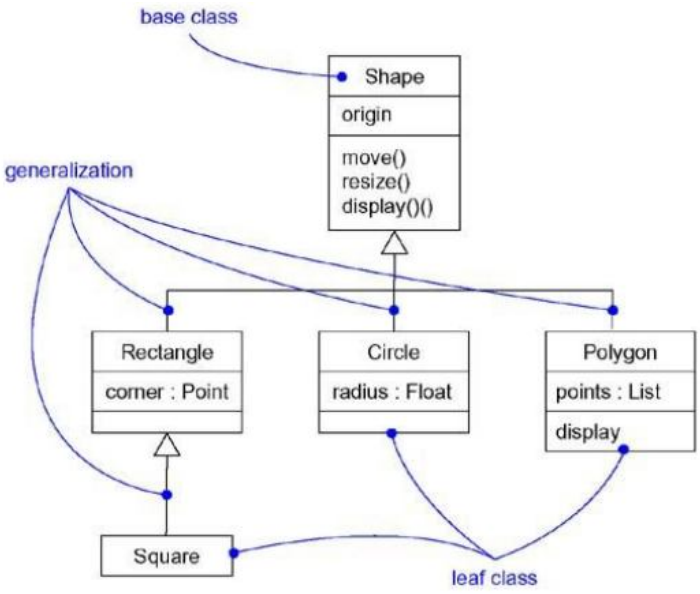
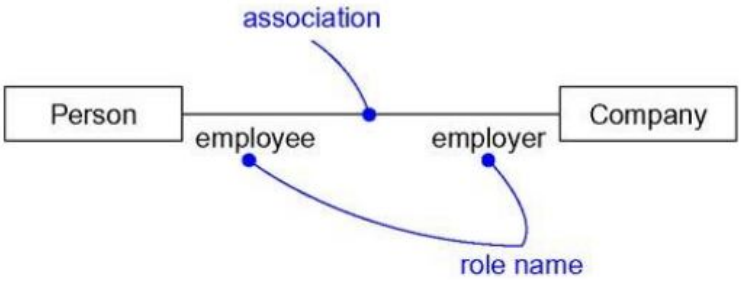
Figura 2 – Diagrama de classe



Fonte: THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL, 2005

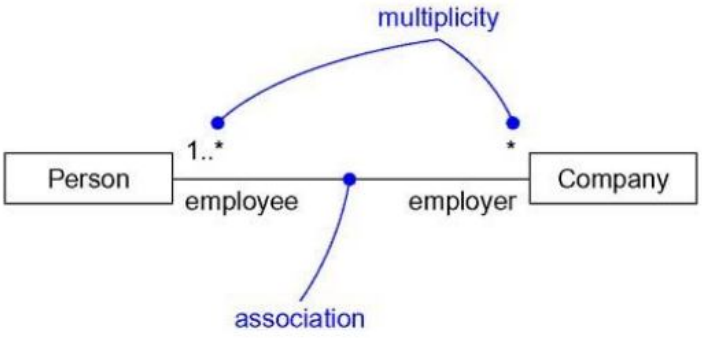
Quando há relação entre classes, esta é representada por um linha. Diferentes tipos de linha representam diferentes tipos de relação. As relações listadas por Booch (2005) são apresentadas na tabela a seguir:

Tabela 5: Representações do diagrama de classes

NOME / SIGNIFICADO	REPRESENTAÇÃO
Dependência: Quando há mudança na especificação de uma classe A, uma classe B é afetada, mas não necessariamente o contrário é verdadeiro.	 Font e: THE UNIF IED MODELING LANGUAGE REFERENCE MANUAL, 2005
Generalização: Generalização é a relação entre uma classe mais geral (superclasse) e uma mais específica (subclasse). Uma subclasse herda todos os atributos e comportamentos da superclasse, podendo ainda sobrescrever estes e adicionar novos.	 Fonte: THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL, 2005
Associação: É uma relação que especifica que um objeto de uma classe está conectado com um objeto de outra classe. A partir de um objeto da associação, é possível navegar até o outro objeto e vice-versa. É possível que uma classe	 Fonte: THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL, 2005

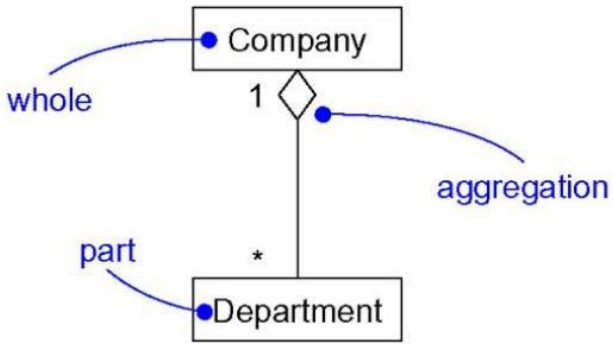
possua associação com ela mesma, o que representa que um objeto desta classe está associado com outro objeto desta mesma.

Multiplicidade: Se for importante representar quantos de um objeto estão associados com outro objeto, é possível utilizar a multiplicidade. Os valores mais comuns são um (1), zero ou um (0..1), muitos (*) e um ou muitos (1..*). É possível também definir valores exatos (por exemplo, 3) ou intervalos mais complexos, como 0..1, 3..4, 6..*, que representa “qualquer valor que não seja 2 ou 5”.



Fonte: THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL, 2005

Agregação: Numa associação comum, os dois ou mais objetos da relação estão no mesmo nível. Caso deseje demostrar que um objeto é uma parte de outro, utiliza-se a agregação. Um exemplo é a relação de uma empresa com seus departamentos: um departamento é uma parte da empresa.



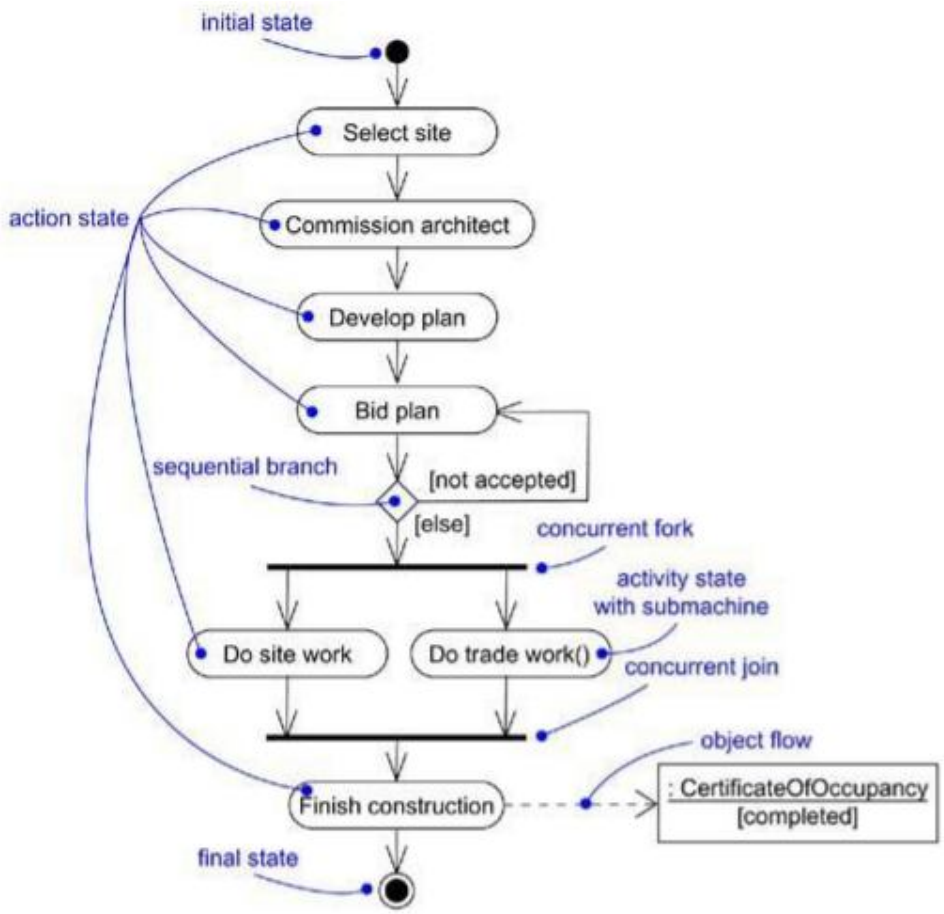
Fonte: THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL, 2005

Fonte: Elaborada pelo autor

6.1.2 Diagrama de Atividade

O Diagrama de Atividade (exemplificado na figura 3) representa a dinamicidade do sistema, mostrando a sequencia de passos (inclusive passos concorrentes) do processo (BOOCH, 2005).

Figura 3 – Diagrama de atividade



Fonte: THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL, 2005

Para Pressman (2011) os componentes deste diagrama são:

a) **Estado inicial**: Representado por um ponto preto sólido, é o ponto inicial da atividade.
b) **Nó ação**: Representado por um retângulo de bordas arredondas, corresponde uma tarefa executada pelo sistema.

c) **Nó de decisão**: É uma ramificação do fluxo baseado em uma condição. Se a condição for verdadeira, segue um fluxo; se for falsa, segue outro. É representado por um losango.
d) **Fork**: Representada por uma barra preta horizontal, é quando ocorre a divisão de uma tarefa em 2 ou mais tarefas concorrentes.
e) **Junção**: Assim como o fork, é representada por uma barra preta horizontal, porém possui finalidade oposta ao do fork: sincronizar dois ou mais fluxos concorrentes.
f) **Estado Final**: Representado por um ponto preto envolvido por um círculo preto, é o fim da atividade.

6.1.3. DIAGRAMA DE CASO DE USO

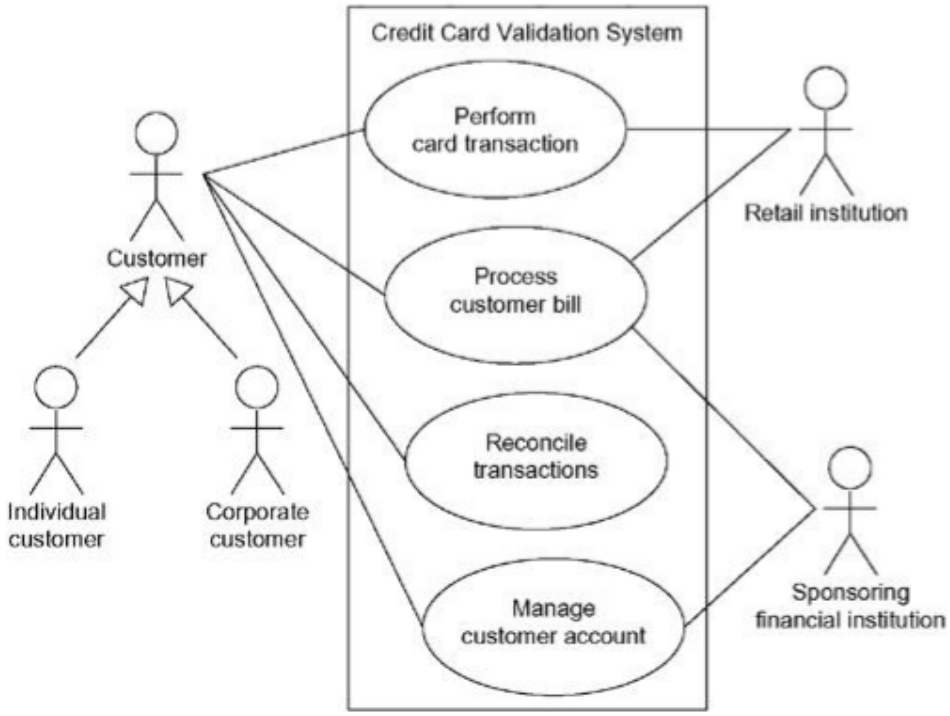


Figura 4 – Diagrama de caso de uso

Fonte: THE UNIFIED MODELING LANGUAGE REFERENCE MANUAL, 2005

Diagrama de caso de uso (exemplificado na figura 4) possui a finalidade de representar o sistema de software a partir do ponto de vista do usuário. (PRESSMAN, 2011) É um diagrama importante para visualizar, especificar e documentar o comportamento de um elemento. (BOOCH, 2005)

Os casos de uso são representados como ovais e os usuários representados pela figura *ator*. Um sistema complexo pode ter mais de um ator, por exemplo o de uma “vending machine”. Nela poderá haver 3 atores: cliente, pessoal da manutenção e fornecedores. Os atores são conectados aos casos de uso por linhas. (PRESSMAN, 2011)

Para demonstrar a fronteira do sistema e lembrar que os atores não estão dentro dele, é colocado um retângulo em volta dos casos de uso. (PRESSMAN, 2011)

6.4 ANÁLISE DE REQUISITOS

Análise de requisitos é a etapa do desenvolvimento de software que tem como finalidade compreender as necessidades do sistema. Para isto, o engenheiro de software conversa diretamente com o usuário para listar corretamente os requisitos.

Para Maciaszek (2001), requisito é *“(...) uma condição ou capacidade que deve ser alcançada ou possuída por um sistema ou componente deste para satisfazer um contrato, padrão, especificação ou outros documentos formalmente impostos.”*

O resultado final da análise de requisitos é o **documento de requisitos**, que possui como principais seções, segundo Bezerra (2015) os Requisitos Funcionais (as funcionalidades do sistema), os Requisitos Não-Funcionais (as características de qualidade que o sistema deve ter) e as Restrições (restrições impostas no desenvolvimento o sistema).

Requisitos funcionais tem como exemplo “O sistema deve permitir que cada professor realize o lançamento de notas das turmas nas quais lecionou”.

De acordo com Bezerra (2015) há 5 requisitos não-funcionais: confiabilidade (o quanto o sistema não irá falhar, tempo médio entre falhas, recuperação de falhas,

erros em linhas de código-fonte), desempenho (o tempo de resposta esperado das funcionalidades), portabilidade (quais plataformas de software e hardware que o sistema deverá funcionar e qual o grau de dificuldade em realizar a portabilidade para outras plataformas), segurança (regras de segurança sobre acessos) e usabilidade (o quão é fácil para o usuário utilizar e se é ou não necessário o treinamento dos usuários).

As restrições impostas ao desenvolvimento do sistema são definidas a partir dos custos, prazos, a plataforma que será utilizada, aspectos legais, limitações do hardware e quais hardwares e softwares devem ser adquiridos.

6.5 INTERAÇÃO HUMANO-COMPUTADOR

IHC (interação humano-computador) é a área de estudo que busca entender como as pessoas interagem com a tecnologia da informação. O principal objetivo desta área, segundo Rosa (2012), é projetar e desenvolver sistemas com a finalidade de melhorar a eficacia e proporcionar uma maior satisfação ao usuário.

IHC não se limita apenas aos conhecimentos de computação. De acordo com Barbosa (2010), ela também se apoia em outras áreas, como a psicologia, sociologia, antropologia, que contribuem para obter informações sobre a cultura do usuário e a forma como ele se comporta em ambientes. Outras áreas que auxiliam são as de design, ergonomia, linguística, semiótica, que serão utilizadas na criação da interface.

6.5.1 INTERFACE

A única forma do usuário interagir com um sistema é através de sua interface. Esta inclui, segundo Barbosa (2010), toda a porção do sistema que o usuário mantém contato físico ou conceitual. O contato físico ocorre através de hardwares e softwares que são utilizados durante a interação, como teclado, mouse, microfone, caixas de som, webcam, monitor, impressora.

O contato conceitual envolve a interpretação do usuário em relação as respostas do sistema. Esta interpretação permite que ele não apenas entenda o que está sendo apresentado pela interface, mas também planejar os próximos passos da interação.

Barbosa (2010) nos explica que a interface delimita para o usuário como que ele pode interagir com o sistema, o que é possível realizar, o que não é possível e em qual ordem. Por exemplo, de nada adianta o usuário tentar dar comandos de voz para um sistema se a interface não aceita este tipo de interação.

6.6 TESTE DE SOFTWARE

Antes de iniciarmos uma discussão sobre teste de software precisamos esclarecer alguns conceitos relacionados a essa atividade. Inicialmente, precisamos conhecer a diferença entre **Defeitos**, **Erros** e Antes de iniciarmos uma discussão sobre teste de software precisamos esclarecer alguns conceitos relacionados a essa atividade. Inicialmente, precisamos conhecer a diferença entre Defeitos, Erros e Falhas. As definições que iremos usar aqui seguem a terminologia padrão para Engenharia de Software do IEEE – Institute of Electrical and Electronics Engineers – (IEEE 610, 1990 citado por Claudio ([201-]).

a) **Defeito** é um ato inconsistente cometido por um indivíduo ao tentar entender uma determinada informação, resolver um problema ou utilizar um método ou uma ferramenta. Por exemplo, uma instrução ou comando incorreto.
b) **Erro** é uma manifestação concreta de um defeito num artefato de software. Diferença entre o valor obtido e o valor esperado, ou seja, qualquer estado intermediário incorreto ou resultado inesperado na execução de um programa constitui um erro.
c) **Falha** é o comportamento operacional do software diferente do esperado pelo usuário. Uma falha pode ter sido causada por diversos erros e alguns erros podem nunca causar uma falha.

A Figura 5 expressa a diferença entre esses conceitos.

Figura 5

Fonte: CLAUDIO ([201-]).

Como pode ser observado na Figura 5, defeitos fazem parte do universo físico (a aplicação propriamente dita) e são causados por pessoas, por exemplo, através do mal uso de uma tecnologia, podendo ocasionar a manifestação de erros em um produto, ou seja, a construção de um software de forma diferente ao que foi especificado (universo de informação). Por fim, os erros geram falhas, que são comportamentos inesperados em um software, que afetam diretamente o usuário final da aplicação (universo do usuário), e podem inviabilizar a utilização de um software.

Segundo Claudio ([201-]), teste de software é o processo de execução de um produto para determinar se ele atingiu suas especificações e funcionou corretamente no ambiente para o qual foi projetado, ao mesmo tempo em que faz parte de todo o processo de engenharia de software. O seu objetivo é revelar falhas em um produto, para que as causas dessas falhas sejam identificadas e possam ser corrigidas pela equipe de desenvolvimento antes da entrega final. Por conta dessa característica das atividades de teste, dizemos que sua natureza é “destrutiva”, e não “construtiva”, pois visa ao aumento da confiança de um produto através da exposição de seus problemas, porém antes de sua entrega ao usuário final.

Ainda segundo o autor, o conceito de teste de software pode ser compreendido através de uma visão intuitiva ou mesmo de uma maneira formal. Existem atualmente várias definições para esse conceito. De uma forma simples, testar um software significa verificar através de uma execução controlada se o seu

comportamento ocorre de acordo com o especificado. O objetivo principal desta tarefa é revelar o número máximo de falhas dispondo do mínimo de esforço, ou seja, mostrar aos que desenvolvem se os resultados estão ou não de acordo com os padrões estabelecidos.

Para Pressman et al. (2011), o processo de testes visa encontrar falhas no sistema para corrigi-las antes de distribuir o sistema, ou de atualizar o software com novos recursos, portanto o sistema deve ser projetado e implementado pensando na facilidade da realização dos testes. Por sua vez os testes devem ser feitos levando em consideração certas características para que nada seja deixado para trás.

Em conformidade com o autor, o conceito de testabilidade consiste em se medir o quão simples é o ato de testar um software. As características apresentadas seguir caracterizam um software a ser testável:

a) Operabilidade: um sistema projetado e implementado tendo em mente a qualidade, terá poucas falhas quando os testes forem realizados.
b) Observabilidade: quando é possível ver com clareza as entradas, saídas e variáveis do sistema fica mais fácil de detectar possíveis falhas.
c) Controlabilidade: entradas geram saídas especificas, e para cada tipo de saída existirá um tipo de entrada especifica. Se o engenheiro puder controlar essas entradas ficará mais fácil realizar os testes.
d) Decomponibilidade: o sistema é construído a partir de módulos e pode ser testado em partes.
e) Simplicidade: Quanto mais simples um sistema for, atingindo o objetivo, mais simples serão os testes.
f) Estabilidade: Quanto menos alterações o software tiver, menos testes precisarão ser feitos.
g) Compreensibilidade: Quanto mais informações estiverem disponíveis para o entendimento do software, mais eficazes serão os testes, isso inclui manuais organizados, detalhados e especificados.

A atividade de teste é composta por alguns elementos essenciais que auxiliam na formalização desta atividade, os quais serão apresentados a seguir.

a) **Caso de Teste:** descreve uma condição particular a ser testada e é composto por valores de entrada, restrições para a sua execução e um resultado ou comportamento esperado. (CRAIG; JASKIEL, 2002).
b) **Procedimento de Teste:** é uma descrição dos passos necessários para executar um caso (ou um grupo de casos) de teste. (CRAIG; JASKIEL, 2002).
c) **Critério de Teste:** serve para selecionar e avaliar casos de teste de forma a aumentar as possibilidades de provocar falhas ou, quando isso não ocorre, estabelecer um nível elevado de confiança na correção do produto. (ROCHA et al., 2001). Os critérios de teste podem ser utilizados como:
    a. **Critério de Cobertura dos Testes:** permite a identificação de partes do programa que devem ser executadas para garantir a qualidade do software e indicar quando o mesmo foi suficientemente testado. (RAPPS; WEYUKER, 1982). Ou seja, determinar o percentual de elementos necessários por um critério de teste que foram executados pelo conjunto de casos de teste.
    b. **Critério de Adequação de Casos de Teste:** Quando, a partir de um conjunto de casos de teste T qualquer, ele é utilizado para verificar se T satisfaz os requisitos de teste estabelecidos pelo critério. Ou seja, este critério avalia se os casos de teste definidos são suficientes ou não para avaliação de um produto ou uma função. (ROCHA et al., 2001).
    c. **Critério de Geração de Casos de Teste:** quando o critério é utilizado para gerar um conjunto de casos de teste T adequado para um produto ou função, ou seja, este critério define as regras e diretrizes para geração dos casos de teste de um produto que esteja de acordo com o critério de adequação definido anteriormente. (ROCHA et al., 2001).

6.6.1 NÍVEIS DE TESTE DE SOFTWARE

O planejamento dos testes deve ocorrer em diferentes níveis e em paralelo ao desenvolvimento do software (Figura 4). Segundo Rocha et al. (2001) definimos que os principais níveis de teste de software são:

a) **Teste de Unidade:** também conhecido como teste unitário. Tem por objetivo explorar a menor unidade do projeto, procurando provocar falhas ocasionadas por defeitos de lógica e de implementação em cada módulo, separadamente. O universo alvo desse tipo de teste são os métodos dos objetos ou mesmo pequenos trechos de código.

b) **Teste de Integração:** visa provocar falhas associadas às interfaces entre os módulos quando esses são integrados para construir a estrutura do software que foi estabelecida na fase de projeto.

c) **Teste de Sistema:** avalia o software em busca de falhas por meio da utilização do mesmo, como se fosse um usuário final. Dessa maneira, os testes são executados nos mesmos ambientes, com as mesmas condições e com os mesmos dados de entrada que um usuário utilizaria no seu dia-a-dia de manipulação do software. Verifica se o produto satisfaz seus requisitos.

d) **Teste de Aceitação:** são realizados geralmente por um restrito grupo de usuários finais do sistema. Esses simulam operações de rotina do sistema de modo a verificar se seu comportamento está de acordo com o solicitado.

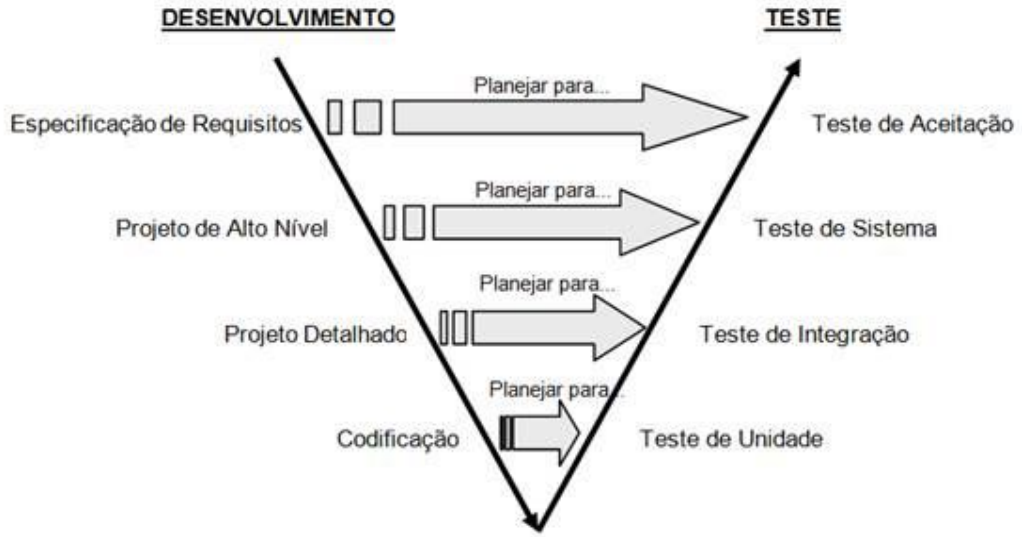
e) **Teste de Regressão:** Teste de regressão não corresponde a um nível de teste, mas é uma estratégia importante para redução de “efeitos colaterais”. Consiste em se aplicar, a cada nova versão do software ou a cada ciclo, todos os testes que já foram aplicados nas versões ou ciclos de teste anteriores do sistema. Pode ser aplicado em qualquer nível de teste.

Dessa forma, em conformidade com a Figura 8, o planejamento e projeto dos testes devem ocorrer de cima para baixo, ou seja:

1. Inicialmente é planejado o teste de aceitação a partir do documento de requisitos;

2. após isso é planejado o teste de sistema a partir do projeto de alto nível do software;
3. em seguida ocorre o planejamento dos testes de integração a partir o projeto detalhado;
4. e por fim, o planejamento dos testes a partir da codificação.

Figura 6 – Modelo V descrevendo o paralelismo entre as atividades de desenvolvimento e teste de software



Fonte: (Craig e Jaskiel, 2002).

6.7 TÉCNICAS DE TESTE DE SOFTWARE

Atualmente existem muitas maneiras de se testar um software. Mesmo assim, existem as técnicas que sempre foram muito utilizadas em sistemas desenvolvidos sobre linguagens estruturadas que ainda hoje têm grande valia para os sistemas orientados a objeto. Apesar dos paradigmas de desenvolvimento serem diferentes, o objetivo principal destas técnicas continua a ser o mesmo: encontrar falhas no software.

De acordo com Claudio ([201-]), as técnicas de teste são classificadas de acordo com a origem das informações utilizadas para estabelecer os requisitos de teste. Elas contemplam diferentes perspectivas do software, e impõe-se a

necessidade de se estabelecer uma estratégia de teste que contemple as vantagens e os aspectos complementares dessas técnicas. As técnicas existentes são: técnica funcional e estrutural.

Outras técnicas de teste que podem e devem ser utilizadas de acordo com necessidades de negócio ou restrições tecnológicas são: teste de desempenho, teste de usabilidade, teste de carga, teste de stress, teste de confiabilidade e teste de recuperação.

Segundo Claudio ([201-]), alguns autores chegam a definir uma técnica de teste caixa cinza, que seria um mesclado do uso das técnicas de caixa preta e caixa branca, mas, como toda execução de trabalho relacionado à atividade de teste utilizará simultaneamente mais de uma técnica de teste, é recomendável que se fixe os conceitos primários de cada técnica.

7 ALGORITMOS

Para desenvolver qualquer sistema de software, não basta escrever códigos numa linguagem de programação, é necessário saber montar algoritmo. Ascencio (1999) define este como "*(...) a descrição de uma sequência de passos que deve ser seguida para a realização de uma tarefa.*" De forma mais específica para a programação, Salvetti (1999) também defende que algoritmo é uma sequência de passos/instruções, porém para resolver um problema computacional.

Os passos para a construção de qualquer algoritmo, segundo Ascencio (2009), são os seguintes:

a) Compreender o problema a ser resolvido, dando destaque aos objetos que o integram e os pontos mais importantes.
b) Determinar quais serão os dados de entrada, ou seja, os dados que serão fornecidos ao sistema.
c) Definir quais dados de saída.
d) Definir o processamento, ou seja, os cálculos que serão realizados com os dados de entrada e quais objetos serão responsáveis por cada cálculo. O processamento gerará os dados de saída.
e) Construir o algoritmo.

f) Testar o algoritmo.

7.1 PARADIGMA DE POO

A Programação Orientada a Objeto é um dos paradigmas mais utilizados na atualidade. Nele, segundo Ascencio (2009), uma coleção de objetos interagem entre si. Diferentes objetos que possuam os mesmos tipos de informações e os mesmos comportamentos podem ser definidos com sendo de uma mesma classe. Por exemplo, se for definido dois carros num programa, eles possuirão os mesmos tipos de informação (quantidade de rodas, quantidade de portas, dimensões) e os mesmos comportamentos (mover para frente, mover para trás, girar a roda, trocar de marcha), então poderão ser definidos como sendo da mesma classe.

De acordo com Horstmann (2010), a definição de classes permite a redução da quantidade de linhas de código e facilita a manutenção do sistema de software.

7.2 A LINGUAGEM JAVA

Horstmann (2010) nos conta que, em 1991 na empresa Sun Microsystems, uma equipe liderada por James Gosling e Patrick Naughton desenvolveram uma linguagem de programação, codinome “Green” com a finalidade de ser utilizada em dispositivos de usuários. Eles a criaram para que fosse simples e que um mesmo código-fonte não necessitasse de alterações para funcionar em diferentes hardwares. Segundo a Oracle, a equipe planejou que fosse utilizada em televisores e, por isso, negociaram com a industria de televisão a cabo, porém sem sucesso.

Em 1994, de acordo com Horstmann (2010), surgiu a ideia de criar um navegador e então este foi apresentado, em 1995, na SunWorld, com a linguagem já denominada Java. No mesmo ano o navegador Netscape incorporou esta nova tecnologia.

Após isto, a linguagem cresceu tanto que, segundo Cass (2015), após comparar 10 diferentes fontes, contata-se que Java se tornou a mais popular, em 2014 e 2015, quando comparada com 39 outras linguagens de programação.

8 ANDROID

O sistema operacional Android foi originalmente desenvolvido pela empresa Android Inc, que foi adquirida, segundo Deitel (2013), pela Google em 2005. Em 2007, liderada pela Google, foi criada em a Open Handset Alliance, num esforço que uniu 34 empresas - atualmente 84 (open handset alliance faq) - com o propósito de desenvolver o Android para que se tornasse um sistema comum para dispositivos móveis, reduzindo custos e melhorando a experiência do usuário.

Em outubro de 2008 foi lançado o primeiro telefone Android. 7 anos depois, segundo o site NetMarketShare, estava presente em 51,76% dos dispositivos móveis, enquanto o segundo lugar ficou com o iOS, com 39,73%.

9 BANCOS DE DADOS RELACIONAIS

Os bancos de dados relacionais são os mais comumente utilizados. Segundo Elmasri (2011) o modelo de dados relacionados foi introduzido primeiramente por Ted Codd, pesquisador da IBM, em 1970, atraindo atenção por sua simplicidade e sua fundação na matemática. De forma simplificada, neste modelo os dados são armazenados em tabelas com linhas e colunas. Cada linha da tabela representa uma coleção de valores de dados. Os nomes da tabela e das colunas ajudam a dar significado aos valores da linha. (ELMASRI, 2011)

De acordo com a Oracle (2016), uma tabela relacional deve seguir algumas regras de integridade para evitar erros e problemas:

a) Cada linha da tabela deve ser distinta. Se houver linhas duplicadas poderá haver problemas na recuperação de dados.

b) As colunas não devem armazenar vetores ou matrizes.

c) O terceiro aspecto em relação a integridade de dados envolve o conceito de valor nulo (*null*). Quando um valor estiver faltando, o banco de dados demonstra isso com o valor nulo. Nulo não se iguala com um campo em branco e nem com zero. Um campo em branco se iguala como outro campo em branco, zero se iguala com outro zero, porém nulo não se iguala com outro nulo.

Quando todas as linhas de uma tabela são diferentes, é possível utilizar uma ou mais colunas para identificar uma linha. Esta coluna (ou grupo de colunas) recebe o nome de chave primária (*primary key*). Qualquer coluna que integre a chave primária não poderá ter valor nulo. (ORACLE, 2016). Como diferentes pessoas podem ter o mesmo nome, não é recomendável que a coluna deste dado seja utilizada para este fim. Na figura 9 a coluna *Employee_Number* seria o melhor candidato a chave primária.

Figura 9 - . exemplo de tabela

Employees Table

Employee_Number	First_name	Last_Name	Date_of_Birth	Car_Number
10001	John	Washington	28-Aug-43	5
10083	Arvid	Sharma	24-Nov-54	null
10120	Jonas	Ginsberg	01-Jan-69	null
10005	Florence	Wojokowski	04-Jul-71	12
10099	Sean	Washington	21-Sep-66	null
10035	Elizabeth	Yamaguchi	24-Dec-59	null

Fonte: https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html

9.1 A LINGUAGEM SQL

A linguagem SQL (*Structured Query Language*) é a mais utilizada linguagem de comunicação entre um software e um banco de dados estruturado. Os órgão responsáveis pela sua padronização são a ANSI (*American National Standards*

Institute) e a ISO (*International Standards Organization*), que em 1986 liberaram sua primeira versão, conhecida como SQL-86 ou SQL1. A primeira revisão aconteceu em 1992 e é denominada SQL-92, também conhecida como SQL2. Após essas, vieram a SQL:2003, SQL:2006 e uma ultima atualização em 2008 (ELMASRI, 2011).

Apesar de haver diferentes sistemas de banco de dados relacional (MySQL, PostgreSQL, Microsoft SQL, SQLite, Oracle), a padronização permite que uma aplicação escrita para um seja facilmente migrável para outro (ELMASRI, 2011).

9.1.1 PRINCIPAIS COMANDOS SQL

Para criar uma tabela, é utilizado o comando CREATE. Um exemplo do uso deste comando, apresentado na documentação online do MySQL, está exemplificado a seguir:

CREATE TABLE pet (name VARCHAR(20), owner VARCHAR(20), species VARCHAR(20), sex CHAR(1), birth DATE, death DATE);

Esta tabela recebe o nome *pet* e possui como colunas *name, owner, species* (que armazenam até 20 caracteres cada), *sex* (1 caractere), *birth* e *death* (que armazenam uma data cada).

Para inserir dados na tabela é utilizado o comando INSERT. O exemplo a seguir, retirado da mesma documentação, mostra sua utilização:

INSERT INTO pet VALUES ('Puffball','Diane','hamster','f','1999-03-30',NULL);

Este comando está inserindo na tabela *pet* o valores *Puffball', 'Diane', 'hamster', 'f', '1999-03-30', NULL.* Como não foi definido a coluna de cada valor, será inserido na mesma ordem que foram criadas as colunas.

O comando UPDATE atualiza os dados, caso os encontre:

UPDATE pet SET birth = '1989-08-31' WHERE name = 'Bowser';

Neste exemplo será atualizado, na tabela *pet*, a data de nascimento de todos os animais que possuam o nome Bowser.

Para o usuário visualizar os dados na tabela é utilizado, segundo ELMASRI (2011), o comando SELECT.

SELECT name FROM pet WHERE species = 'snake' OR species = 'bird';

Será mostrado para o usuário o nome (*name*) de todos os animais da tabela *pet* cuja espécie seja *snake* ou *bird*.

9.3 SQLITE

A função de bancos de dados SQL, segundo o site do SQLite (2016), é implementar um repositório compartilhado de dados empresariais, onde há uma grande preocupação com a escalabilidade, concorrência, centralização e controle. O SQLite possui um foco diferente. Ele tem como finalidade prover o armazenamento local de dados para aplicações e dispositivos. Por isso a preocupação maior está em economia, eficiência, confiabilidade, independência e simplicidade.

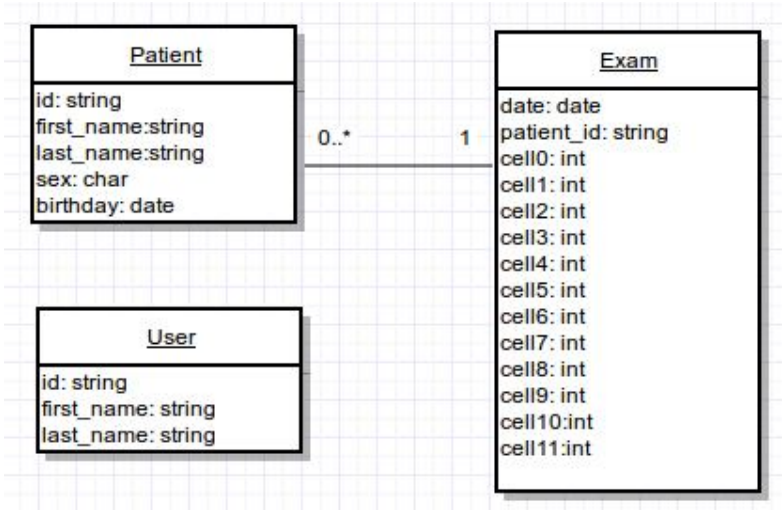
Bancos de dados SQLite não requer um administrador de banco de dados, ele funciona bem aonde não há a necessidade de um especialista no assunto, como em celulares, televisores, consoles de jogos, câmeras fotográficas, relógios, carros, aviões, entre tantos outros. O sistema operacional Android é um dos que utilizam SQLite.

10 METODOLOGIA

Para o desenvolvimento deste projeto, foi realizado um levantamento bibliográfico, analisando quais os equipamentos utilizados atualmente para a criação de um leucograma e como poderiam ser adaptados os deixando mais acessíveis.

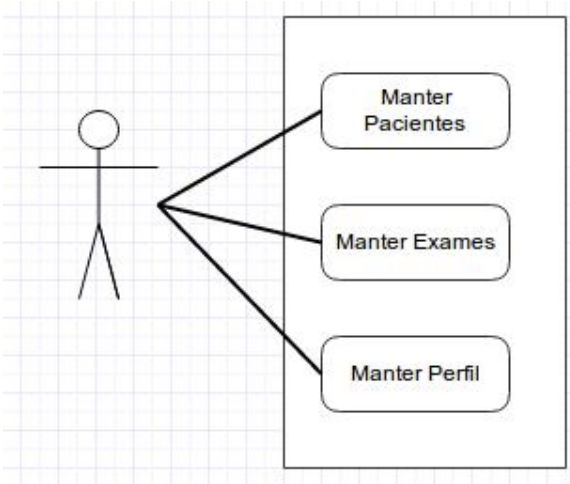
Com estas informações, foi montado os Diagramas de Classe, de Atividade e de Caso de Uso para auxiliar o desenvolvimento do sistema. O software escolhido para a criação destes diagramas da UML foi o Umbrello, pois gera um resultado visualmente agradável, é totalmente compatível com sistemas Linux e é gratuito.

Figura 10: Diagrama de classe do aplicativo



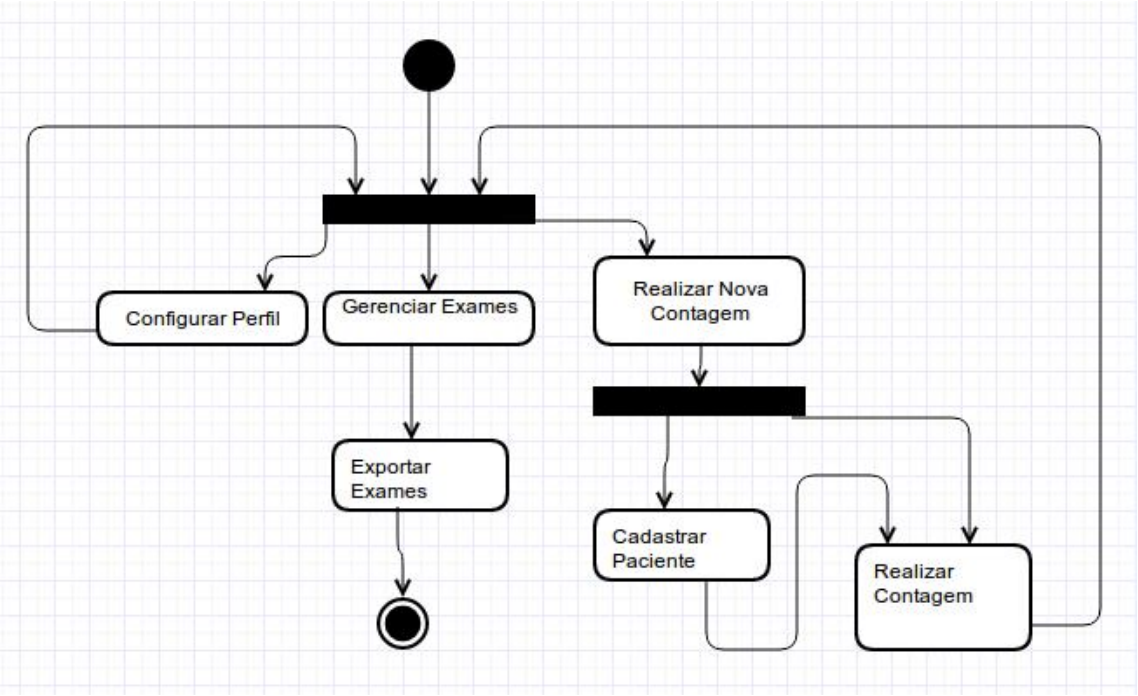
Fonte: Elaborado pelo autor

Figura 11: Diagrama de caso de uso do aplicativo



Fonte: Elaborada pelo autor

Figura 12: Diagrama de atividade



Fonte: Elaborada pelo autor

O próximo passo foi o desenvolvimento do software. Como a grande maioria dos universitários possuem smartphones, foi definido que este seria voltado para dispositivos móveis. Entre os sistemas operacionais voltados para este nicho, a opção escolhida foi o Android, pois como é o sistema portátil mais utilizado no Brasil, permitirá alcançar um público maior.

O desenvolvimento em si ocorreu no IDE (*Integrated Development Environment* – Ambiente de Desenvolvimento Integrado) Android Studio, pois além de ser o recomendado e mantido pela Google, ele é desenvolvido com o foco em Android, diferentemente de outros IDEs, que possuem um foco mais geral. Isto permite que possua ferramentas especificas que auxiliam no desenvolvimento do sistema.

Como o Android foi inicialmente desenvolvido para utilizar a linguagem Java, a IDE escolhida é nativamente compatível com esta, além de ser a linguagem que mais possui documentação voltada para o desenvolvimento de aplicações Android, naturalmente foi Java a linguagem de programação escolhida.

O hardware que foi utilizado no desenvolvimento foi um notebook com processador Intel i5-3210M e 8GB de memória RAM. O sistema operacional foi o Linux Ubuntu 14.04 64 bits, pois além de ser gratuito e nativamente compatível com o Android Studio, possui o suporte de uma grande empresa (Canonical) até abril de 2019 (quando acabará o suporte para esta versão em questão e será recomendado a atualização para uma versão mais recente).

Para testes durante o desenvolvimento, foi utilizado o smartphone Nexus 4, com o Android 6.0.1, e diversos emuladores disponíveis no Android Studio, permitindo visualizar o software em diversos tamanhos de tela e versões do sistema Android, garantindo assim que o produto final se tornou compatível com a maioria dos smartphones atualmente em uso.

O banco de dados é 100% local. Para isso foi utilizado o SQLite, pois além de ser desenvolvido para esta finalidade, todo smartphone Android já possui ele em seu sistema, facilitando assim a instalação e desenvolvimento do software.

A utilização de um banco de dados permite aos usuários armazenarem diversos resultados de contagens possibilitando a comparação não apenas entre diversos pacientes, mas também, no caso de uma sala de aula, diversos alunos compararem seus resultados de um mesmo paciente.

O software funciona de forma a auxiliar a contagem dos leucócitos. O usuário, utilizando um microscópio, visualiza uma amostra e, para cada tipo de glóbulo branco que enxergar, aperta o botão correspondente na interface do software. Uma imagem então aparece mostrando um exemplo do leucócito selecionado para que o usuário compare com o visto no microscópio e garantir assim que selecionou o correto. Após selecionar 100 células, uma mensagem visual e sonora é apresentada informando que já há o número mínimo para realizar uma análise. Caso deseje encerrar a contagem, a quantidade de cada tipo de leucócito será apresentado ao usuário para que este faça suas interpretações.

Com o protótipo pronto, foi testado com profissionais do curso de biomedicina da USC, aonde puderam qualificar o quão útil é e sugerir melhorias. Todas estas informações foram interpretadas e qualificadas para correções e alterações.

Os resultados finais e o software serão divulgados em eventos técnicos e científicos, enviados para publicações correlatas e serão apresentados no

Congresso Anual de Iniciação Científica e Desenvolvimento Tecnológico e Inovação da USC.

11 RESULTADOS FINAIS

A seguir serão apresentados os resultados finais desta pesquisa.

11.1 FUNCIONAMENTO

Ao abrir o aplicativo, o usuário se depara com uma tela inicial que possui quatro botões:

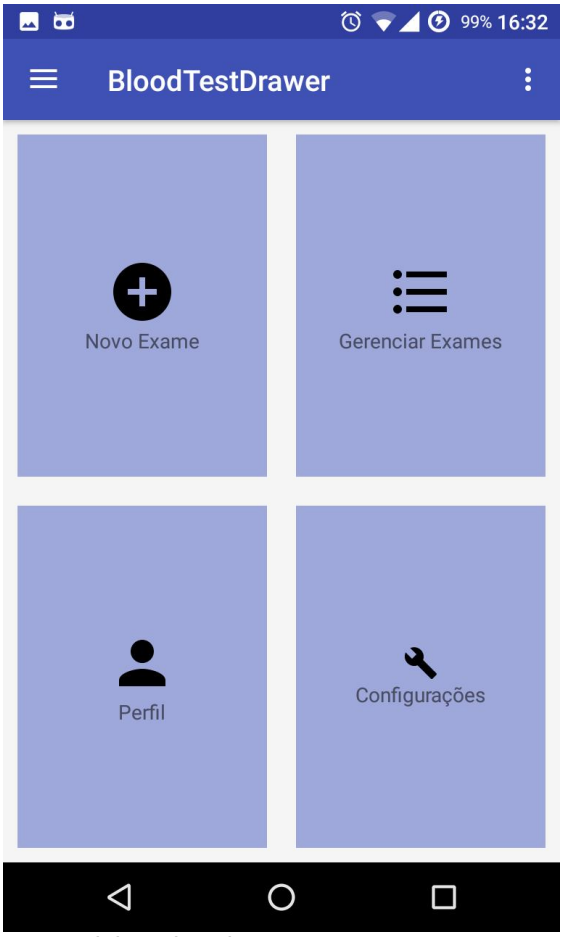
a) **Novo Exame:** permite realizar uma nova contagem com os pacientes já cadastrados. Caso seja um paciente novo, é possível iniciar cadastrá-lo nesta mesma tela.

b) **Gerenciar Exames:** onde é possível visualizar os exames já realizados, exportá-los e/ou os excluir.

c) **Perfil:** nesta tela o usuário insere alguns poucos dados que o identifique. Estes dados serão utilizados apenas no momento de geração do arquivo de exportação, para assim, quem o receber, saber quem enviou.

d) **Configurações:** é definido aqui se durante a contagem, os botões terão som, se vibrarão e se mostrarão imagem de célula.

Figura 13: Tela inicial



Fonte: Elaborada pelo autor.

Além destes botões estarem disponíveis na tela inicial (Figura 13), estão também acessíveis a partir de qualquer tela (exceto durante o exame) através de um menu lateral chamado de *drawer*.

Como pode ser observado na Figura 14, o cadastro do usuário solicita apenas o primeiro nome, o sobrenome e uma identificação. Estes dados são utilizados apenas para, na geração do arquivo de exportação, identificar quem foi o responsável pelos exames. Não é obrigatório este cadastro, porém é uma forma de auxiliar o educador a saber qual aluno fez qual contagem.

O campo id aceita qualquer tipo de dado. Pode ser então um nome, um número que identifica o aluno, ou um CPF, entre outros.

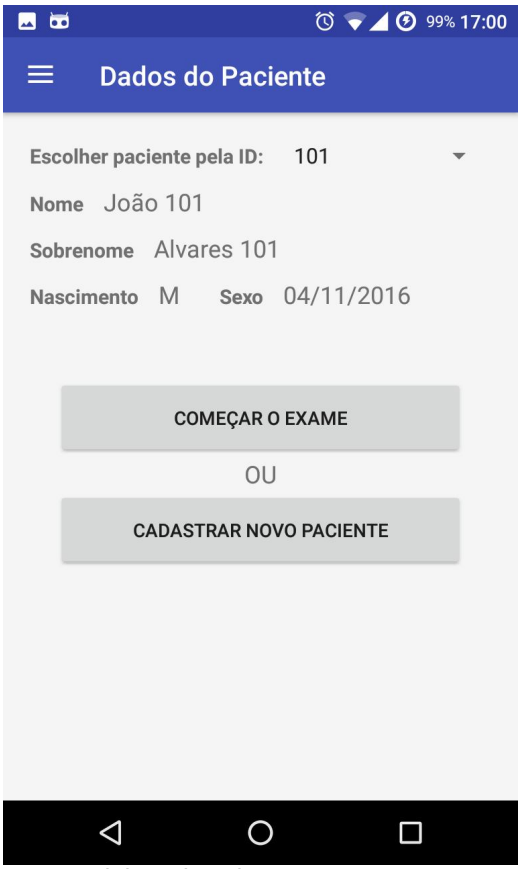
Figura 14: Cadastro do perfil do usuário

Fonte: Elaborada pelo autor.

Ao acessar o botão “Novo Exame”, será apresentado ao usuário a opção de escolher (através de uma identificação id) qual paciente será utilizado para realizar a contagem. Caso não haja paciente algum cadastrado ou deseje realizar a contagem de células de um novo paciente, o botão “Cadastrar Paciente” abrirá uma nova tela permitindo que seja feito o mesmo.

Figura 15: Escolha do paciente

Figura 16: Cadastro do paciente



Fonte: Elaborada pelo autor.

Fonte: Elaborada pelo autor.

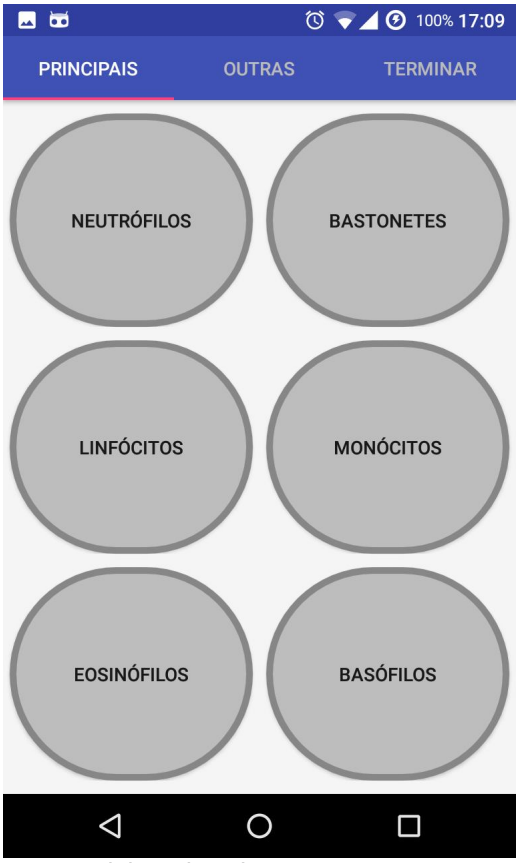
Assim como no cadastro do usuário, o id do paciente permite que seja inserido qualquer caractere alfanumérico, permitindo, por exemplo, que um aluno preencha como “Exercício 3”.

Ao iniciar o exame, o usuário encontra uma tela com três abas, aonde a primeira mostra as células mais comuns encontradas nestes exames, a segunda células menos comuns e, na terceira, a contagem parcial e um botão para finalizar a contagem, que só fica ativado após a contagem alcançar, no mínimo, 100 células.

Ao clicar em um botão, uma imagem representando a célula é mostrada, auxiliando o estudante a identificar se selecionou corretamente a célula. Um som diferente é emitido para cada tipo celular para que o usuário saiba se selecionou a célula correta, sem precisar tirar os olhos do microscópio.

Figura 17: Seleção de célula

Figura 18: Apresentação da célula

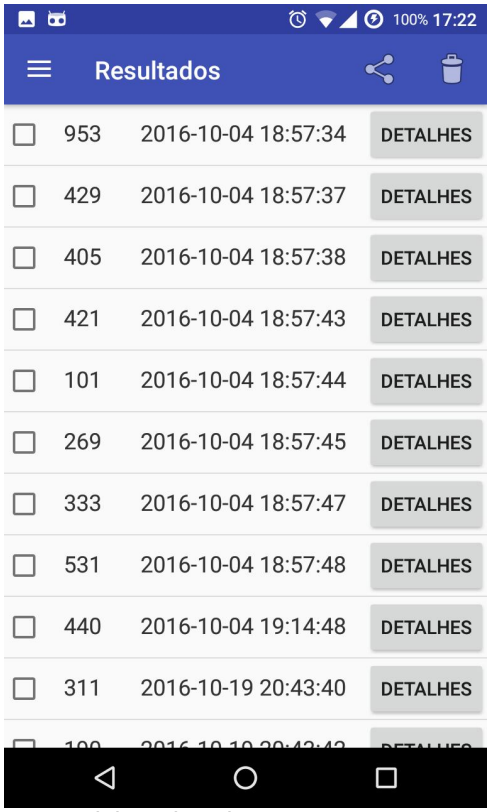


Fonte: Elaborada pelo autor.

Fonte: Elaborada pelo autor.

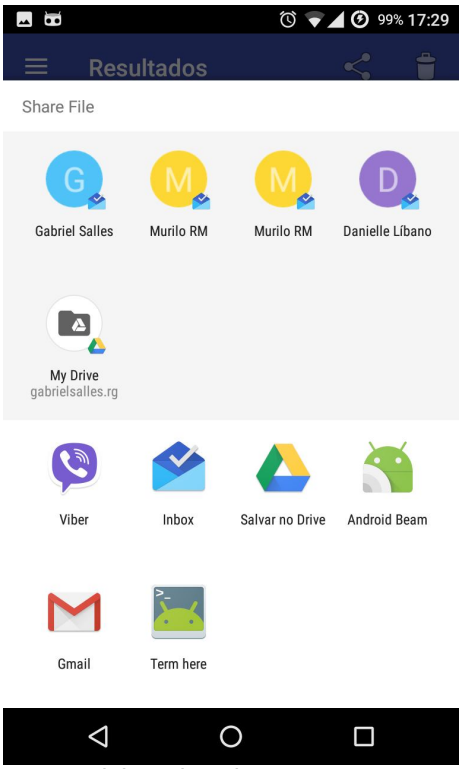
Ao acessar “Gerenciar Exames”, é mostrado ao usuário todos os exames cadastrados, identificados pela “id” do paciente e a data. É possível, através desta tela, além de acessar a contagem feita, excluir exames e/ou enviá-los para algum destino que convêm (o e-mail da professora, por exemplo).

Figura 19: Lista de resultados



Fonte: Elaborada pelo autor.

Figura 20: Exportação de dados



Fonte: Elaborada pelo autor.

Figura 21: Dados exportados em CSV

	A	B	C
1	Usuário	Fulano de Tal	
2	ID	586338	
3			
4			
5	Paciente	João 429 Alvares 429	
6	ID	429	
7			
8	Neutrófilos	10	
9	Bastonetes	10	
10	Linfócitos	10	
11	Monócitos	10	
12	Eosinófilos	10	
13	Basófilos	10	
14	Blasto	10	
15	Promielócito	10	
16	Mielócito	10	
17	Metamielócito	10	
18	Eritoblasto	10	
19			
20			
21	Paciente	João 101 Alvares 101	
22	ID	101	
23			
24	Neutrófilos	10	
25	Bastonetes	10	
26	Linfócitos	10	
27	Monócitos	10	
28	Eosinófilos	10	
29	Basófilos	10	
30	Blasto	10	
31	Promielócito	10	
32	Mielócito	10	
33	Metamielócito	10	
34	Eritoblasto	10	
35			
36			
37	Paciente	João 269 Alvares 269	
38	ID	269	
39			
40	Neutrófilos	10	
41	Bastonetes	10	

Fonte: Elaborada pelo autor.

O arquivo é enviado em formato CSV (*comma separated values*, valores separados por vírgula), um padrão que pode ser aberto em diversas planilhas eletrônicas, como o Ms Office Excel, LibreOffice Calc e Google Sheets.

12 CONSIDERAÇÕES FINAIS

A idealização deste projeto iniciou com o encontro da profª e coordenadora do curso de Biomedicina da Universidade do Sagrado Coração – USC, Mª Daniela Barbosa Nicolielo, e o professor do curso de Ciência da Computação, Dr Elvio Gilberto da Silva, e foi assim encontrada uma necessidade na área da saúde que poderia ser suprida graças aos avanços da computação nas últimas décadas.

O desenvolvimento de um aplicativo mobile não apenas pode substituir os equipamentos mais comuns, como também podem oferecer funcionalidades extras, assim como foi explorado neste projeto. Como tanto o profissional da saúde, quanto o estudante podem utilizar o próprio celular para fazer uso do software em questão, é reduzido a zero os custos de aquisição de novos equipamentos e manutenção dos atuais, deixando assim mais acessível a manutenção de laboratórios e permitindo que os cursos da área de saúde possam direcionar os recursos para outros focos.

Sugere-se como trabalhos futuros alterar o aplicativo para que ele não sirva apenas para o auxílio na criação de um leucograma, mas também de um eritograma e de outros tipos de contagem que possuam necessidades semelhantes.

13 REFERÊNCIAS

Appropriate Uses For SQLite Disponível em: <https://www.sqlite.org/whentouse.html> Acesso em 21/05/2016

About SQLite Disponível em: <https://www.sqlite.org/about.html> Acesso em 21/05/2016

Analytics Without the Bots - Mobile/Tablet Operating System Market Share Disponível em: <https://www.netmarketshare.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qpsp=2015&qpnp=1&qptimeframe=Y&qpcustom=> Acesso em 10/04/2016

A Relational Database Overview Disponível em: <https://docs.oracle.com/javase/tutorial/jdbc/overview/database.html> Acesso em 20/05/2016

ASCENCIO, A.F.G., CAMPOS, E.A.V. de, **Fundamentos da Programação de Computadores**, Pearson, 2 edição, 2009

ASCENCIO, A.F.G. **Lógica de programação com Pascal**. São Paulo: Pearson Education/Makron Book, 1999

BARBOSA, S.D.J, SILVA, B.S.; **Interação Humano-Computador**, 2010, Editora Campus

BEZERRA E. **Princípios de Análise e Projeto de Sistemas com UML,** 3 edição, 2015

BOOCH, G., RUMBAUGH, J., JACOBSON, I. **The Unified Modeling Language User Guide**, 1999

BOOCH, G., RUMBAUGH, J., JACOBSON, I. **The Unified Modeling Language Reference Manual**, 2 edição, 2005

BAIN, B.J. **Células Sangüineas – Um Guia Prático**, 4 edição, 2007

BAIN, B.J. **Células Sangüineas – Consulta Rápida**, 1 edição, 1998

DEITEL, P., DEITEL, A., DEITEL, H., MORGANO, M.; **Android Para Programadores – Uma abordagem baseada em aplicativos**, 2013

ELMASRI, R., NAVATHE, S. B. **Database Systems – Models, Languages, Design, and Application Programming**, 6 edição, Pearson, 2011

FAILACE **Hemograma – Manual de Interpretação**, R. 5o edição, 2009

HORSTMANN, C.S., **Big Java**, 4 edition, 2010, John Wiley & Sons inc.

ISO/IEC/IEEE 24765:2010 Disponível em: <http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=50518> Acesso em: 10/05/2016

JUNQUEIRA, L.C., CARNEIRO, J. **Histologia Básica**, 11 edição, 2008

Dr. Ivan de Lucena Angulo **INTERPRETAÇÃO DO HEMOGRAMA CLÍNICA E LABORATORIAL**, Disponível em <www.sogab.com.br/hemograma2.pdf>*. Acesso em: 15 de maio de 2016*

MACIASZEK, L.A. **Requirements Analysis and System Design. Developing Information Systems with UML**, Addison Wesley, 2001

MySQL 5.7 Reference Manual Disponível em: <https://dev.mysql.com/doc/refman/5.7/en/> Acesso em 21/05/2016

open handset alliance faq Disponível em: <http://www.openhandsetalliance.com/oha_faq.html> Acesso em 15/04/2016

PRESSMAN, R. S. **Engenharia de Software – Uma abordagem Profissional**, 7 Edição, 2011,

PRESSMAN, R. S. LOWE, D. **Engenharia Web**, 2009

ROSA, J.G.S, M, A **Avaliação e projeto no design de interfaces**, 2 edição, 2012, 2AB Editora

SALVETTI, D.D.; BARBOSA, L.M.. **Algoritmos**. São Paulo: Pearson Education/Makron Books, 1999

SOMMERVILLE, I. **Engenharia de Software**, 8ª edição, 2010

SQLiteDatabase Disponível em: <https://developer.android.com/reference/android/database/sqlite/SQLiteDatabase.htm> Acesso em 21/05/2016

SWEBOKv3. Disponível em: <https://www.computer.org/web/swebok/> Acesso em: 10/05/2016

The History of Java Tecnology. Disponível em: <http://www.oracle.com/technetwork/java/javase/overview/javahistory-index-198355.html> acesso em 23/04/2016

The 2015 Top Ten Programming Languages Disponível em: <http://spectrum.ieee.org/computing/software/the-2015-top-ten-programming-languages> Acesso em 23/04/2016, Stephen Cass, 20/07/2015

WAZLAWICK, R. S. **Engenharia de Software – Conceitos e Práticas**, 2013