Universidade Federal dos Vales do Jequitinhonha e Mucuri
Faculdade de Ciências Exatas e Tecnológicas
Departamento de Computação
Bacharelado em Sistemas de Informação

Sistema Móvel de Gerenciamento de Informações Educacionais

Adalberto Vinicius Nunes

Diamantina - MG
2016

ADALBERTO VINICIUS NUNES

SISTEMA MÓVEL DE GERENCIAMENTO DE INFORMAÇÕES EDUCACIONAIS

Trabalho apresentado ao Curso de Bacharelado em Sistemas de Informação do Departamento de Computação da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação

Orientador: Prof. Dr. ALESSANDRO VIVAS ANDRADE

Diamantina - MG

2016

ADALBERTO VINICIUS NUNES

SISTEMA MÓVEL DE GERENCIAMENTO DE INFORMAÇÕES EDUCACIONAIS

Trabalho apresentado ao Curso de Bacharelado em Sistemas de Informação do Departamento de Computação da Universidade Federal dos Vales do Jequitinhonha e Mucuri como requisito parcial para obtenção do grau de Bacharel em Sistemas de Informação

Aprovada no dia 02 de Março de 2016 .

BANCA EXAMINADORA

Prof. Dr. ALESSANDRO VIVAS ANDRADE - Orientador
UFVJM

Prof. Dr. LUCIANA PEREIRA DE ASSIS
UFVJM

Prof. RAFAEL SANTIN
UFVJM

Diamantina - MG
2016

Aos meus familiares e amigos.

Agradecimentos

Agradeço primeiramente a Deus pela a dádiva da vida e por sempre nos conceder força e sabedoria para enfrestarmos as dificuldades ao longo de nossos percursos.

Agradeço aos meus pais, Ana Maria de Jesus Nunes e Adalberto Mario Nunes (in memoriam), pela educação, carinho, apoio e incentivo de sempre.

Agradeço ao meus irmãos pelo companheirismo, diversão e união em todos os momentos da minha vida, tornando essa trajetoria menos árdua.

Agradeço aos meus avós, tios, primos e amigos pelo apoio, conselhos e por sempre se fazerem presentes em minha vida.

Agradeço ao meu orientador Alessandro Vivas Andrade por ter me guiado ao longo deste trabalho, dando todo o suporte necessário.

Lista de Figuras

Lista de Tabelas

Sumário

Resumo

Atualmente o acesso à informação de forma fácil e rápida tornou-se essencial para as pessoas. A popularização dos dispositivos móveis mudou a forma como as pessoas trabalham, se comunicam e buscam informações ao redor do mundo.

O avanço na tecnologia *mobile* deu origem a um novo tipo de software, os aplicativos móveis. Através deles podemos realizar tarefas de forma prática, direta e simples. Tudo isso a qualquer hora e em qualquer lugar. A utilização dos dispositivos móveis apresenta uma série de vantagens, como facilidade na comunicação, redução de custos e otimização de tempo.

No ambiente educacional pode-se encontrar hoje uma grande quantidade de atividades rotineiras que são desempenhadas de forma manual e que consomem tempo dos profissionais. É pensando neste problema que este trabalho propõe o desenvolvimento de um Sistema Móvel de Gerenciamento de Informações Educacionais. O objetivo principal do sistema consiste em gerenciar a frequência dos alunos durante o período letivo. É uma aplicação com funcionalidades simples e práticas desenvolvida para a plataforma Android, que utiliza a linguagem de programação Java. O sistema realiza o cadastro de disciplinas e alunos por semestre, registra a presença ou ausência dos alunos em cada aula e gera um relatório semestral de frequência.

Palavras-Chaves: Aplicativos Móveis; Android; Java; Mobile; XML; Frequência; Computação;

Abstract

Currently, the access to information in an easy and quick way became essential for people. The popularity of mobile devices has changed the way people work, communicate and seek information around the world.

The advancement in mobile technology has given rise to a new type of software, the mobile applications. Through them we can perform tasks in a practical, straightforward and simple way. All of this anytime and anywhere. The use of mobile devices presents a number of advantages, such as ease of communication, reducing costs and optimizing time.

The educational environment can now find a lot of routine activities that are performed manually and are time-consuming for the workers. Thinking about this problem, this paper proposes the development of a Mobile System for Educational Information Management. The system's main objective is to manage the students attendance rate during the school year. It is an application with simple features and practices developed for the Android platform using Java programming language. The system performs the registration of courses and students per semester, records the presence or absence of students in each class and generates a frequency report for each semester.

Keywords: Mobile Applications; Android; Java; Mobile; XML; Attendance; Computing;

Capítulo 1

Introdução

A comissão de cultura, educação e esporte determina que os alunos no ensino presencial cumpram a carga horária mínima de 75% das horas letivas para que possam ser aprovados em cada disciplina (BRASIL. Lei 9.394, 1996, art. 24).

Conforme a Lei nº 9394 (BRASIL, 1996), o controle da frequência de cada aluno fica a cargo da instituição de ensino conforme o regimento e normas do sistema de ensino.

Segundo Avancini (p.96, 2011), no ensino superior o próprio aluno, maior de 18 anos, é responsável pelo zelo de sua frequência, isto significa que o aluno deve estar presente nas aulas ministradas não só para assisti-las, mas também para responder as chamadas realizadas pelo professor e assumir as consequências decorrentes das faltas injustificáveis. Sendo assim a tarefa de controle de frequência ficam a cargo três agentes: o aluno, o professor e o estado.

Ainda de acordo com Avancini (p.98, 2011), sempre que essas informações foram solicitadas pelas partes interessadas elas devem ser prestadas, ficando o professor responsável pelo registro fidedigno da frequência de cada aluno.

No ambiente educacional além do controle de frequência encontra-se uma série de atividades rotineiras que podem ser auxiliadas pelo uso de ferramentas e aplicações desenvolvidas para dispositivos como *smartphones e tablets*. Se os professores usarem esses dispositivos para, disseminar informação, agilizar e otimizar tarefas, então podemos dizer que de fato a utilização dessas tecnologias provocaram mudanças significativas neste ambiente.

Segundo Moura (2012), "o acesso a conteúdos multimídia deixou de estar limitado a um computador pessoal (PC) e estendeu-se também às tecnologias móveis (telemóvel,

PDA, Pocket PC, Tablet PC, Netbook), proporcionando um novo paradigma educacional, o mobile learning ou aprendizagem móvel, através de dispositivos móveis."

Dentre as principais características do uso dos dispositivos móveis dentro do ambiente educacional pode-se destacar: o acesso a informação, mobilidade, execução de tarefas, maior interação entre alunos e entre alunos e professores. Os alunos e professores não ficam mais restritos somente ao uso dos computadores pessoais ou dos computadores que as instituições disponibilizam em seus laboratórios. Isso possibilita melhorar os recursos para aprendizagem, acesso a conteúdos didáticos e o desenvolvimento de métodos e ferramentas inovadoras para este ambiente (SILVA E CONSOLO, 2008).

1.1 Objetivos do Trabalho

Este trabalho tem com objetivo o desenvolvimento de um software para auxiliar os professores na realização de chamadas e controle de presenças dos alunos em sala de aula. Ele consiste em um sistema móvel compatível com a plataforma Android. Dentre as funcionalidades do sistema, teremos: cadastros de disciplinas e alunos, registro de presenças e ausências dos alunos por aula, consultas e geração de um relatório final com a frequência dos alunos durante o período letivo.

Os objetivos específicos deste trabalho são:

- Automatizar tarefas rotineiras desempenhadas por professores;
- Maior mobilidade e agilidade na execução de tarefas;
- Diminuir o tempo gasto na busca de informação;
- Economia de recursos;

1.2 Motivação

Com o desenvolvimento deste aplicativo pretende-se substituir o método tradicional de efetuar as chamadas e controlar a frequência dos alunos em sala de aula. No método tradicional o professor precisa usar listas impressas em papel para fazer tais registros, este acaba que sendo processo lento e custoso, pois sempre que o professor precisar de

informação sobre a frequência de seus alunos nas aulas ele precisa recorrer a todas as listas para fazer uma contagem manual desses registros.

Com a utilização do aplicativo o sistema de lançamento de frequências estará sempre a mão do professor tanto para a realização de chamadas quanto para realizar as consultas. No momento em que se automatiza esta tarefa pode-se garantir uma melhor organização dos registros para o professor, reduzindo cuidados com armazenamento e localização física dos mesmos e aumentando a velocidade de recuperação de informação.

1.3 Organização do Trabalho

O capítulo 2 apresenta a evolução da computação móvel desde os seus primórdios até as tecnologias atuais.

O capítulo 3 é dedicado a plataforma Android, neste capítulo é exposto a organização e arquitetura do sistema e o histórico de suas versões.

No capítulo 4 são apresentadas as linguagens utilizadas no desenvolvimento do sistema, com uma breve descrição de suas características.

O capítulo 5 apresenta todos as ferramentas necessárias para o desenvolvimento do sistema.

No capítulo 6 é exposto os sistema móvel desenvolvido, bem como sua estrutura, requisitos e funcionalidades.

No capítulo 7 é apresentados os testes feitos com aplicativo de Gerenciamento de Informações Educacionais.

O capítulo 8 apresenta a conclusão e trabalhos futuros.

Capítulo 2

Computação Móvel

A computação móvel baseia-se no aumento da nossa capacidade de mover fisicamente serviços computacionais de modo que as pessoas não precisam mais ficarem restritas a dispositivos estáticos, ou seja, o computador torna-se um dispositivo sempre presente, isso expande a capacidade de um usuário utilizar os seus serviços, independentemente de sua localização (ARAUJO, 2003).

Segundo Loureiro (1995, p. 01):

> A computação móvel representa um novo paradigma computacional, que surge como uma quarta revolução na computação, antecedida pelos grandes centros de processamento de dados da década de sessenta, o surgimento dos terminais nos anos setenta e as redes de computadores na década de oitenta. Este novo paradigma permite que usuários desse ambiente tenham acesso a serviços independente de onde estão localizados, e o mais importante, independente de mudanças de localização, ou seja, mobilidade.

Ainda segundo Loureiro (1995), a computação móvel é sustentada por três pilares importantes, mobilidade, processamento e comunicação sem fio.

A mobilidade é a parte essencial da computação móvel, ela possibilita que usuários portadores de dispositivos móveis possam se comunicar diretamente com seus dados mesmo que estes dados estejam hospedados remotamente ao local em que estes usuários se encontram (CAPPELLOZZA; MORAES, 2013).

Os últimos anos representaram um avanço na produção de novas tecnologias para os dispositivos móveis, aparelhos que incialmente possuíam apenas a função de chamadas, hoje são utilizados para comunicação colaborativa, comunicação via satélite, redes locais e sem fio, entre outros. Com essa evolução os dispositivos sofreram transformações em

relação às arquiteturas de hardware (parte física que constitui o computador) e software (parte lógica cuja função é fornecer instruções para o hardware), como por exemplo, os sistemas operacionais (COSTA; FILHO; DUARTE, 2012).

O sistema operacional é um conjunto de programas com a função de gerenciar os recursos de hardware e software e fornecer uma interface ao o usuário final. Os sistemas operacionais variam de acordo com a realização de tarefas. Sistemas operacionais de computadores de grande porte (mainframes) são projetados principalmente para otimizar a utilização do hardware. Sistemas operacionais para computadores pessoais aceitam aplicações comerciais, jogos complexos, entre outros. Já o sistema operacional para dispositivos móveis são projetados para prover um ambiente em que usuário possa se comunicar de forma fácil e objetiva com o dispositivo no momento de executar os programas (SILBERSCHATS et al., 2004).

Falaremos a seguir sobre alguns tipos de dispositivos móveis, descreveremos suas características e funcionalidades principais.

2.1 Dispositivos Móveis

Nas útimas décadas pôde-se perceber o aumento significativo na utilização de dispositivos móveis para as mais diversas finalidades. Dispositivos como PDAs (Personal Digital Assistants), celulares, smartphone e tablets estão cada vez mais presentes no dia-a-dia das pessoas. Cada dispositivo deste pertence a sua vanguarda e é fabricado com uma finalidade específica. Com o aumento do poder computacional desses dispositivos tornou-se possível o surgimento de novos ambientes de programação voltados para estes tipos de equipamentos, onde é viável desenvolver aplicações praticamente independentes de dispositivo e fabricante (OLIVEIRA E MEDINA, 2007).

2.1.1 Telefone Celular

A telefonia móvel teve inicio nos anos 80, era baseada em tecnologia analógica e transmitia para uma região geográfica limitada. Os aparelhos celulares se limitavam apenas a realizar e receber chamadas e não eram muito portáteis, pois eram pesados e tinham dimensões grandes, essa fase foi denominada 1G (1ª Geração). Á medida que os dispositivos de comunicação tornavam-se mais compactos e a tecnologia digital começava a

se destacar no cenário mundial das telecomunicações, surgiu a 2ª geração, ou 2G, tornando possível não só a transmissão de voz mas também de dados além alcançarem maiores distâncias (AGUIAR, 2002).

Seguindo a evolução da telefonia móvel surge então a geração intermediária, 2,5G, esta consegue atingir velocidades maiores na transmissão de dados, utilizando GPRS[1] (General Packet Radio Service) ou EDGE[2] (Enhanced Data Rate for Global Evolution). A terceira geração, 3G, é marcada por um aumento significativo na taxa de transmissão de dados em relação a gerações anteriores, além de apresentar suporte a serviços e aplicações avançados, como acesso a internet em alta velocidade e serviços de multimídias. A quarta geração, 4G, garantem conexões ainda mais rápidas nos celulares e modem sem fio, sendo possível assistir vídeos em alta definição e realizar videoconferência (GUTIERREZ; CROSSETTI, 2003).

Na figura 2.1 pode-se observar a evolução dos telefones móveis desde a década de 80 até os dias atuais.

Figura 2.1: Evolução dos telefones móveis

2.1.2 PDA(Personal Digital Assistant)

São handhelds (dispositivos de mão) criados com o objetivo de serem organizadores pessoais. Seus usuários podem facilmente manter e consultar dados pessoais em qualquer

[1]GPRS é uma tecnologia que tem o objetivo de aumentar as taxas de transferência de dados entre celulares, facilitando a comunicação e o acesso a redes.

[2]EDGE é uma tecnologia que permite melhorar a transmissão de dados e aumentar a confiabilidade da mesma.

lugar e a qualquer momento, pois os dispositivos têm tamanho bastante reduzido, cabem no bolso e podem ser operados na palma da mão. Com o PDA os usuários podem coletar e armazenam informações, gerenciar contatos, acessar a Internet, utilizar serviço de e-mail, entre outros (FIGUEIREDO; NAKAMURA, 2015).

Os PDAs possuem capacidade razoável de processamento, E/S e comunicação em rede sem fio. Alguns agregam funções multimídia, com capacidade de reproduzir áudio, vídeo e tirar fotos e possuem interfaces de rede sem fio embutida. Na Figura 2.2 pode-se observar um modelo de PDA.

Segundo Oliveira (2007) equipamentos como PDAs, por possuírem poder computacional e de armazenamento superiores que os telefones celulares, permitem a execução de objetos educacionais mais elaborados e até o armazenamento de conteúdos afins.

Figura 2.2: Personal Digital Assistant

2.1.3 Smartphone e Tablet

Segundo Voltolini (2014) os smartphones tradução literal, telefones inteligentes, começaram a surgir na década de 90, as empresas precursoras deste tipo de dispositivos

foram a Apple[3] e a IBM[4]. Esses dispositivos são aparelhos móveis, diferentemente dos celulares eles possuem maior capacidade de processamento, armazenamento e execução múltiplas tarefas. As plataformas de smartphone têm suporte para touchscreen (tela sensível ao toque), teclados e permite navegar na internet, executar aplicativos, visualizar fotos, jogar, ouvir música e ver filmes.

Como exemplo de sistemas operacionais para smartphones temos: Google - Android, Apple - iOS, Microsoft - Windows Phone, Mozilla - Firefox OS, RIM - Blackberry OS e Nokia - Symbian. Com eles os usuários têm acesso a uma quantidade maior de aplicativos além de permitirem que desenvolvedores do mundo todo criem e comercializem aplicações com diversas funcionalidades e que atendam os mais variados tipos de usuários. Na Figura 2.3 alguns modelos mais atuais de Smartphones.

Figura 2.3: Modelos de Smartphones

De acordo com Cunha (2012), o tablet é um dispositivo portátil que apresenta mais capacidade e recurso que os smartphones, mas são menos potentes que os computadores pessoais. A característica marcante do tablet é sua tela com maior dimensão, variando de 7 a 10 polegadas, sensível ao toque dos dedos. Esta tela pode ser de dois tipos: tela capacitiva que apresenta maior sensibilidade ao toque e portanto uma melhor resposta e tela resistiva que apresenta menor precisão no reconhecimento do toque, já que é composta

[3]A Apple é uma empresa multinacional norte-americana que tem o objetivo de projetar e comercializar produtos eletrônicos de consumo, software de computador e computadores pessoais

[4]IBM é a sigla de International Business Machines, que significa Máquinas de Negócio Internacionais, e é uma empresa americana que trabalha com produtos voltados para a área de informática, como computadores, hardwares e softwares.

por várias camadas.

Segundo Moreira (2013), os dispositivos sensíveis ao toque são mais intuitivos, pois os usuários possuem o controle do equipamento na ponta dos dedos. Através de gestos simples eles conseguem executar tarefas sem a necessidade de ficarem procurando pelas opções dentro de menus. Sendo assim a tela sensível ao toque garante uma interação mais amigável entre o usuário e o dispositivo.

Na Figura 2.4 da esquerda pra direita nos temos: um iPad (Apple), Galaxy Tab (Samsung) e um TouchPad (HP).

Figura 2.4: Tablets

2.2 Aplicativos Móveis

Todo este avanço nas tecnologias de computação móvel influenciou diretamente e de maneira positiva outro setor que vem crescendo muito nos últimos anos, o setor de aplicativos móveis (apps). Aplicativos Móveis são softwares desenvolvidos para dispositivos móveis como, Smartphones, PDAs e Tablets. Segundo dados do SEBRAE (2014), no mundo todo são mais 1,8 milhões de apps disponíveis para os mais diversos tipos de usuários.

Os aplicativos podem ser adquiridos através das lojas virtuais de aplicativos, como App Store, Google Play, windows phone store, entre outras. Elas são lojas online e possuem tanto aplicativos gratuitos como pagos. É necessário que o usuário tenha um cadastro na loja para adquirir os aplicativos, no caso dos aplicativos pagos é necessário um cartão de crédito associado à conta.

As funcionalidades, facilidade e custo de desenvolvimento de aplicativos, estão tornando este tipo de tecnologia cada vez mais uma opção para a automatização de tarefas e processos, dentro dos ambientes de trabalho, educacionais e de lazer das pessoas.

O mercado de apps oferece soluções robustas, capazes de gerenciar milhares de dados. Com funções diferentes, cada aplicativo pode ser desenvolvido ou adaptado à necessidade de um determinado setor. Mais do que enviar lembretes e conectar pessoas, os aplicativos podem atuar na logística, na organização das equipes, no cadastramento de consumidores e vendedores, entre outros (MOBILE PEOPLE, 2014).

Um dos pontos importantes a se definir quando a questão se trata de desenvolvimento Mobile, é sobre o tipo de aplicação que podemos desenvolver e se esse tipo irá atender as expectativas e suprir as necessidades vigentes. Temos atualmente três tipos aplicativos no mercado: Aplicativos Nativos, Aplicativos Web e Aplicativos Híbridos, cada um com suas características, capacidades e restrições.

2.2.1 Aplicativos Nativos

Os aplicativos nativos são desenvolvidos para uma plataforma específica e devem obedecer ao modelo de programação da mesma, como a linguagem, a persistência de dados, a interface, o acesso a recursos do sistema, entre outros. Este tipo de aplicativo é um executável, adquirido da loja virtual e instalado no próprio dispositivo, sendo assim esse aplicativo poderá usar todos os recursos e funcionalidades do dispositivo e do sistema operacional. Vale ressaltar que um aplicativo nativo desenvolvido para uma plataforma específica não poderá ser executado em outra (BUDIU, 2013).

Vantagens:

- Podem ser usados tanto online como offline.
- Podem aproveitar ao máximo os recursos dos dispositivos (Processamento, Memória, Sistemas de Arquivos, Câmera, GPS, Lista de Contatos, etc).
- Executam com maior velocidade por serem desenvolvidos na linguagem nativa da plataforma e por poderem explorar ao máximo os seus recursos.
- O aplicativo pode ser acessado com apenas um clique sobre o ícone na tela do dispositivo.

2.2.2 Aplicativos Web

Os aplicativos Web diferentemente dos nativos são construídos usando tecnologias e padrões da Web, e executados através dos navegadores, via Internet. São desenvolvidos na linguagem HTML5[5], que permite a criação de Aplicativos Web bem próximos dos Nativos. Os usuários acessam o aplicativo como se tivessem acessando uma página web, os instala em seus dispositivos criando um link para servidor onde a página esta hospedada. As páginas deste tipo de aplicativo não são páginas comuns, elas precisam ser adaptadas para o ambiente em que esta atuando (Mozilla Developer, 2012).

Vantagens:

- Não estão atrelados a uma única plataforma.
- Facilidade de desenvolvimento, uma vez que não precisa seguir o modelo de programação de uma plataforma específica.
- São facilmente encontrados, pois seus conteúdos estão disponíveis na web, podem ser descobertos através de buscas na Internet.
- Fácil manutenção: As atualizações são feitas do mesmo modo como nos Websites.
- Não é necessário o usuário realizar o download de uma nova versão.

2.2.3 Aplicativos Híbridos

Ainda segundo Budiu (2013) outra opção são os aplicativos híbridos, estes são compostos por 2 partes, uma com elementos nativos e outra com elementos web. A parte nativa é codificada e estruturada seguindo os padrões da plataforma, ela será responsável por garantir todo o acesso aos recursos e funcionalidades do sistema operacional, esta parte representa uma pequena fração de todo o aplicativo. Já a parte web, o corpo principal, é desenvolvida usando HTML5 e seu conteúdo é carregado usando o browser dos navegadores.

Os Aplicativos Híbridos assim como os Nativos devem ser adquiridos através das lojas de aplicativos e ficam armazenados e instalados no próprio dispositivo.

Vantagens:

[5]HTML5 É a quinta versão da linguagem HTML, é uma linguagem para estruturação e apresentação de conteúdo para a Web.

- Custos reduzidos e Facilidade de desenvolvimento, pois seu código principal é desenvolvido em HTML5.
- Com a evolução do HTML5 espera-se que menos código de plataforma nativa sejam necessários para os aplicativos.
- A manutenção é feita assim como nos Aplicativos Web.

Capítulo 3

Plataforma Android

O Android foi um projeto iniciado no ano de 2003 pela empresa Android Inc[1]. Em 2005 a Android Inc. foi adquirido pela Google, e em 2007 dando continuidade ao projeto, o Google realizou uma parceria com mais de 40 empresas do setor de telefonia móvel, essa parceria ficou conhecida como OHA[2] (Opem Handset Alliance).

A OHA tinha como intuito a criação de um sistema operacional de código aberto, onde qualquer pessoa pode ter acesso ao código fonte. O software instalado no dispositivo teria os mesmos privilégios que os softwares criados ou licenciados pela fabricante. Dessa forma toda a comunidade de desenvolvedores interessada, poderia criar e customizar seus softwares, pois isso não ficaria restrito apenas as empresas membros da OHA (PEREIRA E SILVA, 2009).

O Android é uma plataforma de grande destaque no cenário mobile, além de possuir uma rica API (Application Programming Interface). Uma API é um conjunto de rotinas e padrões de programação que definem com o desenvolvedor pode realizar uma tarefa através de um biblioteca. A API disponibiliza fácil acesso a vários recursos de hardwares, como rede sem fio (wireless) e GPS. Os aplicativos Androids são desenvolvidos utilizando a linguagem de programação java, uma linguagem bastante difundida e de vasta documentação. A plataforma Android é composta por um sistema operacional, middlewares[3], e um conjunto de aplicativos principais como, os contatos, navegador de

[1]Empresa desenvolvedora de softwares para celulares, sediada em Palo Alto, na Califórnia (EUA)

[2]Open Handset Alliance (OHA) é uma aliança de diversas empresas com a intenção de criar padrões abertos para telefonia móvel

[3]Middleware é um código de software que atua como um mediador entre dois programas existentes e independentes.

Internet, entre outros (MONTEIRO, 2002).

O Android é distribuído sob licença Apache 2.0. Os desenvolvedores possuem acesso ao código-fonte do sistema, podendo então contribuir desenvolvendo e propondo melhorias ao projeto.

3.1 Kernel do sistema

O kernel, também conhecido como núcleo, é uma parte fundamental do sistema operacional, ele é responsável por gerenciar os recursos do hardware usados pelas aplicações, controlar os dispositivos e periféricos do sistema. É função do kernel, também, implementar as principais abstrações utilizadas pelos aplicativos (MAZIERO, 2014).

O kernel do Android foi desenvolvido baseado no sistema operacional Linux, apesar disso não se pode dizer que o Android é um Linux, pois ele não possui alguns padrões que as distribuições Linux apresentam (PEREIRA E SILVA, 2009).

Por ser desenvolvido utilizando um kernel Linux, o sistema operacional Android apresenta um alto grau de segurança. Toda vez que um aplicativo for instalado é criado um novo usuário Linux e diretórios que serão usados somente para aquele software. Cada aplicativo fica isolado dos demais e qualquer tentativa de acesso á informação de outro aplicativo é controlado por um mecanismo de permissão, onde é necessária a autorização do usuário para prosseguir a ação (PEREIRA E SILVA, 2009).

3.2 Arquitetura do Sistema Android

A arquitetura do Sistema Android é um conjunto de programas divididos em 4 camadas: aplicação, framework, bibliotecas e o kernel do sistema. Essa divisão esta exemplificada na Figura 3.1, a seguir falaremos sobre cada uma dessas camadas separadamente.

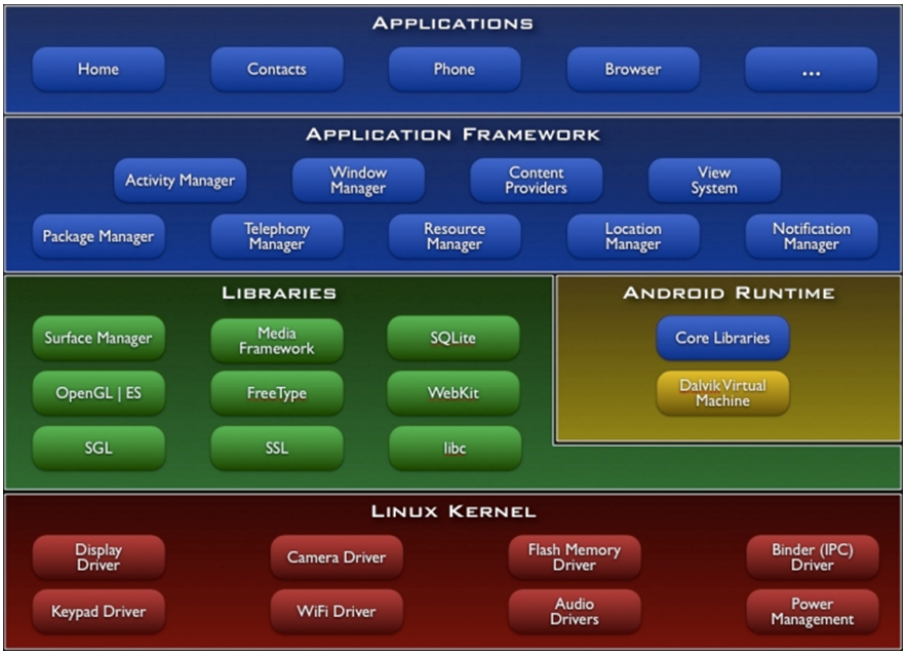

Figura 3.1: Arquitetura do sistema Android

3.2.1 Aplicações

Aplicação é a camada mais alta do sistema, é nela que se localiza todos os aplicativos do sistema, como: agenda de contatos, correio eletrônico, navegador de internet, calculadora, entre outras. As aplicações do Android são escritas em Java e executadas na Dalvik Virtual Machine, a máquina virtual do Android, responsável pelo gerenciamento do dispositivo.

O código e arquivos das aplicações do Android são compiladas utilizando as ferramentas do SDK, gerando assim um pacote Android com sufixo ".APK". O arquivo APK contém todo o conteúdo de uma aplicação, ele é o arquivo que os dispositivos com tecnologia Android usam para instalar o app (Android Developers, 2015).

3.2.2 Framework

Logo abaixo da camada de aplicação tem-se a camada do framework, ela tem a função de fornecer todos os serviços comuns para a criação das aplicações. O framework também realiza a interface com as camadas mais inferiores, facilitando o uso das bibliotecas no desenvolvimento dos aplicativos.

3.2.3 Bibliotecas

Temos também uma camada que é composta pelas bibliotecas e pelo Android Runtime. As bibliotecas são um conjunto de funções já escritas e prontas para serem usadas pelos desenvolvedores.

As bibliotecas do Android são desenvolvidas na linguagem C/C++, elas fornecem um conjunto de códigos que podem ser compatíveis com versões anteriores de APIs do Android ou podem ser compatíveis com apenas recurso de uma API especifica. A vantagem disso é que o aplicativo podem usar recursos de bibliotecas mais recentes e ainda serem compatíveis com versões anteriores do Android (Android Developers, 2015).

Algumas categorias de bibliotecas são: biblioteca de funções, biblioteca de servidores nativos, biblioteca de abstração de hardware, Bionic Libc, biblioteca com características especificas para o Kernel Linux. Elas fornecem recursos como gerenciamento de banco de dados SQLite, funções para navegação web, recursos para gráficos 2D (SGL) e 3D (OpenGL), suporte a formatos de áudio, imagem e vídeo, entre outros.

O Android Runtime é um conjunto de bibliotecas nativas do Java, chamadas de Core Libraries (bibliotecas de núcleo), e é também onde se encontra a Máquina Virtual, Dalvik Virtual Machine, onde os aplicativos são executados (Android Developers, 2015).

3.2.4 Kernel Linux

Na base de toda a arquitetura do sistema operacional Android encontra-se o Kernel do sistema. O Android utiliza a versão 2.6 do kernel Linux, este é responsável pelo gerenciamento de grande parte do sistema, como: gerenciamento de memória, segurança, consumo de energia, controle de acesso ao sistema de arquivo, comunicação entre processos e drivers, além de realizar a interface com o hardware do sistema.

3.3 Conceitos Básicos do Android

Para se desenvolver aplicações para o Android é necessário conhecer alguns conceitos básicos dessa plataforma. Nesta seção discutiremos sobre alguns desses importantes conceitos.

3.3.1 Activity

A Activity é uma das classes mais importantes do Android, representa a tela da Aplicação, sendo responsável por gerenciar a interface com o usuário. Ela controla os estados e a passagem de parâmetros entre as telas e define os métodos que são chamados quando o usuário executa alguma ação. Sempre que uma aplicação Android é executada a sua Activity principal é iniciada.

Ciclo de Vida de Uma Activity

O ciclo de vida de uma Activity são os estados prováveis em que ela pode se encontrar. A Figura 3.2 representa o ciclo de vida de uma Activity.

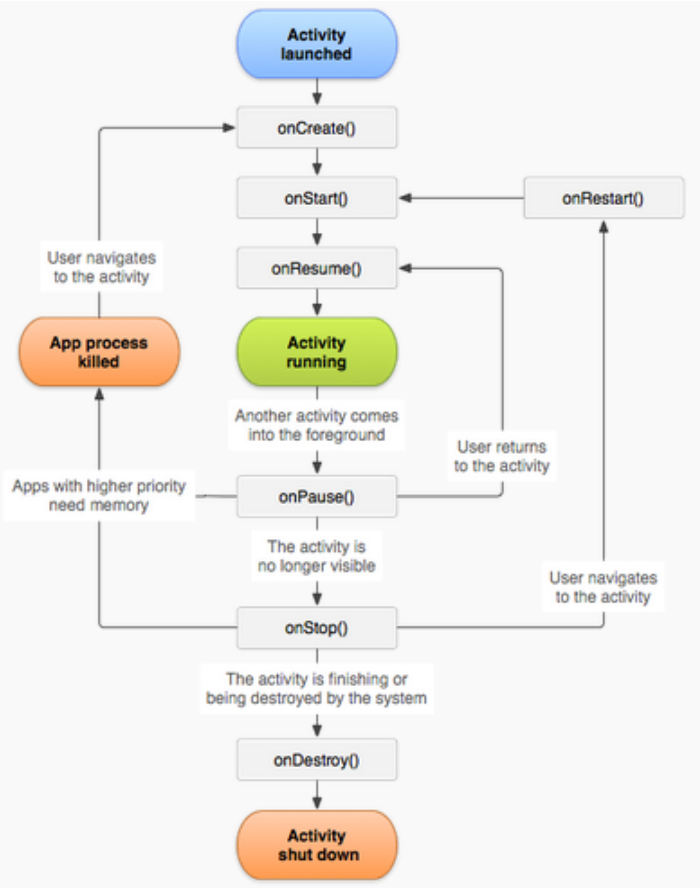

Figura 3.2: Ciclo de Vida de Uma Activity

Cada ciclo se inicia quando ocorre a chamada de um dos métodos que controlam o ciclo de vida de uma Activity e termina quando outros métodos são chamados. Estes métodos são: onCreate( ) é o primeiro método a ser chamado, encarregado de criar os layouts em xml através da view, ele é executado uma única vez; onStart( ) chamado quando a

Activity se torna visível para o usuário, executado logo após o onCreate( ) ou onRestart( ) quando a Activity volta a ter foco; onRestart( ) é chamado quando uma Activity pausada temporariamente volta a ter foco; onResume é chamado quando a Activity esta pronta pra interagir com o usuário, representa o estado de executando é sempre iniciado depois do método onStart( ); onPause( ) este método é chamado para interromper temporariamente o funcionamento de Activity que esta em execução, este método salva o estado da aplicação para que posteriormente ela possa executada novamente de onde parou; onStop( ) este método é chamado quando a Activity está sendo encerrada e não esta mais visível para usuário; onDestroy( ) é chamado para encerrar a execução de uma Activity, ao executar este método a Activity é removida da pilha e seus processos são encerrados, este método pode ser chamado pelo sistema operacional ou pela própria aplicação.

3.3.2 View

A classe View é encarregada de montar os elementos visuais das telas, como: Botão, Checkbox, Imagens, Textos entre outros. É necessário um gerenciador de layout para organizar esses elementos na tela, ou seja, determinar suas posições.

Alguns gerenciadores de layout são: AbsoluteLayout, FrameLayout, TableLayout, RelativeLayout e LinearLayout. A classe-mãe de todos os gerenciadores é a ViewGroup. Observe na Figura 3.3 a Hierarquia da Classe View.

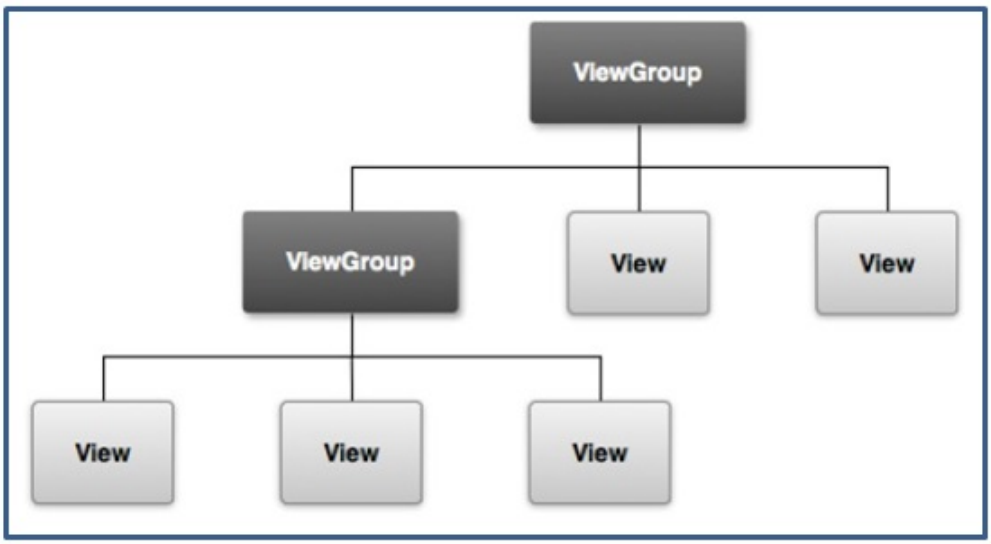

Figura 3.3: Hierarquia da Classe View

3.3.3 Método setContentView(view)

O método setContentView(view) é que estabelece a união entre a Activity e a View, este método recebe como parâmetro a view que será exibida na tela. Ou seja, sempre que se deseja reproduzir uma view na tela é necessário chamar este método dentro da Activity.

3.3.4 AndroidManifest

O AndroidManifest.xml é o arquivo principal da aplicação, é nele que estão contidas todas as configurações que foram definidas pelo desenvolvedor, como nome do pacote, versão da aplicação, versão mínima da API exigida, nome das classes de cada Activity, ícone e nome da Aplicação, entre outras. Todas essas configurações são necessárias para execução da Aplicação. Na Figura 3.4 podemos observa as configurações contidas no arquivo AndroidManifest.xml do Aplicativo Chamadas.

Chamadas Manifest

```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.chamadas"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="16" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/app_name"
        android:theme="@style/AppTheme" >
        <activity
            android:name="br.com.chamadas.ActivityChamadas"
            android:label="@string/app_name" >
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
```

Figura 3.4: Arquivo Manifest do Aplicativo Chamadas

O arquivo AndroidManifest.xml é encontrado no diretório raiz do projeto.

3.4 Versões do Sistema e Suas Funcionalidades

O sistema Android possui uma tradição de que todas as versões de seu sistema operacional recebam nomes de doces ou sobremesas.

No ano de 2009 foi lançada a primeira versão comercializada do sistema, versão

1.5, que recebeu o nome de CupCake, essa versão foi baseada do kernel Linux 2.6.27. Dentre as funcionalidades do Android 1.5 temos: suporte a teclado QWERTY, suporte à Bluetooth A2DP que pode ser utilizado em fones de ouvido estéreo e rádios de automóveis, ferramenta de gravação de vídeos com envio para o YouTube, Widgets, entre outras. Veja na Figura 3.5 o logotipo do Android Cupcake.

Figura 3.5: Logotipo Android Cupcake

Em Setembro de 2009 foi o lançamento do Android 1.6, chamado de Donut. Essa versão do Android veio com algumas modificações. Melhoria na Câmera, com integração entre o modo foto, vídeo e galeria, aplicativo Android Market aprimorado, tornando a navegação mais fácil, indicador de uso de bateria, aplicativos de e-mail, mensagens SMS, Lista de Contatos, opção favoritos no navegador, dentre outros aplicativos. Na Figura 3.6 pode-se observar o logotipo do Android Donut.

Figura 3.6: Logotipo do Android Donut

Em outubro de 2009 foi a vez da versão 2.1 do Android, esta recebeu o nome de

Eclair. Dentre as funcionalidades que esta versão trouxe: Suporte a novos hardwares, permitindo o desenvolvimento de smartphones com mais capacidade de memória, resoluções de tela e processadores mais rápidos, suporte a nove telas principais e papeis de parede animado, gerenciador de bateria, suporte a Bluetooth 2.1, melhorias no navegador Web, zoom digital e suporte à HTML 5. Na Figura 3.7 tem-se o logotipo desse versão.

Figura 3.7: Logotipo Android Eclair

Em Maio de 2010 foi o lançamento do Android 2.2 que recebeu o nome de Froyo. Essa versão apresentou mudanças importantes para a plataforma. Uma das grandes diferenças dessa versão em relação às outras foi o suporte para Flash e a inclusão do Just-In-Time Compilation (JIT) que trouxe um melhor desempenho para os dispositivos, permitindo maior velocidade no processamento das tarefas.

Além disso, foi otimizado o consumo de bateria e adicionado a funcionalidades como a de instalar os aplicativos diretamente no SD Card (cartão me memoria), compartilhar internet através do cabo USB ou WI-FI (Tethering), atualização automática de aplicativos, sincronização PC/Android, pesquisa no sistema operacional e gerenciador de tarefas. Veja na figura 3.8 o logotipo da versão 2.2.

Figura 3.8: Logotipo Android Froyo

A versão 2.3, Gingerbread, foi lançada em Novembro de 2010, algumas das funcionalidades que essa versão trouxe foram: melhoria na interface do usuário, teclado melhorado e mais rápido, facilidade para acessar o gerenciador de tarefas, melhoria no consumo de energia da bateria, suporte ao protocolo SIP para chamadas via Internet e melhoria no gerenciador de download. Veja na 3.9 o logotipo da versão 2.3.

Figura 3.9: Logotipo do Android Gingerbread

O Honeycomb, versão 3.0 do Android, foi lançado em janeiro de 2011, essa versão vem com um diferencial, pois nesta época além dos smartphones tinha-se outro segmento de dispositivos móvel no mercado, os tablets. As funcionalidades dessa versão incluem: layout totalmente adaptado para tablets, suporte a navegação 3D, melhorias nas multitarefas, Google Maps com visualização 3D e melhorias na navegação web (incluindo navegação privada). Na Figura 3.10 tem-se o logotipo da versão 3.0 do Android.

Figura 3.10: Logotipo do Android Honeycomb

Em Outubro de 2011 foi o lançamento da versão 4.0, com o nome de Ice Cream Sandwich. As principais funções que essa versão apresentou foram: unificação da plataforma pra tablets e smartphones, controle da utilização da banda de dados, interface mais flexível e rica em recursos, entrada de texto e correção ortográfica aprimoradas, mecanismo de entrada de voz, ferramentas de acessibilidade, captura de telas (screenshots) e maior conectividade com a nuvem. Pode-se ver na Figura 3.11 o logotipo do Android Ice Cream Sandwich.

Figura 3.11: Logotipo Android Ice Cream Sandwich

Em Julho de 2012 chega ao mercado o Jelly Bean, versão 4.1/4.2 do Android, trazendo recursos como: Google maps off-line, reconhecimento e digitação por voz off-line, Google Now (assistente pessoal virtual), melhoria na central de notificações, interface mais intuitiva e melhor organizada, projeto Butter que foca na melhoria do processamento do dispositivo. Na Figura 3.12 pode-se ver o logotipo do Andorid Jelly Bean.

Figura 3.12: Logotipo Do Android Jelly Bean

Em Outubro de 2013 foi lançado a versão 4.4, Android KitKat,. Como esta versão é uma continuação do Jelly Bean, não apresenta grande diferença, as mudanças mais significativas realizadas foram na parte visual do sistema. Tiveram melhorias também no gerenciamento de memória, na resposta do toque em tela (touchscreen) e na identificação de chamadas e contatos, além de suporte a impressão e infravermelho. Na Figura 3.13 tem-se o logotipo do Android KitKat.

Figura 3.13: Logotipo do Android KitKat

O Lollipop é a versão mais recente do sistema operacional, foi apresentada ao publico em Outubro de 2014. Essa versão vem como uma serie de mudanças tanto no visual como no desempenho do sistema. Dentre as funcionalidades temos: adição de múltiplos perfis, permitindo que os usuários tenham mais de um perfil em penas um dispositivo, mudança na interface do Gmail, novo discador, melhorias nos recursos de

multimídia, além disso, o Lollipop reforçou as ferramentas que garantem a segurança do sistema. Na Figura 3.14 pode-se ver o logotipo do Android Lollipop.

Figura 3.14: Logotipo do Android Lollipop

Capítulo 4

Linguagens Utilizadas

Neste capítulo será discutido sobre as linguagens utilizadas no desenvolvimento do sistema. Destacando-se conceitos, características e estrutura dessas linguagens.

4.1 Programação Orientada a Objetos

Com o surgimento das linguagens de programação orientada a objetos houve uma mudança no modo de se desenvolver softwares. De acordo com Mendes (2009, p. 22) a programação orientação a objetos (POO) apresenta uma abordagem diferente da programação estruturada, a POO procura adotar formas mais próximas da realidade para tratar a complexidade dos softwares.

No modelo de programação estruturada, o desenvolvimento se dava utilizando três estruturas básicas: sequência, decisão e controle. As operações eram realizadas por um grupo de funções ou procedimentos, que eram chamados por um método principal. Neste tipo de programação tinha-se muita redundância e pouca reutilização de códigos.

Já a programação orientada a objeto apresenta um estilo de programação estruturada em camadas, criada com o intuito de aproximar o mundo real do mundo virtual. Segundo Resende e Silva (2005, p. 11), com advento da programação orientada a objetos houve um salto na organização dos sistemas. O mundo real passou a ser representado e modelado na forma de classes e objetos, e como tal, as classes de objetos puderam ser reutilizadas com maior facilidade.

Leite e Rahal (2002) fazem uma alusão a teias para explicar melhor o modelo de programação orientada a objetos, eles afirmam que:

> O paradigma da programação orientada ao objeto possibilita a criação de sistemas que parecem teias, onde cada nó dessa teia é uma classe, mas cada classe pode ser aproveitada para a composição de outros nós da teia, bem como a reutilização desses nós em outros sistemas, diminuindo drasticamente o tempo de codificação e depuração de códigos.

Segundo Mendes (2009, p. 16) o paradigma orientação a objeto, introduzido na década 70, ganhou maior credibilidade depois do surgimento da Linguagem Java. No item 7.2 falaremos mais detalhadamente sobre essa linguagem.

A programação orientada a objetos apresenta alguns conceitos que são importantes para o seu entendimento, discutiremos sobres eles a seguir.

4.1.1 Classe

De acordo com Mendes (2009, p. 29) "uma classe define o modelo de um objeto, ou seja, todas as características que o objeto contém foram definidas pela classe". Uma vez que a classe é criada pode-se criar quantos objetos dessa classe forem necessários e manipula-los para a solução do problema.

Cada classe possui atributos, que são os elementos da classe, ou seja, suas variáveis. E também possuem os métodos, que são funções para se realizar as atividades. Na Figura 4.1 podemos ver o exemplo de uma classe pessoa, bem como os seus atributos e métodos.

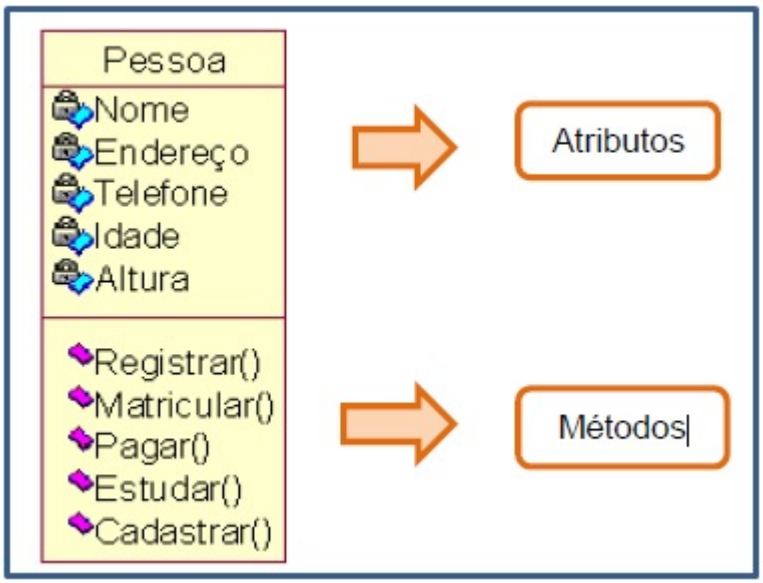

Figura 4.1: Classe Pessoa

4.1.2 Objeto

Um Objeto pode ser uma varável, uma função ou uma estrutura de dados. Cada objeto pertence a uma classe e esta define o seu comportamento através dos atributos e métodos.

O uso dos objetos é feito através do envio de mensagens, ou seja, a mensagem é o meio de comunicação entre os objetos. “Para isso é preciso criar um objeto, identificar o nome do método a ser executado e, caso necessário, identificar também parâmetros que o método recebe ou retorna” (MENDES, 2009 p. 32).

Os objetos utilizam os métodos através de suas interfaces, são elas que definem quais métodos podem ser chamados. Além disso as interfaces são utilizadas para esconder detalhes da implementação daqueles que estão fazendo o uso do objeto.

4.1.3 Herança

De acordo com Deitel (2010, p.15) “a orientação a objetos também faz o uso dos relacionamentos de herança, dos quais as classes de objetos novas são derivadas absorvendo características de classes já existentes e adicionando-se características únicas dessas mesmas classes”.

A estratégia de herança nos permite fazer a reutilização de códigos. Classes filhas herdam propriedades mais genéricas das classes pais. Com isso classes filhas podem ser programadas incluindo somente as diferenças que possuem da classe pai. Na Figura 4.2 temos um exemplo de herança, onde “Cavalo” sendo um “Animal” herda todos os atributos e métodos genéricos da classe animal.

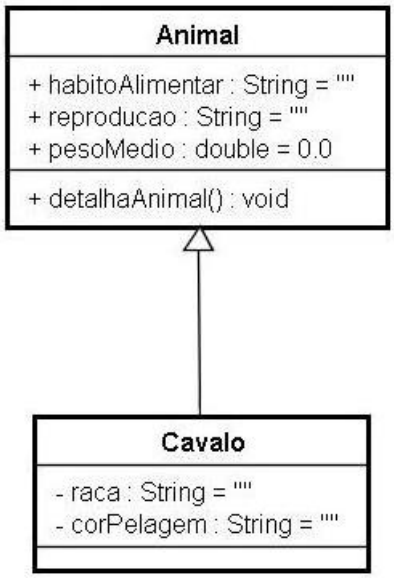

Figura 4.2: Exemplo de Herança

4.1.4 Encapsulamento

Na programação orientada a objeto o encapsulamento é utilizado para controlar acesso a atributos e métodos de uma classe, ou seja, determinar qual parte de uma classe pode ser acessada ou não por um cliente.

Este controle é feito definindo os membros da classe como sendo públicos ou privados. Ao declarar um membro como público isso garantirá um maior acesso a estes, podendo ser utilizados sem estrição. Já os membros privados só podem ser acessados diretamente por instâncias das classes as quais pertencem.

> O encapsulamento é um dos pilares da orientação a objetos sua característica é ocultar partes da implementação, desta forma construir softwares que atinjam suas funcionalidades e escondam os detalhes de implementação do mundo exterior. Os objetos encapsulados funcionam como uma caixa preta, sabe-se da sua interface externa, mas não precisa se preocupar com o que acontece dentro dela. (SINTES, 2002, p.22)

O que Sintes (2002) afirma pode ser ilustrado na Figura 4.3.

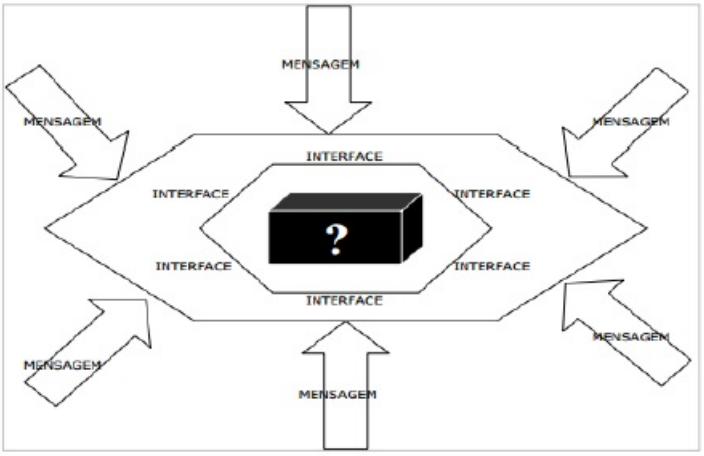

Figura 4.3: Encapsulamento

4.1.5 Polimorfismo

Polimorfismo significa possuir várias formas, essa propriedade permite usarmos o mesmo nome para identificar métodos diferentes. Com a utilização do polimorfismo duas ou mais classes podem derivar de uma mesma superclasse e invocarem métodos que possuem a mesma identificação mas comportamentos diferentes.

Segundo Deitel (2010, p. 305) o polimorfismo permite escrever programas que processam objetos que compartilham a mesma superclasse como se todos fossem objetos da superclasse, o que pode simplificar a programação. Com isso podemos projetar e implementar sistemas que são facilmente extensíveis, novas classes podem ser adicionadas com pouca ou nenhuma modificação a partes gerais do programa.

4.2 Java

A Linguagem Java foi desenvolvida pela Sun Microsystems e apresentada ao público em 1995. Ela é originada da linguagem C/C ++, que são duas linguagens muito utilizadas em todo mundo. Java é uma linguagem portável, adequada para a implementação de aplicativos baseados na Internet e na World Wide Web. Ela oferece vários recursos importantes como, imagens gráficas, componentes de interface gráfica com o usuário, tratamento de exceções, multithreading, multimídia (áudio, imagem, animação e vídeo), processamento de arquivos, computação distribuída, processamento de banco de dados, entre outras (DEITEL, 2003, p. 11).

Ela é uma linguagem de programação de alto nível, ou seja, com alto grau de abstração, dispensando conhecer características da arquitetura do computador para realizar a codificação. Ela segue o paradigma de programação orientada a objetos já discutido acima.

O Java pode ser utilizado para o desenvolvimento de variados tipos de softwares, atualmente ele esta presente em aplicativos corporativos de grande porte, em sistemas voltados para Web, em aplicativos para dispositivos móveis como Smartphones, Tablets e PDAs , em servidores Web entre outros.

A linguagem Java oferece uma vasta coleção de APIs (Aplication Programming Interfaces). Elas são bibliotecas de classes Java disponibilizadas pela Sun para que o programador possa utilizar algumas funcionalidades padrões já implementadas.

Além de todas essas características que o Java nos trás, ela é linguagem com um volume de documentação bem extenso que pode ser facilmente acessada pela Internet.

4.2.1 Java Virtual Machine " JVM (Máquina Virtual Java)"

A portabilidade da linguagem Java é uma de suas grandes vantagens. Por ser uma linguagem de programação multiplataforma, os programas desenvolvidos em Java podem ser executados em qualquer sistema operacional desde que possua a Máquina Virtual Java instalado.

> A Máquina Virtual é um aplicativo de software que simula um computador, mas oculta o sistema operacional e o hardware subjacentes dos programas que interagem com ela. Se a mesma JVM for implantada nas várias plataformas de computador, os aplicativos que ela executa podem ser utilizados em todas as plataformas (DEITEL, 2010 p. 10).

Na Figura 4.4 está descrito o funcionamento da JVM. Incialmente nos tem-se o código fonte em Java, "Programa.java", este código é compilado e então é gerado um arquivo com a extensão ".class". Este arquivo contém os bytecodes que serão lidos pela Máquina Virtual. Os bytecodes resultantes da compilação serão então executados pela JVM e traduzidos para a linguagem de máquina. Esta etapa de tradução se caracteriza como uma das importantes de um programa Java.

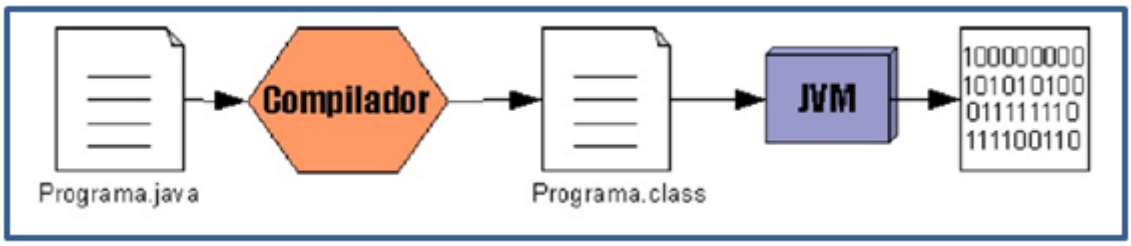

Figura 4.4: Funcionamento da Máquina Virtual Java

4.2.2 Aplicativo Java

De acordo com Deitel (2010, p.29) "um aplicativo Java é um programa de computador que é executado quando você utiliza o comando java para carregar a Java Virtual Machine".

Na Figura 4.5 está exemplificados as fases de desenvolvimento de um programa em Java.

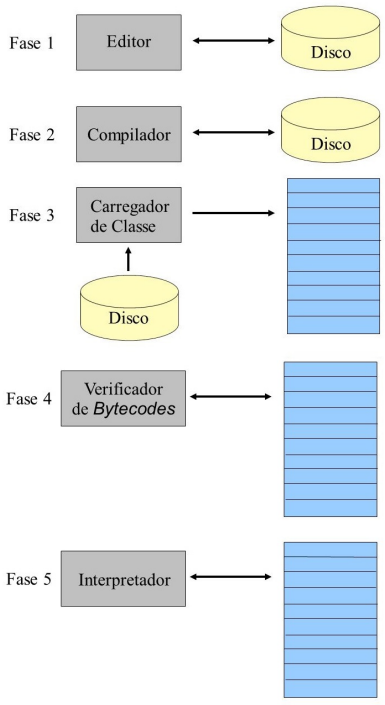

Figura 4.5: Fases de um Aplicativo Java

- Fase 1: O programa é criado em um editor e armazenado em disco.
- Fase 2: O Compilador cria os bytecodes e os armazena em disco com extensão .class.

- Fase 3: O Carregador de Classe lê os arquivos .class que contem os bytecodes e os coloca na memória.
- Fase 4: O verificador confirma se todos os bytecodes são válidos e não violam restrições e segurança do Java.
- Fase 5: Para o Aplicativo a JVM executa os bytecodes que estão na memoria e os traduz para a linguagem de máquina.

4.3 Linguagem de Marcação

As linguagens de marcações são utilizadas para organizar e definir como os conteúdos serão exibidos na tela. De acordo com Fugeri (2006), as linguagens de marcação permitem criar documentos com uma estrutura de representação que seja compreendida por diversos sistemas de software, independentemente da máquina onde estão sendo executados.

Uma Linguagem de marcação é um conjunto de convenções utilizadas para a codificação de textos. Ela deve especificar que marcas são permitidas, quais são exigidas, como se deve fazer distinção entre as marcas e o texto e qual o significado da marcação. (ALMEIDA, 2002)

Uma estrutura comum de marcação presente nessas linguagens são as tags. As tags são utilizadas para passar uma instrução e determina o inicio e fim de um texto. Um exemplo simples disso é quando se usa o marcador, <TITLE>, para referenciar ao título de um documento, assim todo o texto delimitado por ele corresponderá a esse título.

Existem hoje, muitas linguagem de marcações disponíveis para desenvolvimento de sistemas, falaremos a seguir, mais detalhadamente, sobre a linguagem XML.

4.3.1 XML

A Linguagem XML, Extended Markup Language, é derivada da linguagem SGML, Standard Generalized Markup Language, que no português significa, Linguagem Padronizada de Marcação Genérica. A SGML foi criada no início dos anos 70 e padronizada pela ISO (Organização Internacional para Padronização), ela surgiu da necessidade em que a empresa IBM tinha para gerenciar e armazenar grande quantidade de documentos.

O XML é uma linguagem de marcação que possui um padrão que facilita o compartilhamento de informação na Internet e entre aplicações. Uma das principais características dessa linguagem é o fato de ser extensível, ou seja, ela permite que novas tags de marcações sejam definidas pelos próprios desenvolvedores. Almeida (2002), afirma que a definição de tags próprias confere à linguagem XML habilidades semânticas que possibilitam melhorias significativas em processos de recuperação e disseminação da informação.

O XML organiza os conteúdos como: textos, botões, campos de textos e demais componentes, de maneira hierárquica nas telas. Por ser uma linguagem criada inicialmente para atuar na Internet ela é independente de plataforma.

Observe na Figura 4.6 alguns elementos da linguagem XML, como: as tags, marcadores indicando o início e final do texto. Pode-se notar também que as marcas não seguem um padrão fixo, elas são criadas pelos desenvolvedores e definidas com termos que melhor descrevem o conteúdo.

```
<?xml version="1.0" encoding="iso-8859-1" ?>
- <noticias>
  - <noticia>
      <titulo>NOTICIA 1</titulo>
      <autor>Autor 1</autor>
      <data>24/01/2007</data>
      <texto>Conteudo 1</texto>
    </noticia>
  - <noticia>
      <titulo>NOTICIA 2</titulo>
      <autor>Autor 2</autor>
      <data>23/01/2007</data>
      <texto>Conteudo 2</texto>
    </noticia>
  - <noticia>
      <titulo>NOTICIA 3</titulo>
      <autor>Autor 3</autor>
      <data>22/01/2007</data>
      <texto>Conteudo 3</texto>
    </noticia>
  </noticias>
```

Figura 4.6: Exemplo de Código na Linguagem XML

No Android a linguagem XML é utilizada para criação dos layouts das aplicações. A criação desses layouts usando o XML será explicada no capítulo 9.

Capítulo 5

Ferramentas Utilizadas No Desenvolvimento

Para o desenvolvimento de aplicações para o Android é necessário adquirir, instalar e configurar um conjunto de ferramentas. Estas ferramentas irão trabalhar simultaneamente dentro do espaço de desenvolvimento para a produção das aplicações. Neste trabalho foram utilizadas as seguintes ferramentas: O Android SDK (kit de Desenvolvimento de Software) que contem o emulador e outras ferramentas necessárias, o Eclipse que é o ambiente de desenvolvimento Java e o JDK (kit de desenvolvimento Java). A seguir será descrito sobre cada uma dessas ferramentas como também suas funções.

5.1 Android SDK

O Android SDK é um Kit de desenvolvimento para o sistema operacional Android, fornecido pela Google para que os desenvolvedores possam criar suas próprias aplicações e roda-las direto na plataforma, além de realizar os testes e depurar a aplicação.

Dentro do Android SDK tem-se as bibliotecas, um conjunto de ferramentas para desenvolvimento e depuração, como o Android SDK Tools, e uma API para a linguagem Java. Esta API (Application Programming Interface), Interface de Programação de Aplicações, será responsável por estabelecer a comunicação entre os códigos e definir o comportamento dos objetos na interface. Veja na Figura 5.1 o painel do Android SDK Manager e alguns de seus componentes.

O Android SDK é suportado nos seguintes sistemas operacionais:

- Windows
- Mac OS X10.5.8 ou posterior
- Linux

Ele pode ser encontrado para download no endereço: http://developer.android.com/sdk/.

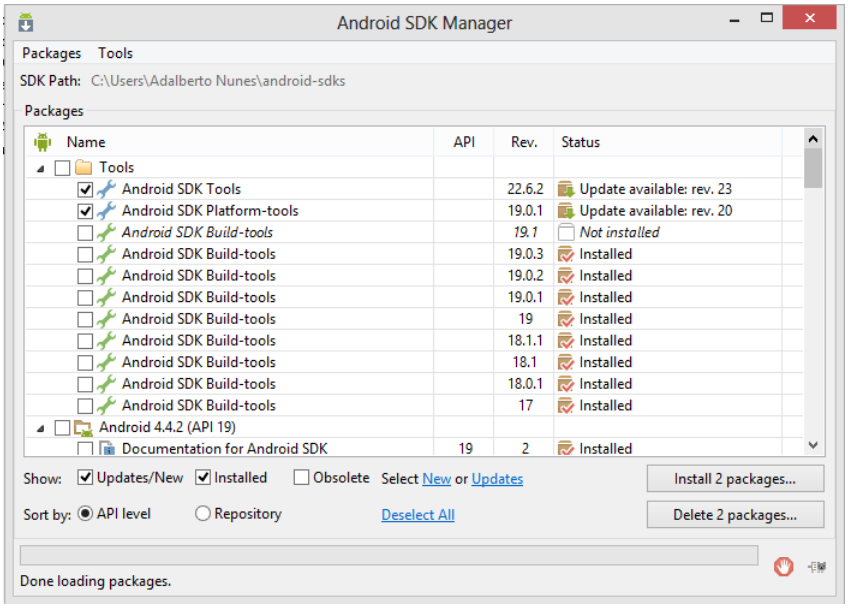

Figura 5.1: Adroid SDK Manager

O Android SDK possui também um emulador, AVD (Android Virtual Devices) que é um dispositivo virtual que irá rodar no computador simulando o celular, com este emulador é possível visualizar e efetuar testes na aplicação que esta sendo desenvolvida como se ela estivesse sendo executada em um dispositivo real. A Figura 5.2 exemplifica o Emulador do Android.

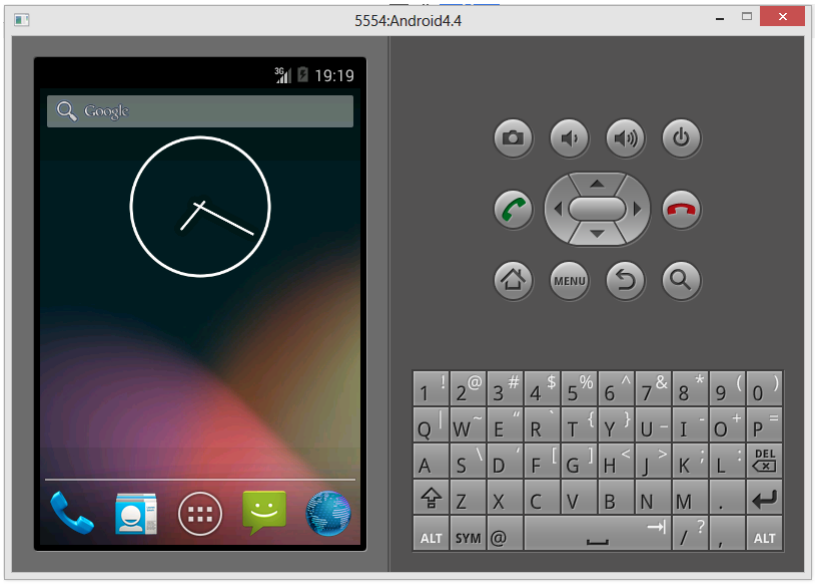

Figura 5.2: Emulador Android 4.4

5.2 Eclipse

IDE (Integrated Development Enviroment) é um ambiente integrado de desenvolvimento, ele engloba um conjunto de ferramentas que apoiam o desenvolvedor na codificação dos softwares, com o uso desses ambientes é possível facilitar e agilizar os processos de produção. Alguns dos componentes de um IDE são:

- Editor de programas: Que é onde de fato o desenvolvedor irá escrever os códigos. Estes editores apresentam vários recursos que auxiliam a codificação, como: auto-correção, sinalização de erros, organização do código, endentação entre outros.
- Compilador ou Interpretador: Que irá transformar todo o código fonte escrito para um código de máquina, para que o computador possa entender
- Linker: Liga os códigos com as bibliotecas utilizadas, transformando o programa em um executável.

Para o desenvolvimento da aplicação, o IDE utilizado foi o Eclipse, por ser um ambiente de desenvolvimento mais usado e preferido pelo Google, além do Eclipse pode-se utilizar outras IDEs como : Netbeans ou Intellij IDEA.

O Eclipse é uma plataforma de desenvolvimento de software livre extensível, baseada em Java. Por si só, é simplesmente uma estrutura e um conjunto de serviços para

desenvolvimento de aplicativos de componentes de plug-in. Felizmente, o Eclipse vem com um conjunto padrão de plug-ins, incluindo as amplamente conhecidas Ferramentas de Desenvolvimento Java (JDT)(Eclipse, IBM).

O Eclipse não se limita apenas a linguagem Java, pode-se desenvolver neste ambiente usando outras linguagens como C/C++, PHP e COBOL. Na Figura 5.3 podemos ver a interface do Eclipse. No canto esquerdo temos projetos que estão sendo desenvolvidos, no centro temos o Editor de Programas e na parte superior temos os menus e algumas ferramentas.

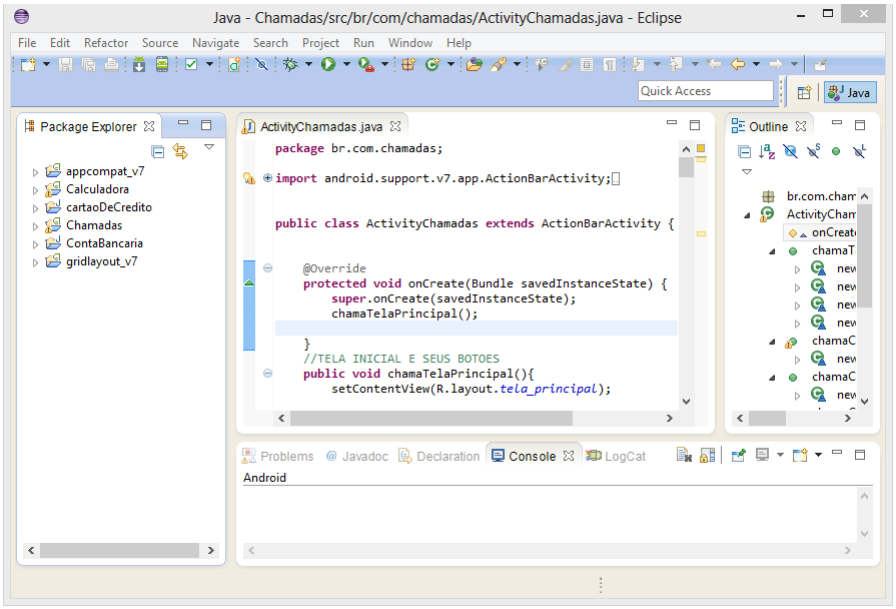

Figura 5.3: Ambiente Integrado de Desenvolvimento, Eclipse

Para desenvolvimento de Aplicações para Android usando o Eclipse é necessário a utilização de um plugin[1] chamado Android Development Tools (ADT), ele irá integrar o emulador ao Eclipse. Este plugin adicionará mais recursos ao Eclipse , dando mais agilidade e facilidade no desenvolvimento, auxiliando na criação da interface do usuário e na depuração dos aplicativos usando ferramentas do SDK.

5.3 JDK (Java Development Kit)

O JDK, Kit de Desenvolvimento Java é um conjunto de utilitários para o desenvolvimento e testes de softwares para a plataforma Java, este pacote é disponibilizado pela

[1]Plugin todo programa, ferramenta ou extensão que se encaixa a outro programa principal para adicionar mais funções e recursos a ele.

Oracle.

Neste Kit contém todas as ferramentas necessárias para a criação e execução de Aplicações Java, além da Máquina Virtual Java (JVM), que tem a função de carregar e executar os Aplicativos. Dentre as ferramentas que compõem este pacote temos: As Bibliotecas (API's), o Javac, que é o compilador Java utilizado para compilar as classes dos programas, o JavaDoc, que é o gerador de documentação, o Jar, programa de compactação, e o JavaDebugger, que é um depurador de linha de comando para classes Java.

5.4 Banco de Dados (SQLite)

A persistência de dados no Android é feita utilizando a biblioteca SQLite, essa biblioteca é desenvolvida na linguagem C e é multiplataforma, podem ser integrada a sistemas desenvolvidos em várias linguagens ,como: Java, php, C++, Delphi, entre outras. O SQLite é uma banco de dados Open Source (código aberto) que suporta a sintaxe SQL para a realização das operações (inserção, exclusão e atualização) como também para manipulação e realizações de consultas.

Diferentemente da maioria dos outros bancos de dados SQL, o SQLite não precisa acessar um SGBD (Sistema de Gerenciamento de Banco de Dados). Ele cria uma arquivo em disco, lê e escreve diretamente sobre este arquivo (SQLite, 2014).

O Android oferece todo o suporte para o SQLite. Para utilização do Banco de dados nos dispositivos é necessário fornecer apenas as instruções para criação e atualização do banco, e todo o gerenciamento é feito de maneira automática pela plataforma do Android. Cada aplicação pode criar um ou mais banco de dados, que ficam armazenados no próprio dispositivo, localizados no diretório relativo ao nome do pacote. Importante ressaltar que um banco de dados só é visível à aplicação que o criou. Para utilizar a biblioteca do SQLite no desenvolvimento android é necessário importa-la para o projeto, como mostra a Figura 5.4.

```
import java.util.ArrayList;
import java.util.List;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteException;
import android.database.sqlite.SQLiteOpenHelper;
import android.content.ContentValues;
import android.content.Context;
import android.database.Cursor;
import android.util.Log;
```

Figura 5.4: Importando a Biblioteca SQLite

O banco de dados no android pode ser criado de várias formas:

- Utilizando uma API do Android para o SQLite;
- Usando um cliente do SQLite como o SQLite Expert Personal[2], que é gratuito, ou;
- Utilizando o aplicativo Sqlite3[3] pelo console do emulador Mais detalhes sobre a implementação

[2]SQLite Expert Personal. Disponível em: < http://www.sqliteexpert.com/download.html > Acesso em: 24/11/2014.

[3]Sqlite3. Disponível em: < http://www.sqlite.org/sqlite.html > Acesso em: 24/11/2014

Capítulo 6

Sistema Desenvolvido

Neste capitulo será descrito sobre o sistema de gerenciamento de informações educacionais. Destacando-se a estrutura e organização do projeto da aplicação, requisitos e funcionalidades da aplicação, criação da interface do usuário e as etapas de construção do código.

6.1 Estrutura e Organização do Projeto da Aplicação

6.1.1 Aplicativo de Chamadas

O Aplicativo desenvolvido é um sistema para controle de frequência dos alunos. É um sistema para dispositivos móveis com a plataforma Android, para a instalação e execução do Aplicativo é necessário a versão 2.2 do sistema operacional Android ou superior.

Com este sistema o professor poderá registrar a frequência dos alunos em suas aulas durante o período letivo, como também acompanhar o saldo de faltas e presenças de cada aluno por disciplina, tudo isso de uma maneira mais rápida, eficiente e gastando menos recursos.

6.1.2 Requisitos do Sistema

O levantamento de requisitos é o inicio para toda atividade de desenvolvimento, seja de um software comercial, um sistema web ou um aplicativo. Nesta etapa foi feita uma análise do domínio da aplicação, para se compreender mais a cerca ambiente em que

o sistema será inserido. Foi identificado as partes interessadas para que o levantamento dos requisitos fossem concentrados nas necessidades dos usuários finais, que no caso deste projeto são os professores.

Os requisitos levantados para o Aplicativo de realização de Chamadas foram os seguintes:

- Realizar o cadastro das disciplinas que o professor leciona.
- Realizar o cadastro dos alunos que estão matriculados em tais disciplinas.
- Alterar dados cadastrados, como: Remover disciplina, remover ou adicionar aluno.
- Efetuar a chamada pelas datas em que as aulas são ministradas.
- Consultar o quadro de frequência dos alunos por disciplina.

6.1.3 Protótipo

O protótipo é uma fase importante do desenvolvimento, é nesta fase que se discute detalhes sobre a interface e funcionalidades do sistema. Com a apresentação do protótipo é possível reduzir consideravelmente as chances de fazer alteração e correções nos códigos quando o projeto estiver em um nível mais avançado.

Levantado todos os requisitos do Aplicativo, foi desenvolvido um protótipo contendo apenas recursos gráficos para simular os aspectos e funcionalidades do sistema.

6.1.4 Interface do Usuário

A interface do usuário é uma parte fundamental da aplicação, é através dela que o usuário irá interagir com o sistema para executar suas atividades. A interface precisa ser construída focando em seus usuários finais, para que estes não tenham dificuldades ao usar a aplicação. É preciso que a interface tenha um layout claro e bem organizado, com componentes intuitivos, textos objetivos e diretos que não provocam dúvidas àqueles que estão utilizando.

As telas da aplicação foram desenvolvidas usando a linguagem xml, mas também podem ser desenvolvidas usando apenas o Java. No desenvolvimento desse aplicativo optou-se por usar o xml por se tratar de uma forma mais fácil de gerenciar a Interface

gráfica e parte lógica da aplicação e também pelo fato de termos uma maior quantidade de documentação de desenvolvimento em projetos Android usando a junção dessas duas linguagens, o Java para parte lógica e o xml para interface gráfica.

Para a construção da parte gráfica da aplicação é necessário criarmos arquivos xml e colocá-los dentro da pasta layout do projeto.

6.1.5 Telas do Aplicativo Chamadas

Tela Inicial

A tela inicial do aplicativo apresenta para o usuário um menu contendo cinco opções, ao escolher uma dessas opções ele será direcionado para uma nova tela da aplicação. As opções são as seguintes:

- Cadastro: Clicando neste botão o usuário será direcionada a uma nova tela da aplicação composta por 2 opções, uma para realizar o cadastro das disciplinas e outra para realizar o cadastro dos alunos na disciplina.

- Alterar Dados: Clicando neste botão o usuário será direcionado a uma tela da aplicação onde será possível realizar a alteração dos dados que foram cadastrados. Nesta opção pode ser feito a atualização de dados como também a exclusão de alunos ou disciplinas.

- Chamada: Escolhendo esta opção o usuário será direcionando a tela de realização da chamada. Nesta tela ele selecionará uma data e uma disciplina para a efetuação da chamada.

- Frequência: Escolhendo esta opção o usuário será direcionado a tela a tela de consulta, onde ele poderá verificar o saldo de faltas e presenças dos alunos.

- Sair: Clicando no botão sair o usuário irá encerrar o Aplicativo. Aplicação chama o método finish( ) que encerra a execução da Activity..

Essa interface da aplicação é construída utilizando o RelativeLayout, gerenciador de layouts que permite posicionar componentes, abaixo, acima ou ao lado de outro componente. Podemos visualizar a tela inicial da aplicação na Figura 6.1.

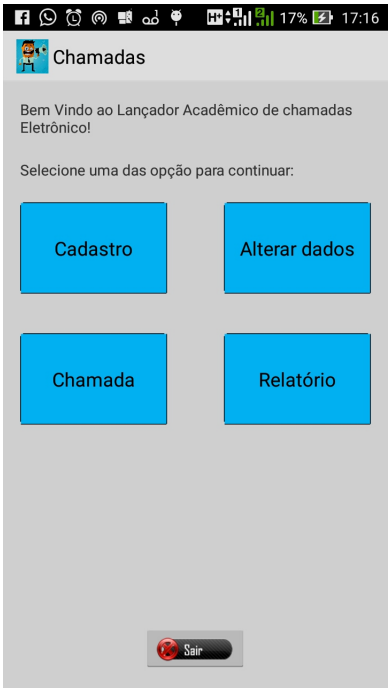

Figura 6.1: Tela Inicial da Aplicação

Na figura 6.2 pode-se observar parte do código para a criação da tela principal do sistema. Neste código tem-se o método onCreat, que é o primeiro método a ser chamado, responsável por criar o layout. Logo em seguida são declarados os botões que compõe a tela e que dão acesso as funções do sistema.

```
public class ActivityChamadas extends ActionBarActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        chamaTelaPrincipal();
    }

    // botões da tela principal
    public void chamaTelaPrincipal(){
        setContentView(R.layout.tela_principal);

        Button botao_chamada = (Button) findViewById(R.id.botao_chamada);
        Button botao_consulta = (Button) findViewById(R.id.botao_consulta);
        Button botao_cadastro = (Button) findViewById(R.id.botao_cadastro);
        Button botao_alterar = (Button) findViewById(R.id.botao_alterar);
        ImageButton botao_sair = (ImageButton) findViewById(R.id.sair_principal);
```

Figura 6.2: Código Tela Inicial

Tela de Seleção do Cadastro

Na tela de seleção de cadastro é apresentado duas opções ao usuário, uma opção para o usuário cadastrar disciplinas e uma outra opção para efetuar o cadastro de alunos. É necessário que o usuário escolha uma das opções para poder prosseguir, e para que possa realizar o cadastro de alunos no sistema é necessário que se tenha pelo menos uma disciplina cadastrada.

Esta interface é construída utilizando RelativeLayout e para a seleção foi usado um RadioGroup com duas opções na qual somente uma pode ser escolhida. Esta tela está representada na Figura 6.3.

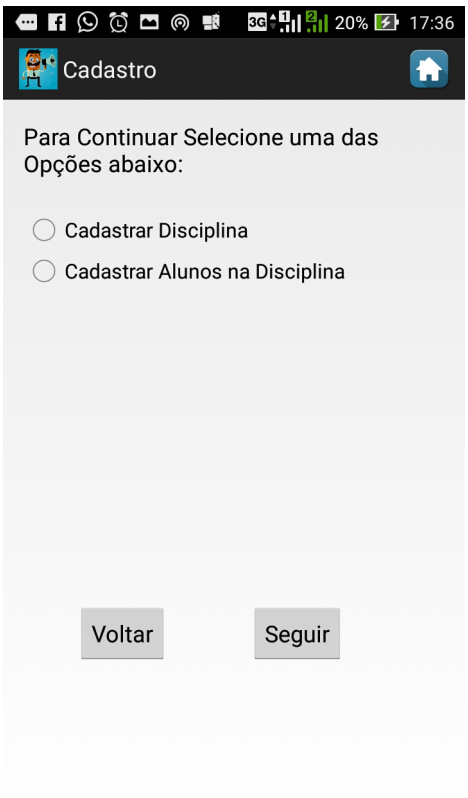

Figura 6.3: Tela de Seleção de Cadatro

Tela de Cadastro de Disciplinas

Na tela de cadastro de disciplinas o usuário irá inserir os dados referentes a cada disciplina. Os dados são os seguintes: Nome da disciplina, código, nome do professor e ano/semestre.

Esta tela foi desenvolvida usando RelativeLayout e possui componentes do tipo

TextView e EditText e Botton. Podemos observar esta tela na Figura 6.6.

Figura 6.4: Tela de Cadastro de Disciplinas

Tela de Cadastro de Alunos

Na tela de cadastro de alunos o usuário irá inserir os dados referentes aos alunos que estão matriculados em suas disciplinas, os dados são: Número de matrícula, nome do aluno, nome da displina e código da disciplina . Os dados referentes a disciplina são preenchidos automaticamente quando o professor escolhe a disciplina em que o aluno será cadastrado, os dados desda disciplina são carregados do banco de dados.

Esta tela foi desenvolvida usando RelativeLayout e possui componentes do tipo TextView e EditText e Botton. Podemos observar esta tela na Figura 6.5.

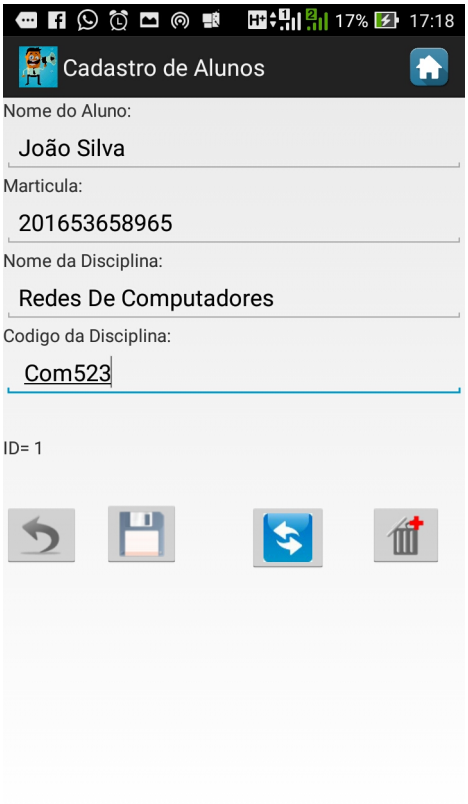

Figura 6.5: Tela de Cadastro de Alunos

Tela de Alterar Dados

O aplicativo possui a função de alterar os dados cadastrados, o professor pode escolher essa opção clicando na botão alterar dados na tela principal. Esta opção irá direcionar o professor a outra tela onde o mesmo escolherá a disciplina ou o aluno que irá ser modificado. Na tela de alterar dados tem-se a opção de atualizar dados e também a opção de excluir um aluno ou uma disciplina. Na figura 6.6 pode observar a tela de alterar os dados. esta opção utiliza a mesma tela de cadastro.

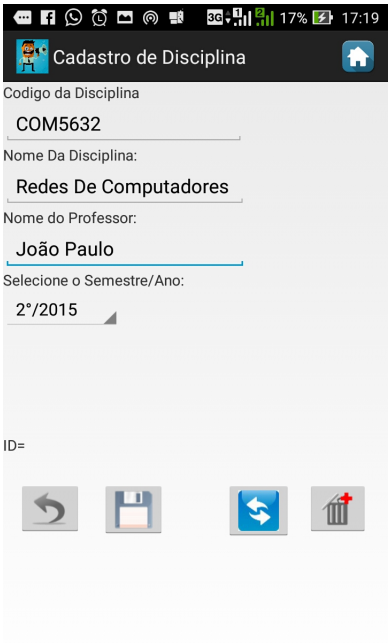

Figura 6.6: Tela de Altera Dados

Tela de Seleção de Disciplina

A tela de seleção de disciplina é responsável por listar todas as disciplinas cadastradas. Esta tela é exibida em 3 cenários do aplicativo, são eles: Quando o usuário deseja cadastrar um aluno, quando o usuário deseja realizar chamada e quando o usuário deseja emitir o relatório de frequência.

Esta tela foi desenvolvida usando RelativeLayout e possui componentes do tipo TextView e ListView e Botton. Pode-se observar esta tela na Figura 6.7.

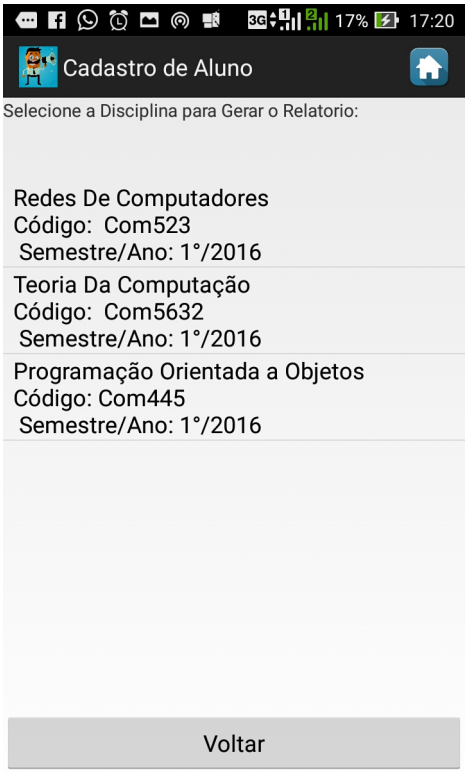

Figura 6.7: Tela de Seleção de Disciplina

Na Figura 6.8 têm-se dois métodos, o primeiro é encarregado de criar a ListView, ou seja, criar a lista com os componentes que serão exibidos na tela. O segundo método é o OnItemClick que é responsável por pegar a posição do item da lista que foi clicado e armazenar suas informações para serem enviados a tela seguinte.

```
BDaplicacao selecao = new BDaplicacao(this);
final ListView busca_disciplina2 = (ListView) findViewById(R.id.mostra_disciplinas);
mostra_disciplinas2 = (List<Disciplina>) selecao.selectdisciplina();
ArrayAdapter<Disciplina> adp = new ArrayAdapter<Disciplina>
(this, android.R.layout.simple_list_item_1, mostra_disciplinas2);

busca_disciplina2.setAdapter(adp);

busca_disciplina2.setOnItemClickListener(new OnItemClickListener() {

    @Override
    public void onItemClick(AdapterView<?> arg0, View arg1, int position,
            long id) {

        Disciplina  disc = (Disciplina) busca_disciplina2.getItemAtPosition(position);

        Intent intent2 = new Intent(Escolhe_Disciplina2.this, Cadastrar_Aluno.class);

        intent2.putExtra("CODIGO_DISC", disc.getCodigo().toString());
        intent2.putExtra("NOME_DISC", disc.getNome_disc().toString());
        intent2.putExtra("ID_ALUNO", disc.getId_disc().toString());

        startActivity(intent2);
        finish();
```

Figura 6.8: Código ListView

Tela de Realizar Chamada

Na tela de realizar chamadas são exibidos os dados da disciplina e posteriormente são listados todos os alunos cadastrados na mesma. Em cada item da lista é exibido o nome do aluno e um CheckBox que o professor deve marcar, caso o aluno esteja presente na aula.

Esta tela foi desenvolvida usando RelativeLayout e possui componentes do tipo TextView e ListView, CheckBox e Botton. Podemos observar esta tela na Figura 6.9.

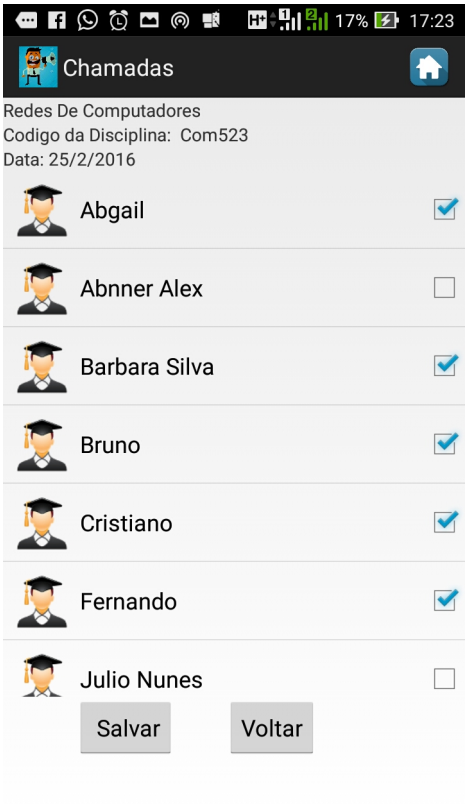

Figura 6.9: Tela de Realizar Chamadas

A Figura 6.10 mostra o trecho do código responsável pela criação do layout dinâmico. Este método infla o layout para cada item da lista, o objeto do tipo LayoutInflater recebe um arquivo XML e o transforma em uma View com as informações que serão exibidas na tela.

```
public View getView(int position, View convertView, ViewGroup parent) {
    final Aluno aluno = lista.get(position);

    LayoutInflater inflater = (LayoutInflater) context.getSystemService
            (Context.LAYOUT_INFLATER_SERVICE);

     View layout = inflater.inflate(R.layout.infla_aluno, null);
     TextView nometeste = (TextView) layout.findViewById(R.id.nome_final);
     nometeste.setText(aluno.getNome_aluno());
```

Figura 6.10: Método Layout Dinâmico

Tela de Emissão de Relatório

Ao escolher a disciplina o usuário é direcionado para a tela onde é gerado o relatório. O relatório é exportado na forma de um arquivo que será salvo na pasta do aplicativo, este

arquivo contem os dados da disciplina, dos alunos e as frequências até a data da emissão.

Esta tela foi desenvolvida usando RelativeLayout e possui componentes do tipo TextView e Botton. Podemos observar esta tela na Figura 6.11.

Figura 6.11: Tela de Emissão de Relatório

6.1.6 Modelagem do Banco de Dados

O banco de dados é o conjunto de registros de um sistema, eles são armazenados de forma estruturada de modo a facilitar a busca e geração de informação para aqueles que as utilizam.

Realizar a modelagem do banco de dados no desenvolvimento de software é uma etapa importante do projeto. Através da modelagem é possível visualizar como os dados serão organizados, facilitando a implementação e a manutenção do banco.

A Figura 6.12 apresenta a modelagem lógica do banco de dados do sistema. Estão representadas três tabelas com os seus respectivos campos e também o relacionamento entre elas.

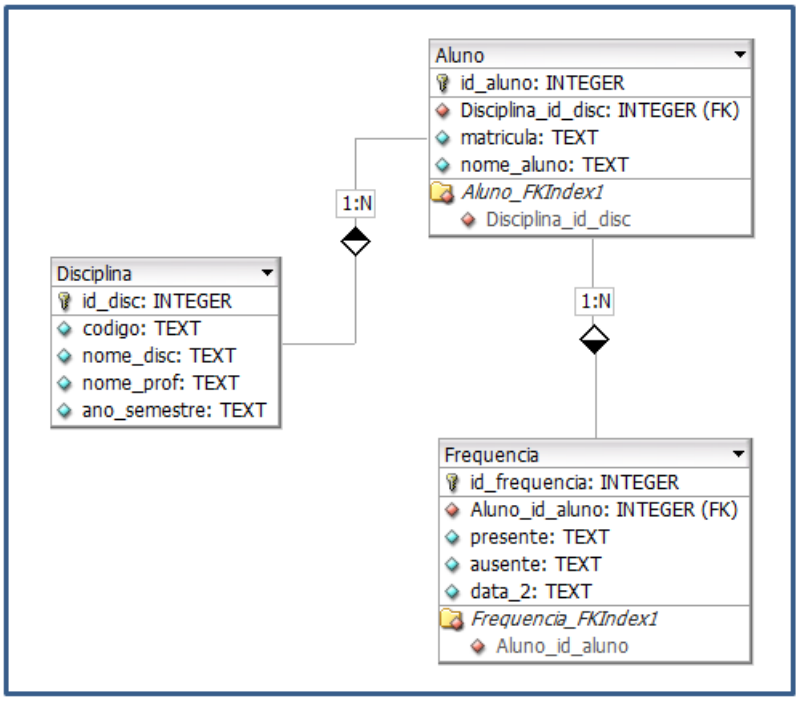

Figura 6.12: Modelagem do Banco de Dados da Aplicação

O banco de dados da aplicação foi desenvolvido utilizando a API do Android para o SQLite. Foram criados scripts SQL como mostra a Figura 6.13. Os scripts são executados pela API do Android para a criação das tabelas e parar popular o banco.

```
public void onCreate(SQLiteDatabase db) {
    String TABELA_DISC = "CREATE TABLE Disciplina ("
            + "id_disc INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,"
            + "codigo TEXT NOT NULL UNIQUE,"
            + "nome_disc TEXT NOT NULL,"
            + "nome_prof TEXT NOT NULL,"
            + "ano_semestre TEXT)";

    String TABELA_ALUNO = "CREATE TABLE Aluno ("
            + "Id_aluno   INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,"
            + "Disciplina_id_dis INTEGER NOT NULL"
            + "matricula  TEXT    NOT NULL UNIQUE,"
            + "nome_aluno TEXT    NOT NULL,";

    String TABELA_FREQUENCIA = "CREATE TABLE Frequencia ("
            + "Id_frequencia   INTEGER PRIMARY KEY AUTOINCREMENT NOT NULL,"
            + "Aluno_id_aluno INTEGER NOT NULL,"
            + "data       TEXT  NOT NULL,"
            + "presenca   TEXT,"
            + "ausencia   TEXT)";
```

Figura 6.13: Script em SQL para a Criação das Tabelas

Capítulo 7

Testes

Neste capítulo serão apresentados os testes feito com o sistema de Gerenciamento de Informações Educacionais.

Os principais pontos a serem testados no aplicativo foram: a entrada de dados, a persistência dos dados, facilidade na execução da chamada e a emissão do relatório de frequência dos alunos.

A entrada de dados é feita através do cadastro da disciplina no aplicativo e posteriormente o cadastro dos alunos que irão cursar essa disciplina durante o semestre. Veja na Tabela 7.1 os dados que foram cadastrados.

DISCIPLINA: Redes de Computadores CÓDIGO DA DISCPLINA: COM026 PROFESSOR: Alessandro vivas Ano/Semestre: 2015/02	
ALUNOS MATRICULADOS	NÚMERO DA MATRÍCULA
Adriano Silva	20101016022
Alvaro da Cunha	20101016087
Beatriz Pires	20101015543
Bernardo Nunes	20111024563
Bianca Rodrigues	20111023456
Carlos Henrique	20111026789
Carolina Nunes	20121015678
Daniel Silva	20121013456
Evellyn de paula	20131023456
Fernanda Vasconcelos	20131023456
Fabricio Marques	20141014567
Igor de jesus	20141014567
Mateus Souto	20151016785
Pedro Henrique	20151013459

Tabela 7.1: Dados Cadastrados no Sistema

No teste foi efetuado o cadastro da disciplina de Redes de Computadores ofertada no segundo semestre do ano de 2015, contendo 14 alunos matriculados. Dentre os dados cadastrados temos: nome da disciplina, código, nome do professor, ano/semestre, nome do aluno e matrícula.

Para verificar a persistência dos dados no aplicativo foi feito a busca das disciplinas e dos alunos cadastrados no sistema. Os dados são exibidos na tela e agrupados em formas de lista. Na Figura 7.1 pode-se observar o resultado das buscas feitas no aplicativo. Na esquerda da imagem tem-se a disciplina cadastrada e na direita os alunos que se encontram matriculados .

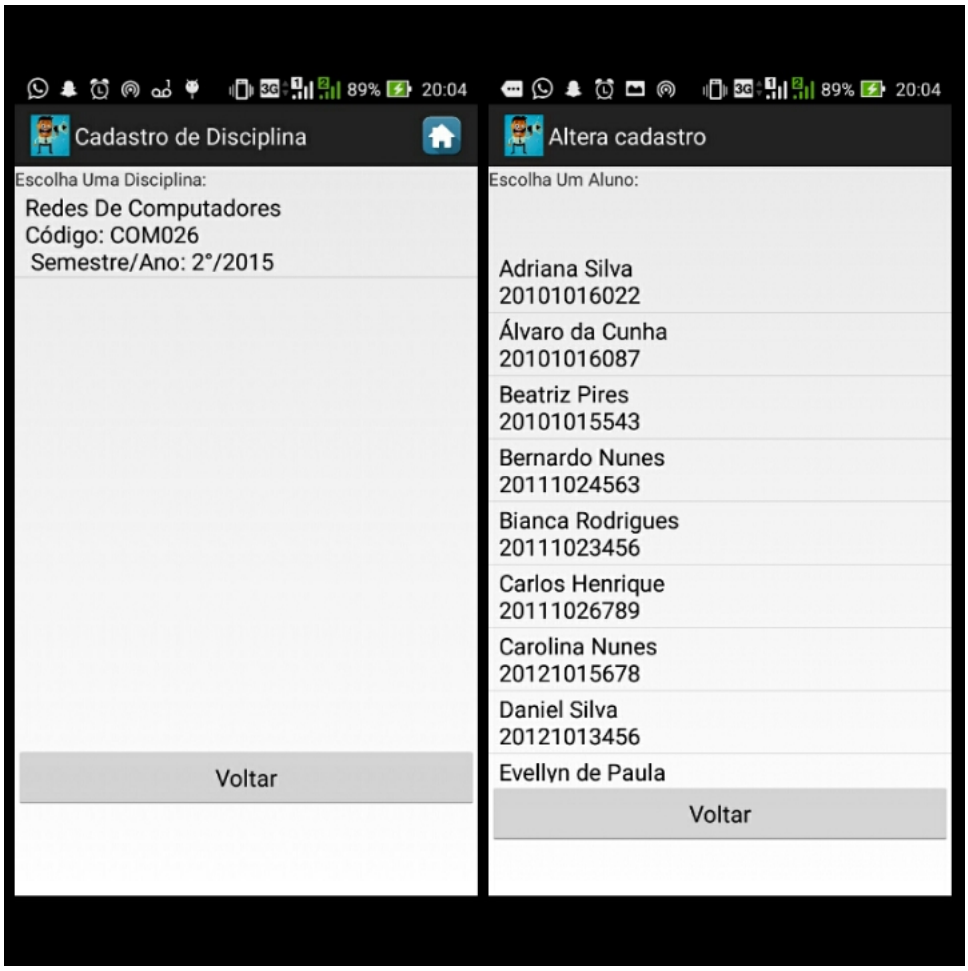

Figura 7.1: Disciplinas e Alunos cadastrados no Sistema

Para gerar o relatório referente à frequência dos alunos durante o semestre o sistema realiza consultas no banco de dados do aplicativo e agrupa todos os dados em um arquivo de texto. Este arquivo é exportado para um diretório específico e fica armazenado na memória do dispositivo. Para que o aplicativo possa acessar e escrever direto na memória é

necessário obter a permissão do sistema operacional. Essa permissão deve ser especificada no código fonte do aplicativo, dentro do arquivo AndroidManifest.xml, como podemos ver na Figura 7.2.

Figura 7.2: Permissão para acessar e escrever na memória do dispositivo

Ao usuário escolher a opção de exportar o relatório o sistema gera e armazena um arquivo com nome “RelatorioFrequencia” com a extensão “.csv” na memória do dispositivo. Pode-se ver um exemplo de relatório gerado na Figura 7.3.

Figura 7.3: Armazenamento do Relatório

O relatório de frequência é organizado por colunas contendo: nome do aluno, matrícula, data da chamada, nome da disciplina, código, presença e ausência. A chamada de cada aluno é agrupada em sequência para facilitar verificação da frequência. O arquivo do relatório pode ser executado por qualquer software editor de planilhas, como: Excel da Microsoft e Calc Libreoffice. Na Tabela 7.2 pode-se observa parte de um relatório que foi exportado do aplicativo.

ALUNO	MATRICULA	DATA	DISCIPLINA	CODIGO	PRESENÇA	AUSENCIA
Adrina Silva	20101016022	03/03/2015	Redes de Computadores	COM026	PRESESENTE	*
Adrina Silva	20101016022	06/03/2105	Redes de Computadores	COM026	*	AUSENTE
Adrina Silva	20101016022	11/01/2015	Redes de Computadores	COM026	PRESESENTE	*
Beatriz Pires	20101015543	03/03/2015	Redes de Computadores	COM026	PRESESENTE	*
Beatriz Pires	20101015543	06/03/2105	Redes de Computadores	COM026	PRESESENTE	*
Beatriz Pires	20101015543	11/01/2015	Redes de Computadores	COM026	*	AUSENTE
Bernardo Nunes	20111024563	03/03/2015	Redes de Computadores	COM026	PRESESENTE	*
Bernardo Nunes	20111024563	06/03/2105	Redes de Computadores	COM026	PRESESENTE	*
Bernardo Nunes	20111024563	11/01/2015	Redes de Computadores	COM026	*	AUSENTE
Bianca Rodrigues	20111023456	03/03/2015	Redes de Computadores	COM026	PRESESENTE	*
Bianca Rodrigues	20111023456	06/03/2105	Redes de Computadores	COM026	PRESESENTE	*
Bianca Rodrigues	20111023456	11/01/2015	Redes de Computadores	COM026	*	AUSENTE
Carlos Henrique	20111026789	03/03/2015	Redes de Computadores	COM026	PRESESENTE	*
Carlos Henrique	20111026789	06/03/2105	Redes de Computadores	COM026	PRESESENTE	*
Carlos Henrique	20111026789	11/01/2015	Redes de Computadores	COM026	*	AUSENTE
Carolina Nunes	20121015678	03/03/2015	Redes de Computadores	COM026	PRESESENTE	*
Carolina Nunes	20121015678	06/03/2105	Redes de Computadores	COM026	PRESESENTE	*
Carolina Nunes	20121015678	11/01/2015	Redes de Computadores	COM026	PRESESENTE	*
Daniel silva	20121014567	03/03/2015	Redes de Computadores	COM026	*	AUSENTE
Daniel silva	20121014567	06/03/2105	Redes de Computadores	COM026	PRESESENTE	*
Daniel silva	20121014567	11/01/2015	Redes de Computadores	COM026	*	AUSENTE

Tabela 7.2: Relatório de Frequência

No final dos testes pode-se concluir que o aplicativo de Gerenciamento de informações Educacionais cumpre com a sua finalidade de efetuar a chamada e facilitar o controle de frequência através do relatório. É relevante destacar também que todos os dados inseridos no aplicativo podem ser acessados, atualizados ou excluidos a qualquer momento pelo usuário.

Capítulo 8

Conclusão e Trabalhos Futuros

Com o desenvolvimento do sistema móvel de gerenciamento de informações educacionais notou-se uma grande melhoria tanto na forma de realizar a chamada como também no controle da frequência dos alunos. O aplicativo é bem preciso e com funcionalidades objetivas, o que possibilita o professor realizar a chamada de forma prática e rápida. Os dados relativos a frequência de cada aluno são organizados de forma clara garantindo uma melhor apuração no final de cada semestre.

A utilização cada vez mais dos dispositivos móveis dentro do ambiente de trabalhos seja ele educacional ou não, motivou ainda mais para o desenvolvimento deste aplicativos, vemos nisso uma grande oportunidade de criação de ferramentas para facilitar o dia a dia dos profissionais e também impulsionar a produtividade.

Como resultado final deste trabalho temos o sistema móvel para lançamento e controle de frequências dos alunos. O sistema esta disponível para dispositivos móveis com plataforma Android, versão do sistema 2.4 ou superior. O sistema é gratuito e pode ser adquirido na Loja virtual do Google, Google Play.

Como trabalho futuros pretendemos atualizar o aplicativo para uma versão web, onde o repositório de frequência poderão ser consultados via internet tanto pelo professores quanto pelos alunos e também a criação de novos módulos que facilite a inserção dos dados no sistema.

Referências Bibliográficas

[1] AVANCINI, H. B. . Da qualidade à responsabilidade na educação superior. Revista Científica Trajetória Multicursos, v. III, p. 96-104, 2011

[2] MATEUS, R. G.; LOUREIRO, F. A. Introdução a Computação à Móvel. -1.ed-. Departamento de Ciência da Computa cão da UFMG, 1998.

[3] MONTEIRO, J. B. Google Android, Crie Aplicações para Celulares e Tablets. -1.ed-. Casa do Codigo, São Paulo-SP, 2012.

[4] MAZIEIRO, C. A. Sistemas Computacionais: Conceitos e Mecanismos. Departamento Acadêmico de Informática da UTFPR, 2014.

[5] PEREIRA, L. C. O.: SILVA, M. L. Android para Desenvolvedores - Rio de Janeiro : Brasport, 2009.

[6] SACCOL, Amarolinda Iara da Costa Zanela; REINHARD, N. Tecnologias de Informação Móveis, sem Fio e Ubíquas: definições, mapeamento do Estado-da-Arte e oportunidades de pesquisa. In: XXVIII Encontro da ANPAD, 2004, Curitiba. XXVIII Encontro da ANPAD, 2004. v. 1. p. 1-15

[7] OLIVEIRA R. L.; MEDINA D. R. Desenvolvimento de Objetos de Aprendizagem para Dispositivos Móveis: Uma Nova Abordagem que Contribui para a Educação. Novas Tecnologias na Educação, Rio Grande do Sul, v.5 n.1, Julho, 2007.

[8] LECHETA, Ricardo R. Google Android: aprenda a criar aplicações para dispo-sitivos móveis com o Android SDK. Novatec, 2010. ISBN 978-85-7522-244-7.

[9] ANDROID DEVELOPER. Develop. Disponívem em: < http://developer.android.com/sdk/index.html > Acesso em: 10/09/2014.

[10] SEBRAE. Aplicativos para celulares movem mercado. Disponível em: <http://www.sebrae2014.com.br/Sebrae2014/Alertas/Aplicativos-para-celulares-movem- mercado-bilion

[11] BUDIU, R. Mobile: Native Apps, Web Apps, and Hybrid Apps. Disponivel em: < http://www.nngroup.com/articles/mobile-native-apps/ > Acesso em: 05/11/2014.

[12] MORETTI João. Apps Corporativos Para Alcançar Resultados. Disponível em:< http://www.mobilepeople.com.br/pt/> Acesso em: 02/11/2014.

[13] IBM. Introdução à Plataforma Eclipse. Disponível em : <http://www.ibm.com/developerworks/br/library/os-eclipse-platform/> Acesso em: 11/09/2014.

[14] FUGERI S. O papel das linguagens de marcação para a Ciência da Informação. TransInformação, Campinas, v. 18, n. 3, set./dez., 2006. Disponível em:< http://periodicos.puc-campinas.edu.br/seer/index.php/transinfo/article/view/670> Acesso em: 20/11/2014.

[15] ALMEIDA M. B. Uma introdução ao XML, sua utilização na Internet e alguns conceitos complementares. Ci. Inf., Brasília, v. 31, n. 2, p. 5-13, maio/ago. 2002.

[16] SQLite. About SQLite. Disponível em:< http://www.sqlite.org/about.html> Acesso em: 22/11/2014. RESENDE, Antônio Maria Pereira de; SILVA, Claudiney Calixto da. Programação Orientada a Aspectos em Java. Rio de Janeiro: Brasport, 2005.

[17] Hire Mobile Developer, Google : Android N Switches To Oracle?s OpenJDK From Java API. Disponível em: < http://www.hiremobiledeveloper.com/blog/category/android/ > Acesso em : 23/03/2015

[18] Orientação a objetos: Conceitos Básicos. Disponível em:<http://www.macoratti.net> Acesso em: 02/12/2014.

[19] Analysis Tecnologia e Informação, O PDA. Disponível em:< https://analysistecnologia.wordpress.com/tipos-decomputadores/o-pda/ > Acesso em: 09/12/2014.

[20] LEITE, Mario; RAHAL JÚNIOR, Nelson Abu Sanra. Programação Orientada ao Objeto: Uma Abordagem Didática. UNICAMP: Revista Infotec, 2002. Disponível em:<http://www.ccuec.unicamp.br> Acesso em : 29/11/2014.

[21] SINTES, Anthony. Aprenda Programação Orientada a Objetos. São Paulo: Makron Books, 2002.

[22] MOURA, Adelina. Geração Móvel: um ambiente de aprendizagem suportado por tecnologias móveis para a Geração Polega. Disponível em: <http://adelinamouravitae.com.sapo.pt/gpolegar.pdf> Acesso em: 12/02/2016.

[23] SILVA M. G. M.; CONSOLO A. T. Uso de dispositivos móveis na educação - o SMS como auxiliar na mediação pedagógica de cursos a distância. 2008. Disponível em: < www.5e.com.br/infodesign/146/Dispositivos-moveis.pdf > Acessado em: 12/02/2016.

[24] OLIVEIRA L. R.; MEDINA R. D. Desenvolvimento de objetos de aprendizagem para dispositivos móveis: uma nova abordagem que contribui para a educação. V. 5 Nº 1, Julho, 2007. Disponívem em: <http://www.cinted.ufrgs.br/ciclo9/artigos/4aLeandro.pdf> Acesso em: 12/02/2016.

[25] ARAUJO R. B.;Computação Ubíqua: Princípios, Tecnologias e Desafios. XXI Simpósio Brasileiro de Redes de Computadores, 2003. Disponível em: <http://professordiovani.com.br/rw/monografia-araujo.pdf> Acesso em: 13/02/2016.

[26] CAPPELLOZZA A.; MORAES G. H. S. M. A influência da infraestrutura de tecnologia da informação sobre a mobilidade computacional de usuários e a computação em nuvem. REVISTA DE TECNOLOGIA APLICADA (RTA) , Vol. 2, No. 3, Set-Dez 2013, p.03-15. Disponível em: <http://www.faccamp.br/ojs/index.php/RTA/article/view/764> Acesso em: 13/02/2016.

[27] COSTA N. P .O; FILHO N. F. D.; DUARTE A. F. Comparative Evaluation of Operating Systems For Devices Mobile: Focus On Functionality. 9ª CONTECSI, 2012. Disponível em:<http://www.contecsi.fea.usp.br/> Acesso em: 13/02/2016.

[28] SILBERSCHATZ, A; GALVIN, P. B.; GAGNE, G. (2007). Sistemas Operacionais com Java.7º Edição. Trad. Daniel Vieira. Sup. Téc. Sergio G. Souza. Rio de Janeiro: Elsevier, Editora Campus.

[29] GUTIERREZ R. M. V.; Crossetti CROSSETTI P. A. A Indústria de Telequipamentos no Brasil:evolução Recentes e Perspectivas. BNDES Setorial, Rio de Janeiro, n. 18, p. 23-90, set. 2003.

[30] FIGUEIREDO C. M. S.; NAKAMURA E.Computação Móvel: Novas Oportunidade e Novos Desafios. 2015.Disponível em: <https://www.researchgate.net/publication/268435975> Acesso em: 15/02/2016.

[31] Mochileiro Digital, O que é um tablet e o que ele faz. Disponível em: <http://www.mochileirodigital.com.br/tecnologia/eletronicos/o-que-e-um-tablet/> Acesso em: 16 de Fevereiro de 2016.

[32] Moreira E.; Os gestos nas telas sensíveis ao toque são realmente intuitivos. 2013. Disponível em: <http://www.oeduardomoreira.com.br/os-gestos-nas-telas-sensiveis-ao-toque-sao-realmente-intuitivos/> Acesso em: 16 de fevereiro de 2016.

[33] Developer Mozilla, Começando a construir aplicativos. Disponível em: <https://developer.mozilla.org/pt-BR/Apps/Getting-Started> Acesso em: 16 de Fevereiro de 2016.