UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL**

RUHAN BRUNO CAROLINO DA SILVA

PRESENCE

Desenvolvimento de um aplicativo *Android* para Gerenciamento de Frequência Acadêmica

Bandeirantes

2012

RUHAN BRUNO CAROLINO DA SILVA

PRESENCE

Desenvolvimento de um aplicativo *Android* para Gerenciamento de Frequência Acadêmica

Trabalho de Conclusão de Curso apresentado à Universidade Estadual do Norte do Paraná – *Campus* Luiz Meneghel – como requisito parcial para obtenção do grau de bacharel em Sistemas de Informação.

Orientador: Prof. Me. Cristiane Yanase Hirabara de Castro

Bandeirantes

2012

RUHAN BRUNO CAROLINO DA SILVA

PRESENCE

Desenvolvimento de um aplicativo *Android* para Gerenciamento de Frequência Acadêmica

Trabalho de Conclusão de Curso apresentado à Universidade Estadual do Norte do Paraná – *Campus* Luiz Meneghel – como requisito parcial para obtenção do grau de bacharel em Sistemas de Informação.

COMISSÃO EXAMINADORA

_____
Prof. Me. Cristiane Y. Hirabara de Castro
UENP – *Campus* Luiz Meneghel

_____
Prof. Me. Bruno Miguel N. de Souza
UENP – *Campus* Luiz Meneghel

_____
Prof. Me. Carlos Eduardo Ribeiro
UENP – *Campus* Luiz Meneghel

Bandeirantes, 27 de Junho de 2012.

RESUMO

Este trabalho de graduação consistiu no desenvolvimento do *PRESENCE* um aplicativo *Android* para Gerenciamento de Frequência Acadêmica. Um aplicativo *Android* é um programa de computador especifico para a plataforma *Android*. Essa plataforma foi desenvolvida em 2008 pela Google e consiste em um sistema operacional para dispositivos móveis do tipo *smartphones* e *tablets*. O gerenciamento de frequência acadêmica atualmente é realizado em universidades no ambiente de sala de aula por meio de uma pauta onde o professor registra uma data e o nome dos alunos que estão presentes em sala. O professor possui uma pauta para cada disciplina ministrada com uma lista de todos os alunos matriculados na disciplina. Objetivando simplificar e tornar a forma de gerenciar a frequência acadêmica mais interessante foi desenvolvido um aplicativo para *smartphone* denominado *PRESENCE*. O aplicativo proposto realiza o controle das disciplinas ministradas pelo professor, e gerencia a frequência de cada aluno matriculado de forma prática e criativa. O sistema possibilita integração com outros sistemas acadêmicos por meio de um padrão de arquivo XML e aplicativos nativos da plataforma *Android*, tais como, aplicativo de E-mail e leitor de código de barras. Para cada modulo desenvolvido do sistema foram criadas uma ou mais implementações sendo descritas as funcionalidades do sistema e a interface gráfica em conjunto com sugestões e possibilidades para desenvolvimentos futuros.

Palavras-chave: *Android*, Gerenciamento de Frequência Acadêmica, Dispositivos Móveis.

ABSTRACT

This graduate work was the development of an Android application to PRESENCE Frequency Management Academic. An android is an application specific computer program for the Android platform. This platform was developed by Google in 2008 and consists of an operating system for mobile devices like smartphones and tablets. The frequency management is currently conducted in academic universities in the environment of the classroom through an agenda where the teacher records the daily discipline and students who are present in the room. The teacher has an agenda for each subject taught with a list of all students enrolled in the course. Aiming to simplify and how to manage the academic often most interesting application was developed for a smartphone called PRESENCE. The proposed application performs the control of courses taught by the teacher, and manages the frequency of each student enrolled in a practical and creative. The system allows integration with other academic systems through a standard XML file and the Android native applications such as E-mail application and barcode reader. For each module of the developed system were created with one or more implementations described system functionality and user interface together with suggestions and possibilities for future developments.

Keywords: Android, Academic Frequency Management, Mobile Devices.

LISTA DE SIGLAS

API - *Application Prrogramming Interface*

CE - *Compact Embedded*

FSF - *Free Software Foundation*

GPL - *General Public License*

GPS - *Global Positioning System*

GPS - *Global Positioning System*

JVM - *Java Virtual Machine*

ME - *Micro edition*

MEC - Ministério da Educação

OHA -*Open Handset Alliance*

OO- Orientação a Objeto

OSI - *Open Source Iniciative*

PDA - *Personal Digital Assistant*

SDK - *Software Development Kit*

SO - Sistema Operacional

SQL - *Structured Query Language*

UML - *Unified Modeling Language*

UP - *Unified Process*

USB *– Universal Serial Bus*

XML -*Extensible Markup language*

LISTA DE FIGURAS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

O uso de dispositivos móveis do tipo *smartphones* e *tablets* tem crescido cada vez mais, eles estão presentes em todos os lugares, principalmente no ambiente de trabalho e no ambiente escolar (ANASTASIOS, 2008 apud Switzer e Csapo, 2005). Em um mundo onde a palavra “mobilidade” esta cada vez mais conhecida os usuários querem realizar o máximo de tarefas por meio de um dispositivo móvel. Pessoas desejam tirar foto, acessar internet, ouvir musica, pagar contas, usar GPS (*Global Positioning System)*, jogar e fazer ligações por meio de um único dispositivo.

Neste contexto, alguns bancos já oferecem serviços e aplicativos onde o cliente pode pagar suas contas através de um dispositivo. Já em outros países, como nos Estados Unidos, por exemplo, é possível pagar a compra do supermercado por meio de um dispositivo, tal como, *smartphone* ou *tablet*.

Sendo assim, podemos perceber que quanto mais recurso o dispositivo oferece mais interessante e útil ele se mostra. Dispositivos lançados recentemente no mercado dispõem de funções que vão muito além do que era esperado há alguns anos atrás, em termos de processamento e capacidade. De um simples meio de comunicação, agenda eletrônica ou *mp3 player*, passaram a ser dispositivos inteligentes ou *smartphones.* Esses dispositivos estão se tornando pequenos computadores que realizam tarefas complexas com um poder de processamento que cresce continuamente. (FILGUEIRAS, 2010).

Segundo LECHETA (2010), estudos mostram que hoje em dia mais de três bilhões de pessoas possuem um aparelho celular, e isso corresponde a mais ou menos metade da população mundial.

Junto com os dispositivos móveis o mercado de aplicativos móveis também evoluiu, isto é, o relacionamento entre usuários, desenvolvedores e fabricantes passou a ser mais dinâmico com o surgimento dos portais de aplicativos (Holzer *et al.* 2009).

Os portais de aplicativos têm como objetivo fornecer aos desenvolvedores um lugar em comum para disponibilizar suas aplicações, funciona

como um site do *YouTube*, ou seja, basta cadastrar e enviar seu aplicativo. Também é possível fazer avaliação de aplicativos disponíveis e comentários. (LECHETA, 2010).

O desenvolvimento de aplicativos comportados por esses portais está em alta no mercado de dispositivos móveis. Em 2012 há uma grande variedade de modelos de celulares com diferentes sistemas operacionais. Entre os sistemas operacionais podemos destacar *Windows Mobile* da *Microsoft*, *iPhone OS* da *Apple*, *Blackbarry OS* da *Blackbarry* e *Android* da Google.

Anunciado em setembro de 2007 pela Google, o *Android* apresenta uma boa perspectiva de crescimentos para os próximos anos como apresentado na Tabela 1.

Sistema Operacional	Participação no mercado em 2011	Previsão de participação no mercado em 2015	Crescimento (2011 – 2015)
Android	39.5%	45%	23.8%
BalckBerry	14.9%	13.7%	17.1%
iOS	15.7%	15.3%	18.8%
Symbian	20.9%	0.2%	-65.0%
Windows Mobile	5.5%	20.9%	67.1%
Outros	3.5%	4.6%	28.0%
Total	100%	100%	19.6%

Tabela 1 Previsão de crescimento para plataformas móveis (IDC, 2011).

Com a proliferação do sistema *Android* nestes dispositivos e com a promessa de aumento deste crescimento, aumentam também as possibilidades de utilização destes recursos. Já que esta plataforma possibilita um poder e uma liberdade muito grande aos seus usuários (UZEJKA, 2011).

O *Android* é um sistema operacional de código aberto, gratuito e baseado no *Kernel Linux*. Conhecido como pilha de software, pois junto com o sistema operacional ele traz um conjunto completo de aplicativos, chamados de aplicações nativas. Todas as aplicações para *Android* são construídas utilizando a linguagem Java.

Dentre as aplicações nativas tem destaque o Navegador, Discador, tocador de musica, galeria de imagem, gerenciador de contatos, calculadora, gerenciador de E-mail entre outras. O *Android* suporta ainda tecnologias como *touchscreen* (tela sensível ao toque) e GPS (*Global Positioning System* – Sistema de

Posicionamento Global) que permite localizar a posição em termos de latitude e longitude da terra. (MARTINS, 2009).

Segundo CARDOSO (2011), ao desenvolver aplicativos ou jogos para dispositivos móveis atualmente, o programador deve estar ciente da importância que representa a plataforma *Android* juntamente com o Java *ME* e o *iOS*. O *Android* abrange grande parte do mercado de sistemas operacionais para tablets e celulares.

Em outro contexto, o cenário atual em ambientes acadêmicos mostra que as universidades tem buscado cada vez mais desenvolver seus próprios produtos de *software*, objetivando o mercado de trabalho e o mercado consumidor. Como pode ser observado na pesquisa de MELO (2009) diversas instituições tem procurado desenvolver seus próprio produtos de *softwares* acadêmicos e administrativos. Outras optam por adquirir soluções de prateleiras disponíveis no mercado que nem sempre tratam os diversos processos acadêmicos e administrativos de forma integrada e tão pouco disponibilizam todas as informações demandadas pelo MEC.

Atualmente grande parte das universidades utilizam softwares para fazer o gerenciamento acadêmico e administrativo. Todo controle de frequência e notas dos discentes é feito por meio e um sistema acadêmico instalado em um servidor na universidade. Mas a questão é: como essas informações chegam até esse software acadêmico? Qual o esforço de trabalho envolvido nesse processo? E ainda, existem soluções melhores para assegurar o menor esforço e a segurança dessas informações? Responder essas questões é a grande motivação deste trabalho.

Antes das informações chegarem até o sistema acadêmico elas são registradas previamente em uma pauta. Uma pauta é geralmente conhecida com “livro de chamada” ou “livro de presença”. Cada docente possui uma pauta para cada disciplina que ministra, isto é, para cada disciplina existe uma lista de presença com o nome dos alunos matriculados na disciplina. Se o professor ministra quatro disciplinas na universidade ele terá que carregar consigo quatro “livros de chamada”.

Basicamente as informações contidas na pauta ou “livro de chamada” são: lista de nomes, data da aula, disciplina, nome do professor, notas e lista de presença.

O livro de presença é preenchido em sala de aula pelo professor, que registra a aula e verifica os alunos que estão presentes. Depois do livro de chamada preenchido, as notas, as aulas e principalmente a lista de presença devem ser registradas no sistema acadêmico da universidade.

O fato das aplicações para *Android* serem construídas utilizando a linguagem Java é um fator motivacional, tendo em vista, que a linguagem Java é uma das linguagens de programação mais utilizadas, por utilizar paradigmas como OO (Orientação a Objeto) e também por oferecer portabilidade.

Outro fator que motiva o uso da tecnologia Android é justamente por ser um sistema gratuito e de Software Livre, facilitando a customização do sistema operacional ou a substituição de aplicativos de forma prática.

Além disso, existe um conjunto de empresas reconhecidas no mundo da tecnologia e comunidades que participam no desenvolvimento da plataforma *Android*, tais como a Google, Motorola, Sansung que fazem parte da OHA(*Open Handset Alliance*).

Tendo em vista algumas facilidades que a plataforma *Android* oferece, e o aquecimento do mercado em relação ao desenvolvimento de aplicativos para dispositivos moveis, criou-se a oportunidade de desenvolver um aplicativo que auxilie a tarefa do professor.

O aplicativo tem como objetivo proporcionar ao professor uma forma alternativa e flexível de realizar o gerenciamento de frequência acadêmica por meio de um *smartphone*. O aplicativo possibilitará a integração com outros sistemas acadêmicos. O presente trabalho apresenta uma solução por meio do desenvolvimento de um aplicativo móvel capaz de auxiliar no gerenciamento de frequência acadêmica.

1.1 OBJETIVO

O objetivo é de proporcionar ao professor uma forma alternativa e flexível de gerenciamento de frequência, por meio do desenvolvimento de uma aplicação para dispositivo móvel para gerenciamento de frequência que possibilite integração com sistemas acadêmicos.

1.1.1 Objetivos específicos

Os objetivos específicos desse trabalho foram:

1. Modelar o aplicativo de acordo com uma abordagem de desenvolvimento de software escolhida;
2. Projetar o aplicativo com base nas especificações de requisitos;
3. Projetar estruturalmente a arquitetura do aplicativo, escolhendo ferramentas de software livre para desenvolvimento da aplicação;
4. Desenvolver o *Presence*;
5. Definir uma forma de comunicação entre a aplicação e o sistema acadêmico.

1.2 JUSTIFATIVA

Devido a grande importância que os dispositivos móveis tem hoje na vida das pessoas, o desenvolvimento de aplicações para celulares, *smartphones* entre outros dispositivos tem crescido muito.

A mobilidade que os dispositivos móveis oferecem nos permite carregar aplicativos das mais diversas características para qualquer lugar. Utilizando um *smartphone* ou *tablet* as pessoas podem acessar aplicativos uteis que auxiliam nas tarefas do dia a dia, além disso, é possível usar tais aplicativos independente de localização geográfica e a qualquer hora.

No contexto do ambiente universitário onde os professores carregam os livros de presença para cada disciplina ministrada. Tais livros poderiam ser substituídos por um aplicativo de celular. A função do livro seria agregada a um aplicativo que controlaria a frequência de cada aluno por disciplina, eliminando assim a necessidade do professor carregar livros de presença.

Tendo em vista a ausência de programas que fazem o gerenciamento de frequência através de dispositivos móveis cria-se uma oportunidade para o desenvolvimento dessa solução.

1.3 ORGANIZAÇÃO

O presente trabalho está organizado como se segue. No Capitulo II será apresentado uma introdução sobre aplicações para dispositivos móveis, apresentando também uma introdução aos dispositivos móveis. Também será descrito o Mercado de aplicativos móveis, plataforma *Android* e os conceitos relacionados a software livre. No Capitulo III é apresentada a proposta inicial do aplicativo *Presence bem como os* métodos e ferramentas escolhidos. O desenvolvimento do trabalho e os detalhes da implementação são apresentados no Capitulo IV, analogamente o Capitulo V apresenta a conclusão.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo serão apresentados os conceitos e definições relacionadas a aplicações móveis, *software* livre e o sistema operacional *Android*, que serviram como base para a proposta do aplicativo. Também são apresentados levantamentos teóricos de estudos realizados atualmente nessas áreas.

2.1 APLICAÇÕES PARA DISPOSITIVOS MÓVEIS

A tecnologia móvel vem conquistando cada vez mais o mercado, nos últimos anos os populares computadores de mão tornam-se um dos meios de comunicação móvel de maior crescimento (ANASTASIOS, 2008). Atualmente já existem mais de 5.000 modelos de dispositivos móveis no mercado (RUSSO *et al.*, 2010).

Houve um grande crescimento na variedade de dispositivos móveis que processam dados e acessam internet (ANASTASIOS, 2008 apud Anderson e BlackWood, 2004). Entre esses dispositivos estão os *iPods* (*smartphone*), telefones celulares, *tablets* e PDAs (*Personal Digital Assistant*).

O número desses aparelhos tem aumentado cada vez mais a medida que estão entre os dispositivos mais úteis e populares. (ANASTASIOS, 2008 apud Switzer e Csapo, 2005).

Fabricante	Unidades 2011	Fatia de mercado 2011 (%)	Unidades 2010	Fatia de mercado 2010 (%)
Nokia	105,353.5	23.9	117,461.0	28.2
Samsung	78,612.2	17.8	71,761.8	17.2
LG Electronics	21,014.6	4.8	27,478.7	6.6
Apple	17,295.3	3.9	13,484.4	3.2
ZTE	14,107.8	3.2	7,817.2	1.9
Research In Motian	12,701.1	2.9	12,508.3	3.0
HTC	12,099.9	2.7	6,494.3	1.6
Motorola	11,182.7	2.5	8,961.4	2.1
Huawei Device	10,668.2	2.4	5,478.1	1.3
Sony Ericsson	8,475.9	1.9	10,346.5	2.5
Others	148,990.9	33.8	135,384.1	32.5
Total	440,502.2	100	417,085.7	100

Tabela 2 Venda mundial de dispositivos móveis- (Gartner, 2011).

Na tabela 2 é possível observar o número de vendas em todo o mundo no período de 2010 e 2011 segundo (Gartner, 2011). É possível perceber o crescimento da venda de dispositivos móveis no período de 2010 para 2011 em todo o mundo. Tendo em vista essa mobilização do mercado móvel, empresas lideres de tecnologia, comunidades e desenvolvedores são estimulados a participar ativamente no mercado.

Há tempos que a computação móvel vem atraindo a atenção da comunidade cientifica (HOLZER, 2009 apud Forman e Zahojan, 1994). Consagrados atores que participam do mercado móvel tais como Google, Apple, Microsoft possuem seus próprios sistemas operacionais. Essas grandes empresas disponibilizam ferramentas e outros recursos para desenvolvedores de aplicações móveis.

Tendo em vista toda essa revolução que causam os dispositivos móveis, mais especificamente os *smartphones* e *tablets*, o suporte aos desenvolvedores e as oportunidades para desenvolvimento contribuem positivamente para o crescimento do mercado móvel (HOLZER *et al*., 2010).

2.2 MERCADO DE APLICATIVOS MÓVEIS

Esta seção apresenta as características do atual mercado de aplicações móveis e como se da o relacionamento entre desenvolvedores, portais de aplicativos e usuários.

O desenvolvimento do mercado móvel atual é movido por cinco grandes empresas, algumas delas fornecedoras de plataformas, sistemas operacionais e ferramentas para desenvolvimento.

São elas: Nokia com o Sistema Operacional (SO) Symbian, RIM com SO *Blackberry*, Microsoft com a família de SOs *Windows CE* e *WP (Windows Phone*) e a *Apple* com o SO *iPhone*.

A Nokia se uniu com a Microsoft para o lançamento do sistema operacional *Windows Phone Seven*, pois o seu sistema operacional que há alguns anos atrás dominava o mercado móvel perdeu uma quantidade considerável de espaço no mercado atual em 2012.

Além desses, o destaque fica para a plataforma *Android* da Google que resolveu lançar o seu próprio sistema operacional em 2008, e como esperado já ocupou seu lugar no mercado se tornando um competidor de peso no mercado móvel. (HOLZER *et al*.,2010).

Esses grandes atores são responsáveis por mudar significativamente a estrutura do mercado através da imposição e execução de suas próprias regras.

As implicações dessas mudanças não se referem apenas as operadoras de rede móvel e fabricantes de telefone móvel, elas também trazem oportunidade e limitações para os desenvolvedores de aplicações móveis. HOLZER *et al*., 2010).

Uma das mudanças mais importantes foi a criação de portais de aplicações móveis tal como, *Apple Store* e o *Google Play* (antigo *Android Market*) que possibilitam a centralização de aplicativos para a plataforma.

Com o surgimento de portais centralizados de aplicativos a distancia entre o desenvolvedor e o usuário foi diminuída, contribuindo positivamente e facilitando assim a disponibilização de aplicativos pela rede mundial de computadores.

2.3 DISTRIBUIÇÃO DE APLICATIVOS MÓVEIS

Com o portal de aplicativos o processo de distribuição ficou muito mais simples para todos os envolvidos e estimulou ainda mais os desenvolvedores a criar aplicações.

O processo de distribuição de aplicativos é demonstrado na Figura 1:

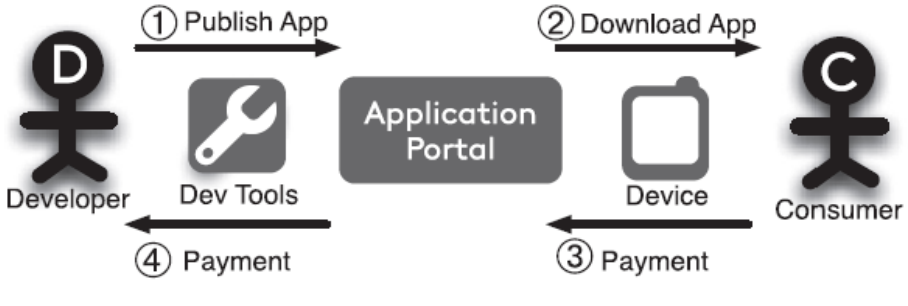

Figura 1 Processo de distribuição de aplicativos móveis fonte (HOLZER, 2010).

Na Figura 1 é possível observar o processo de distribuição, em que o desenvolvedor cria aplicativos utilizando as ferramentas disponibilizadas pela plataforma, após a criação do aplicativo o desenvolvedor publica sua aplicação no portal de aplicações. O usuário faz *download* da aplicação pelo seu dispositivo móvel, o pagamento é feito do cliente para o portal, que repassa o pagamento ao desenvolvedor.

2.4 VANTAGENS E DESVANTAGENS DO PORTAL DE APLICATIVOS

Com o portal de aplicativos diminui-se a distância entre o desenvolvedor e o usuário. Um único portal pode centralizar diversas aplicações facilitando na busca do usuário por um aplicativo novo, bem como na publicação de aplicativos pelo desenvolvedor (HOLZER *et al.*,2010).

Em contrapartida os desenvolvedores têm sua liberdade limitada, tornando-se dependentes das regras do portal que é controlado por uma empresa tal como a Google ou a Apple que impõe suas próprias regras.

Na **Erro! Fonte de referência não encontrada.** é apresentado o *looping* que demonstra o *feedback* positivo do mercado.

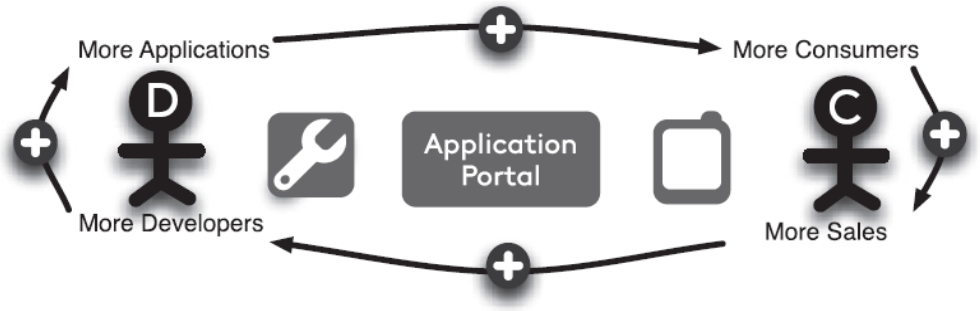

Figura 2 Feedback positivo para os dois lados do mercado (HOLZER, 2010).

Essa estrutura de mercado forma um *looping* favorecendo ambos os lados. Quanto mais aplicações mais usuários, quanto mais usuários mais vendas, quanto mais vendas mais desenvolvedores e por fim quanto mais desenvolvedores mais aplicações (HOLZER *et al.*,2010).

2.5 SOFTWARE LIVRE

Nesta seção serão abordadas todas as questões relacionadas ao desenvolvimento e utilização de *softwares* livres. Serão apresentadas abordagens e definições relacionadas a esse contexto.

A plataforma *Android* é um software livre, e por isso é importante para o desenvolvedor conhecer o que isso realmente significa, pois sabendo das possibilidades que um software livre oferece o *Android* pode ser melhor explorado.

Definição:

Software Livre é aquele que respeita a liberdade do usuário e da comunidade, segundo (STALLMAN, 2002) o termo "*Software* Livre" ou "*Free Software*" tem o significado Livre de Liberdade ou em inglês "*Free as in freedom*", pois o termo "*Free*" em inglês também significa grátis e pode ser confundido com *Software* Grátis.

Um *software* é denominado *Software* Livre se ele atende as seguintes Liberdades:

- Liberdade de executar o programa para qualquer proposito;
- Liberdade de modificar o programa de acordo com suas necessidades; isto é, liberdade para alterar o código fonte;
- Liberdade de redistribuir cópias gratuitamente ou mediante pagamento;
- Liberdade para distribuir versões modificadas do programa que a comunidade possa se beneficiar de suas melhorias.

Um programa só é considerado um *software* livre se ele segue todas essas liberdades. Programas de *software* livre em geral são de fácil acesso. Porém, a simples obtenção de um programa não significa que a pessoa pode fazer o que quiser com ele. As licenças de software livre são documentos através dos quais os detentores dos direitos sobre um programa de computador autorizam usos de seu trabalho que, de outra forma, estariam protegidos pelas leis vigentes no local. Além do uso como usuário final, esses usos autorizados permitem que desenvolvedores possam adaptar o software para necessidades mais específicas, utilizá-lo como fundação para construção

de programas mais complexos, entre diversas outras possibilidades (SABINO *et al.*, 2009).

As principais organizações internacionais responsáveis pela promoção e proteção do *software* livre são *Free Software Foundation* - (FSF) e *a Open Source Iniciative* – (OSI).

2.5.1 Vantagens do Software Livre

Estima-se que hoje há centenas de milhões de usuários de software livre no mundo. Se considerarmos também usuários indiretos, que usam serviços baseados em *software* livre, como Google, Amazon ou *eBay*, esse número é ainda maior (SABINO *et al.*, 2009).

Segundo (SABINO *et al.*, 2009) a principal vantagem do *software* livre é permitir o compartilhamento do código-fonte. Como resultado desse compartilhamento, a duplicação de esforços é evitada, quando mais de uma entidade está interessada no desenvolvimento de uma aplicação com características similares, o custo de desenvolvimento é reduzido.

REYMOND (2002) afirma que software livre tem condições de ter maior qualidade do que seus equivalentes fechados. Uma das justificativas para essa afirmação de Reymond é conhecida como “A Lei de Linus” que diz “dados olhos suficientes, todos os *bugs* (erros) são superficiais”.

Isso significa que, com o maior número de usuários que tem acesso ao programa e até ao código-fonte, o *software* é testado melhor e os problemas existentes no código são encontrados mais rapidamente. Outro fator que contribui para a qualidade é o orgulho pessoal do desenvolvedor, pois a partir do momento que seu código poderá ser lido por mais pessoas é esperado que ele seja mais cuidadoso com seu trabalho (SABINO *et al.*, 2009).

2.5.2 Desvantagens do Software Livre

HEXSEL (2002) diz que dentre as desvantagens associadas a utilização de software livre destacam-se:

- Interface de usuário não é uniforme nos aplicativos;
- Instalação e configuração podem ser difíceis;
- Mão de obra escassa ou custosa para desenvolvimento e suporte.

Também podemos levantar algumas desvantagens do *software* livre em relação a alternativas fechadas. Um dos principais motivos que leva uma empresa a optar por um *software* fechado quando há um similar livre disponível é a ausência de garantias e suporte. (SABINO *et al.*, 2009).

Dessa forma, em casos em que a empresa precisa fornecer garantias aos seus clientes, ou quando a indisponibilidade de um sistema pode causar grandes prejuízos, pode ser melhor que a empresa adquira uma solução em que eventuais problemas sejam delegados a um fornecedor ou que esse tenha que indenizar a empresa.

Já do ponto de vista de quem produz *software*, optar pelo modelo aberto pode ser visto como desvantagem á medida que a propriedade intelectual está exposta. (SABINO *et al*.,2009).

2.6 PLATAFORMA ANDROID

Para um bom desenvolvimento de uma aplicação para a plataforma é necessário conhecer muito bem seu funcionamento e sua arquitetura. Neste capitulo serão apresentados as características do sistema operacional Android.

Algumas particularidades que foram importantes na implementação do aplicativo *Presence* serão explicadas nos tópicos subsequentes.

O *Android* é mais conhecido como sistema operacional de código aberto para aplicativos móveis, porem oferece recursos e funcionalidades que vão muito além das funções de um Sistema Operacional.

Segundo LECHETA (2009) o *Android* é uma plataforma para *smartphones*, baseada no sistema operacional Linux, possui diversos componentes, com uma variada disponibilidade de bibliotecas e interface gráfica, além de disponibilizar ferramentas para a criação de aplicativos.

A plataforma *Android* pode ser descrita como uma junção de um sistema operacional com um *middleware* (programa de computador que faz a mediação entre outros *softwares*) dentre os *softwares* estão aplicações básicas que garantem o seu bom funcionamento (FILGUEIRAS, 2009). Recursos oferecidos pelo *Android*:

- *Framework* de aplicação que possibilita a reutilização e substituição dos componentes.
- Navegador integrado baseado no *WebKit*, cujo código fonte é aberto.
- Biblioteca gráfica 2D, Gráficos 3D baseados na especificação *OpenGL ES* 1.0 .
- SQLite para armazenamento estruturado de dados.
- Suporte para formatos de áudio, vídeo e imagem utilizados (MPEG4, H.264, MP3, AAC, AMR, JPG, PNG, GIF).
- Telefonia GMS (depende do hardware).
- *Bluetooth*, EDGE, 3G e *WiFi* (depende do hardware).
- Câmera, GPS, bússola e acelerômetro (dependente do hardware).

Uma das principais vantagens de se trabalhar com *Android* é poder usufruir do seu rico ambiente de desenvolvimento incluindo um emulador de dispositivo, ferramentas para *debug*, memória, desempenho e um *plug-in* para o Eclipse IDE (FILGUEIRAS, 2009).

A plataforma é baseada no sistema operacional Linux e oferece total liberdade para o usuário gerenciar as aplicações nativas bem como os recursos disponíveis. Criado pela Google em 2008 o seu desenvolvimento é mantido atualmente por um grupo de empresas líderes de mercado OHA – *Open Handset Alliance.*

Esse grupo OHA foi criado com a intenção de padronizar uma plataforma de código aberto e livre para celulares, justamente para atender todas as expectativas e tendências do mercado atual. (LECHETA, 2010).

Figura 3 Algumas das principais empresas da OHA - Open Handset Alliance Fonte: Autor

O fato de o *Android* ser de código aberto contribui muito para seu aperfeiçoamento, uma vez que desenvolvedores de todos os lugares do mundo podem contribuir para seu código fonte, adicionando novas funcionalidades ou simplesmente corrigindo falhas.

Já os desenvolvedores podem desfrutar de uma plataforma moderna com diversos recursos incríveis. (LECHETA, 2010).

2.6.1 Aplicações Nativas

As aplicações nativas são aplicativos que fazem parte do pacote de aplicações da plataforma *Android*. O *Android* possui um conjunto de aplicações básicas que fazem parte da camada de aplicações da plataforma.

Assim como os sistemas operacionais Linux e Windows possuem aplicações básicas para oferecer o mínimo de funções para os usuários, tais como navegador, calendário, bloco de notas entre outras, o *Android* dispões de aplicações do mesmo gênero chamadas de aplicações nativas.

Essas aplicações podem ser facilmente substituídas por outras aplicações que realizam a mesma função.

Algumas aplicações nativas do *Android* (OHA, 2012):

- Navegador
- Discador
- Galeria de Imagem
- Tocador de Música
- Gerenciador de Contatos
- Calculadora
- Alarme
- Mensageiro

Como pode ser visto na figura 4 uma das aplicações nativas do *Android* é a aplicação de E-mail. Uma aplicação completa que possibilita o envio de E-mail bem como a gerenciamento de uma conta de E-mail.

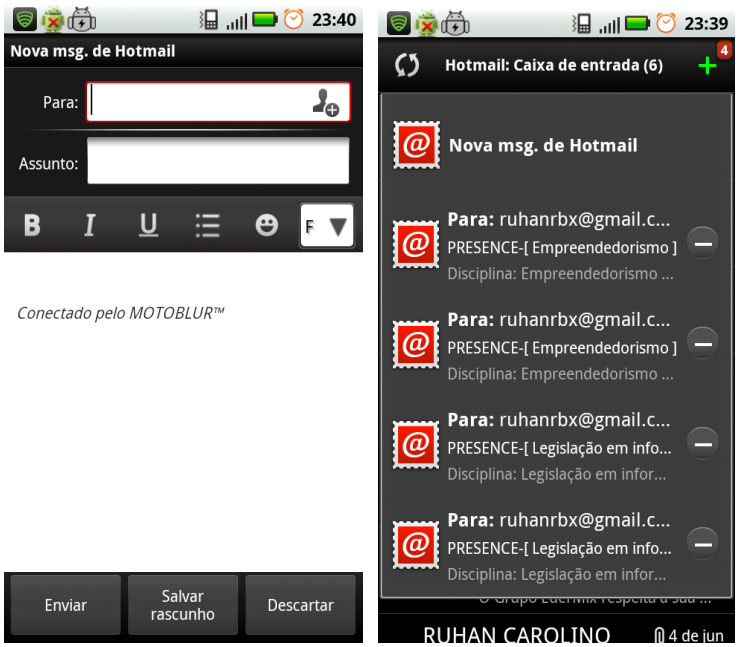

Figura 4 Aplicação nativa de E-mail Fonte: Autor

2.6.2 Arquitetura

Neste tópico são apresentados os componentes da arquitetura *Android* que podem ser observados na figura 5, em seguida é possível acompanhar o detalhamento de cada componente da arquitetura.

Na figura 5 a seguir é possível visualizar a organização da arquitetura *Android*.

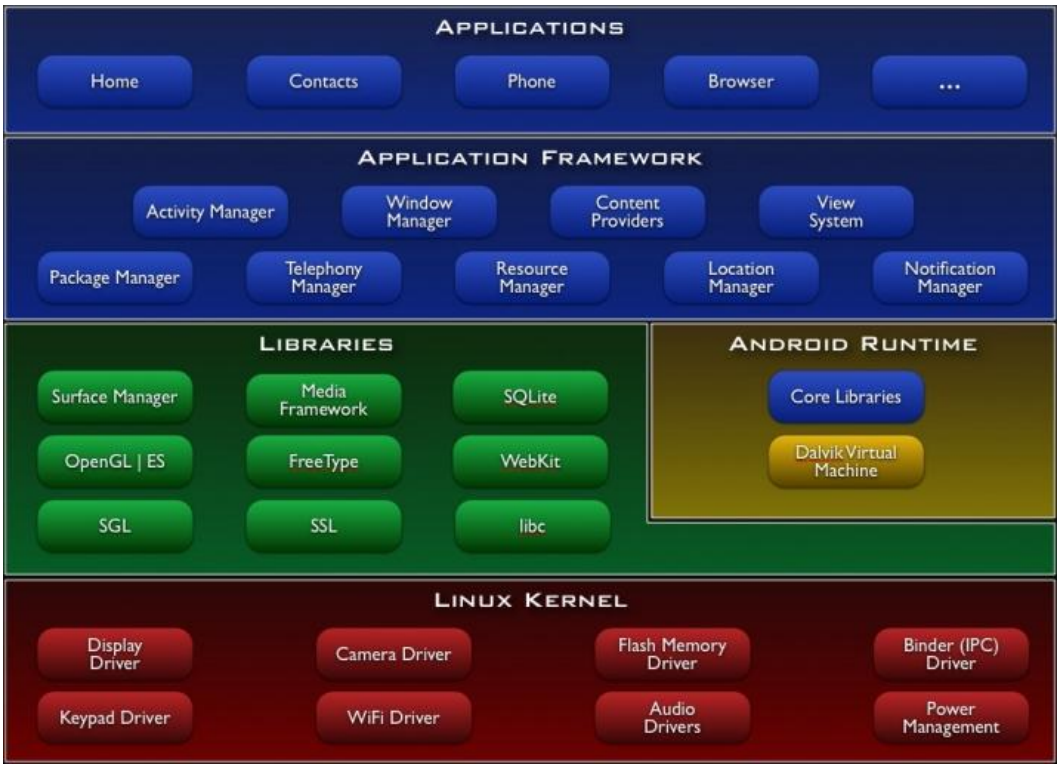

Figura 5 Arquitetura *Android* Fonte: (ANDROID, 2012).

2.6.3 Linux Kernel

Na Base da pilha se encontra a camada *Linux Kernel*. O Android possui o *Kernel* 2.6 do Linux, essa camada é responsável por fazer toda abstração do *hardware* para o resto do sistema.

O *Linux kernel* é responsável também pelo gerenciamento de memória, segurança do sistema, gerenciamento de processos, gerenciamento de energia, modelos de drivers, internet *WiFi* entre outros recursos (ANDROID, 2012).

2.6.4 Maquina Virtual Dalvik

Na camada em amarelo na figura 5, *Android Runtme* encontra-se a *Core Libraries* e *Dalvik Virtual Machine*. A *Core Libraries* ou Bibliotecas do Núcleo permitem que as aplicações para *Android* sejam escritas em Java. E por mais que as aplicações sejam escritas em Java, no *Android* elas não são mais executadas pela JVM – *Java Virtual Machine* como acontece em outras plataformas.

A *Dalvik Virtual Machine* é quem compila o código em Java e empacota em seguida com outros recursos utilizados pela aplicação gerando um arquivo .apk. Antes das classes serem empacotadas elas são compiladas para o formato *Dalvick Executable* (.dex), para que a maquina *Dalvick* consiga executa-las.

A *Dalvik Virtual Machine* foi otimizada especialmente para dispositivos móveis, com isso é possível executar varias instâncias da maquina virtual, uma para cada processo do sistema. (ANDROID, 2012).

2.6.5 Bibliotecas

Em verde na Figura 5 temos a camada de bibliotecas em C/C++, as quais são responsáveis por recursos utilizados por componentes de todo o sistema, são eles: SQLite - Banco de dados relacional do *Android*, *OpenGL* -Biblioteca 3D, SGL – *Engine* de Gráficos 2D, Biblioteca Multimídia ou *Media libraries* – Possibilita a gravação de vídeo e áudio e vários formatos e suporta vários outro tipos de arquivos, tais como, MPEG4, H.264,MP3,AAC, AMR,JPG, e PNG. (ANDROID, 2012).

2.6.6 Application Framework

Localizado na segunda camada em azul de cima pra baixo na figura. O *framework* de aplicações é uma das camadas mais importantes no *Android*, é uma das características que torna o *Android* mais interessante.

Basicamente essa camada possibilita a interação entre aplicações no sistema, com ela é possível construir uma aplicação que se comunique com qualquer outra aplicação instalada no sistema de uma forma fácil.

Todas as aplicações sendo elas nativas ou não, compartilham a mesma a API – (*Application Programming Interface* – Interface de Programação de Aplicativos). O *framework* também permite total controle sobre as aplicações nativas do sistema, tornando possível substitui-las por outras aplicações (ANDROID, 2012).

2.6.7 Aplicações

A camada no topo da figura é formada por todas as aplicações nativas, alguma delas são:

- E-mail
- SMS
- Calendário
- Mapas
- Navegador web
- Lista de Contatos
- Galeria de Imagem

Todas as aplicações são escritas utilizando a linguagem Java.

2.6.8 Fundamentos de um Aplicativo Android

Na implementação de um aplicativo *Android* existem quatro tipos de componentes que podem ser utilizados.

- *Activity*
- *Service*
- *BroadcastReceiver*
- *Content Provider*

Diferente do que é encontrado geralmente em outros sistemas, os aplicativos *Android* não possuem um único ponto de entrada, como uma função *main*() por exemplo. Tais componentes são instanciados pelo sistema e podem ser utilizados conforme a necessidade. (FILGUEIRAS, 2009).

Na implementação do aplicativo *Presence* apenas o componente *Activity* foi utilizado. Uma *Activity* ou uma Atividade representa uma tela (UI- *User interface*) da aplicação. Essa tela representa uma interação visual com o usuário, isso quer dizer que uma atividade esta sendo exibida para o usuário quando executada.

Para cada *Activity* existe uma interface (tela) que pode ser definida por um arquivo XML ou por linha de código.

Como já foi visto nos tópicos anteriores um aplicativo *Android* é implementado na linguagem Java, na Figura 6 é possível observar a representação de uma classe filha da classe *Activity* em linhas de código.

```
3  import android.app.Activity;
5
6  public class ProjetoActivity extends Activity {
7      /** Called when the activity is first created. */
8      @Override
9      public void onCreate(Bundle savedInstanceState) {
10         super.onCreate(savedInstanceState);
11         setContentView(R.layout.main);
12     }
13 }
```

Figura 6 Representação de uma Activity Autor

Para se desenvolver aplicativos consistentes para a plataforma *Android* é muito importante conhecer muito bem o ciclo de vida de uma *Activity*. Quem gerencia o ciclo de vida de uma *Activity* é o Sistema Operacional, porem é possível manipular o ciclo de vida controlando os métodos referentes a cada estado da atividade conforme as necessidades.

È necessário conhecer os possíveis estados em que uma *Activity* se encontra, que podem ser: executando, temporariamente interrompida em segundo plano ou completamente destruída como demonstra a Figura 7. (LECHETA, 2009).

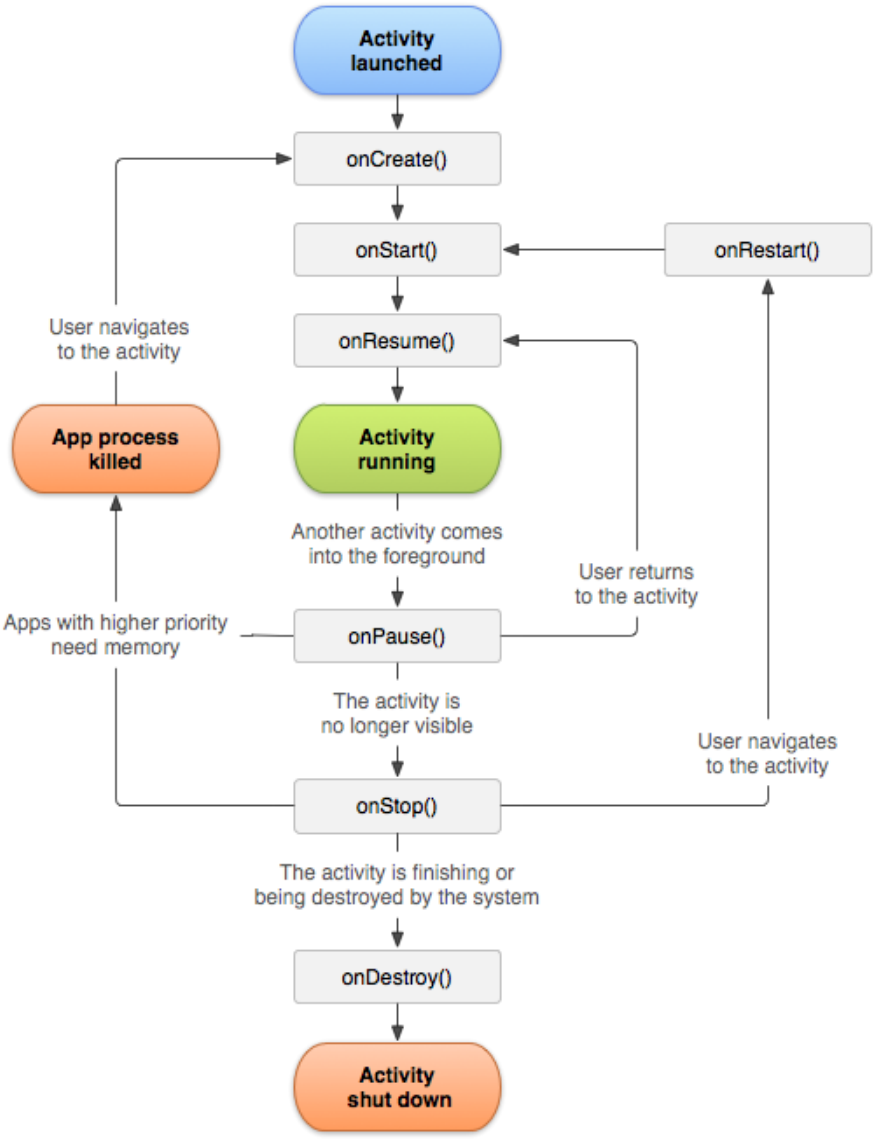

Figura 7 Ciclo de vida de uma Activity – Fonte (ANDROID, 2012).

Um exemplo de um caso para se implementar o controle dos métodos do ciclo de vida de uma *Activity* é quando o *smartphone* entra em modo de espera ou recebe uma ligação telefônica. Durante a utilização de um aplicativo *Android* usando um *smartphone* você esta sujeito a receber ligações telefônicas, o que acontece com o aplicativo que esta sendo utilizado se a ligação for atendida? O aplicativo fecha? O aplicativo executa em plano de fundo?

Os métodos do ciclo de vida servem para tratar desses tipos de problemas, é possível implementar cada um dos métodos abaixo localizados no Quadro 1:

Método	Descrição
onCreate(bundle)	Método obrigatório que é chamado uma única vez(método semelhante ao método main( ) do Java.
onStart( )	Método chamado quando uma *Activity* está ficando visível ao usuário. Pode ser chamado depois dos métodos onCreate( ) ou onRestart( ), dependendo do estado da aplicação.
onRestart( )	Método chamado quando uma *Activity* foi parada temporariamente e está sendo iniciada outra vez. Esse método chama o método onStart( ) de forma automática.
onResume( )	È chamado quando a *Activity* esta no topo da pilha, isto é, visível para o usuário. Desta forma esta a *Activity* estará pronta para receber a interação do usuário.
onPause( )	Se algum evento ocorrer, como o smartphone entrar em modo de espera/dormir para economizar energia, a *Activity* do topo da pilha que esta executando pode ser temporariamente interrompida. Para isso o método onPause( ) é chamado para salvar o estado da aplicação , para que posteriormente quando a *Activity* voltar a executar, tudo possa ser recuperado, se necessário no método onResume( ).
onStop( )	O método é chamado quando a *Activity* esta sendo encerrada, e não esta mais visível ao usuário, o que pode ocorrer quando outra Activity é iniciada.
onDestroy ()	Método que literalmente encerrar a execução de uma *Activity*. Esse método pode ser chamado automaticamente pelo sistema operacional para liberar recursos.

Quadro 1 Métodos do ciclo de vida de uma Activity
Fonte: adaptado de LECHETA (2009)

2.6.9 Banco de Dados no Android

Como visto na arquitetura da plataforma *Android* o banco de dados suportado pelo *Android* é o SQLite localizado na camada bibliotecas dentro da arquitetura *Android*. O SQLite é um banco de dados leve e poderoso integrado que possibilita o uso normal de banco de dados por um aplicativo.

> Embora o armazenamento em banco de dados seja a forma mais comum de persistência, O *Android* também permite que arquivos sejam salvos facilmente, além de ter um sistema simples de persistência de chave e valor chamado de preferências (LECHETA, 2009).

Existem varias formas de se criar um banco de dados no *Android*, uma delas é utilizar a API do *Android* para o SQLite., com a API é possível criar o banco, as tabelas e executar script SQL utilizando linhas de código na implementação do próprio aplicativo.

Outra forma de criar um banco de dados é por meio de um cliente do SQLite. Para o desenvolvimento desse trabalho a ferramenta escolhida para a criação da base de dados foi o SQLite *Expert Personal*. O SQLite *Expert Personal* é um cliente gratuito para o Banco de dados SQLite, que pode ser baixado no seguinte endereço http://www.sqlite.org). È possível visualizar a ferramenta na Figura 8.

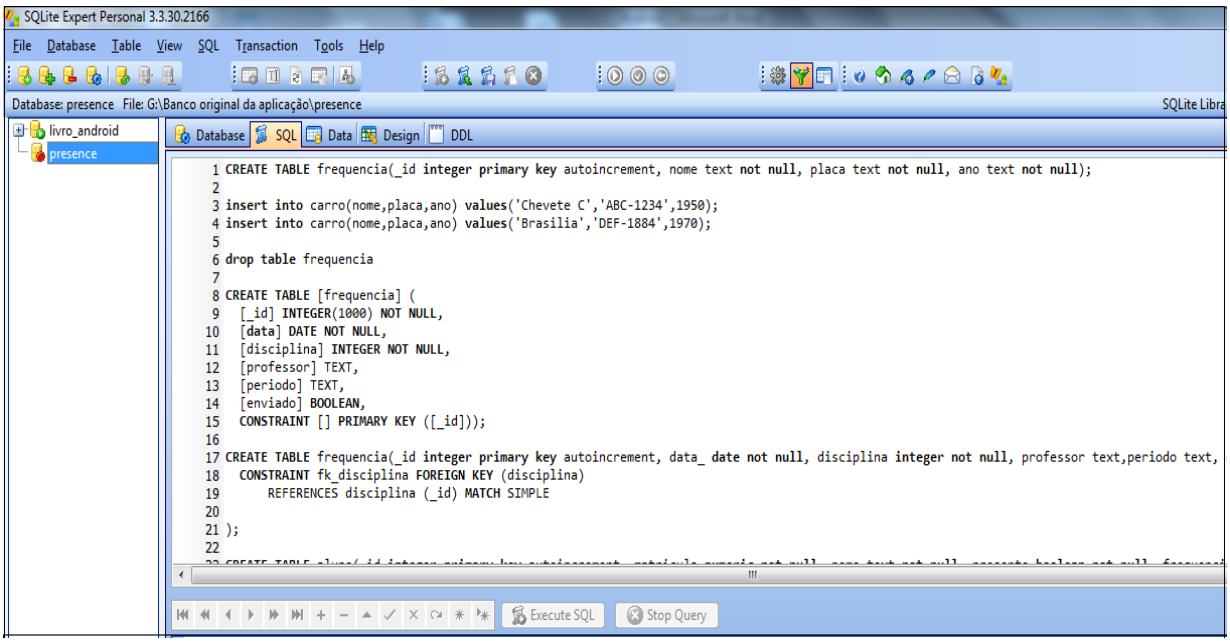

Figura 8 SQLite Expert Personal Fonte: Autor

2.6.10 Google Play Portal de Aplicativos para Android

O antigo portal *Android Market* passou a se chamar Google *Play* em 2012. O nome do portal foi alterado para um nome mais abrangente, pois além de aplicativos para *Android* pode se encontrar aplicativos para smart Tvs, navegadores entre outros.(ANDROID, 2012).

O portal mais conhecido para distribuição de aplicações *Android* e divulgação da plataforma é o Google *Play*. Com o objetivo de criar um lugar em comum entre usuários do *Android* o Google criou seu portal de aplicações.

Para poder entrar no Google *Play* basta fazer um cadastro e baixar ou enviar aplicativos. Também é possível fazer comentários e avaliar aplicativos de outros usuários, contribuindo assim para a criação de *ranking* com os melhores aplicativos. (LECHETA, 2010).

A centralização de aplicativo para a plataforma *Android* contribui muito não só para os desenvolvedores mas também para o usuário que procura por bons aplicativos e avaliações de outros usuários.

O Google *Play* também garante segurança para os usuários do portal, antes das aplicações ficarem disponíveis para *download* elas são analisadas cuidadosamente com o objetivo de barrar aplicações com código malicioso e evitar futuros danos ao usuário que faz o *download*.

O desenvolvedor precisa pagar uma taxa de US$ 25 para publicar uma aplicação e concordar com os termos de uso. Após esta etapa o aplicativo pode ser baixado por qualquer usuário do Google *Play*. (LECHETA, 2010).

Um aplicativo pode ser gratuito ou pago, 70% dos lucros dos aplicativos vendidos são revertidos para o desenvolvedor. A distribuição de aplicativos por meio de um portal de aplicações tal como Google Play contribuiu muito no crescimento do mercado móvel. (LECHETA, 2010). Pagina inicia do Google Play na Figura 9.

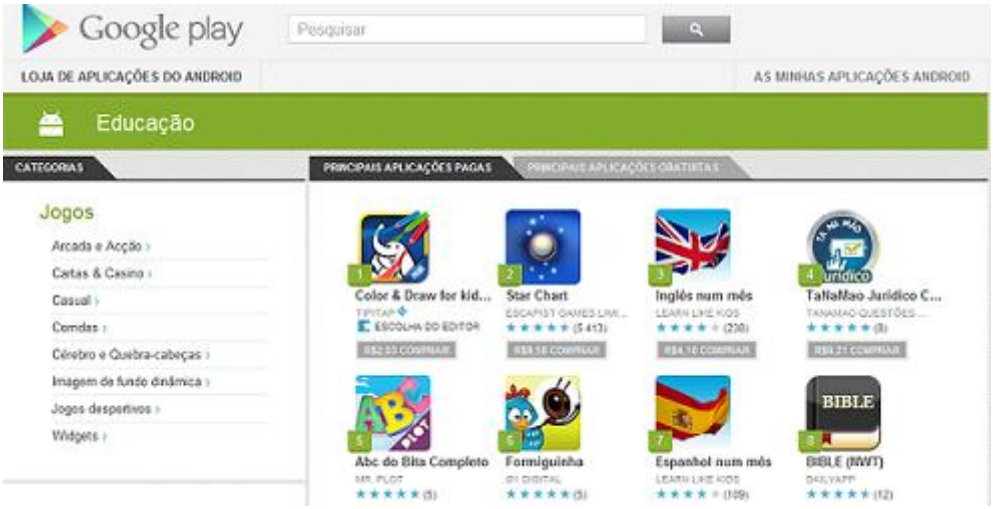

Figura 9 Portal de aplicativos Google Play Fonte: Autor

3 PROPOSTA DE UM APLICATIVO ANDROID PARA CONTROLE DE FREQUÊNCIA

3.1 PROBLEMATIZAÇÃO

Durante o processo de investigação que deu origem a esta proposta foram observadas algumas dificuldade encontradas no gerenciamento manual e tradicional da frequência de alunos em um ambiente acadêmico:

- A grande parte dos professores ainda tem dificuldade em gerenciar a frequência dos alunos utilizando documentos impressos e métodos tradicionais;
- Após o controle de frequência ter sido realizado em sala de aula, utilizando documentos impressos, as informações devem ser atualizadas no sistema acadêmico. Por esta razão o esforço de trabalho do professor acaba sendo dobrado durante esse processo;
- Ao inserir, no sistema de gerenciamento acadêmico, os dados preenchidos nesses documentos impressos as informações podem perder a consistência devido ao atraso na inserção, erros e falhas humanas (digitação, esquecimento, perda das informações impressas), entre outros.

A partir dessas observações esta proposta tem como foco responder duas questões chave relacionadas ao gerenciamento de controle de frequência acadêmica:

- Questão 1: é possível desenvolver um sistema de baixo custo para o gerenciamento de frequência que possa ser utilizado em dispositivos móveis?
- Questão 2: quais as vantagens de se utilizar um sistema com estas características em um ambiente acadêmico?

Os métodos e os materiais utilizados para o desenvolvimento da proposta são descritos nas seções a seguir.

3.2 MATERIAIS E MÉTODOS

O presente estudo é caracterizado como uma pesquisa exploratória, pois envolve um levantamento bibliográfico, análise do ambiente com o objetivo de fornecer maior familiaridade com o problema com o intuito de torná-lo mais explicito.

Uma pesquisa exploratória tem a finalidade de fornecer um modelo para solução de um determinado problema e verificar seu impacto no ambiente proposto (CLEMENTE, 2007).

Para conduzir o processo de desenvolvimento será utilizado o processo de engenharia de software UP – *Unified Process* ou processo unificado. Na Figura 10 abaixo é possível visualizar as fases que compõem o Processo Unificado:

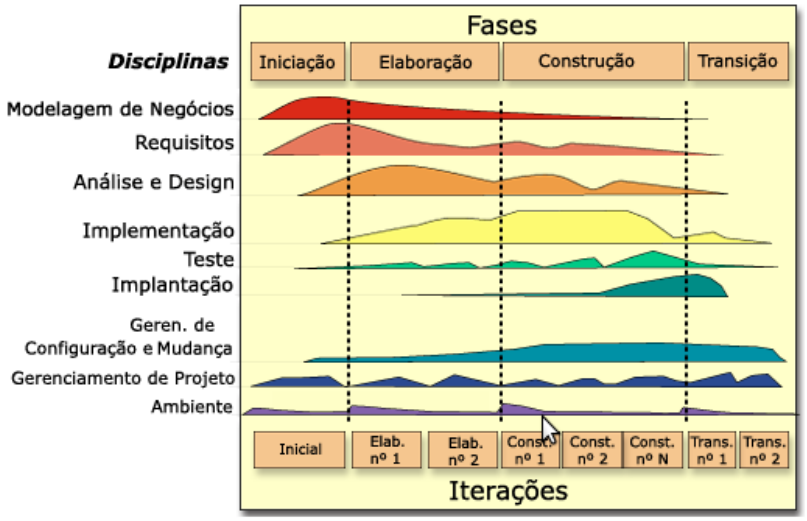

Figura 10 Arquitetura geral do UP Fonte: Autor

Dentre as fases do Processo Unificado serão utilizadas as seguintes ferramentas:

- ASTAH para elaboração do diagrama de caso de uso, diagrama de atividades e de arquitetura.
- Eclipse IDE – Para implementação do projeto *Android* junto ao *plugin* ADT (*Android Developers Tools*).
- *Android* SDK (*System Development Kit*) – para uso das ferramentas disponíveis.

- O sistema operacional utilizado será Ubuntu 10.04 Lucid Lynx
- O dispositivo para testes será uma aparelho Motorola *Defy* com *Android* 2.2, 800Ghz ,512mb de *ram*.

O ambiente de desenvolvimento Eclipse IDE é fortemente recomendado por possuir componentes compatíveis que se integram ao *Android* bem como o Sistema operacional Ubuntu que por ser uma distribuição do sistema operacional Linux faz toda abstração do hardware. Sendo assim não é necessário a instalação de plug-ins para conectar um *smartphone* ao computador para eventuais testes da aplicação como ocorre no sistema operacional Windows.

3.3 SISTEMA PROPOSTO

O desenvolvimento deste trabalho tem por objetivo proporcionar ao professor uma forma alternativa e flexível de gerenciamento de frequência. A melhor maneira de realizar isso é a implementação de uma aplicativo *Android* gerenciador de frequência.

Para isso será utilizada a biblioteca Java simple-xml.2.5.2.jar, um banco de dados SQLite, ferramentas do SDK *Android* e o Eclipse IDE como ambiente de desenvolvimento.

3.3.1 Usuário e Funções

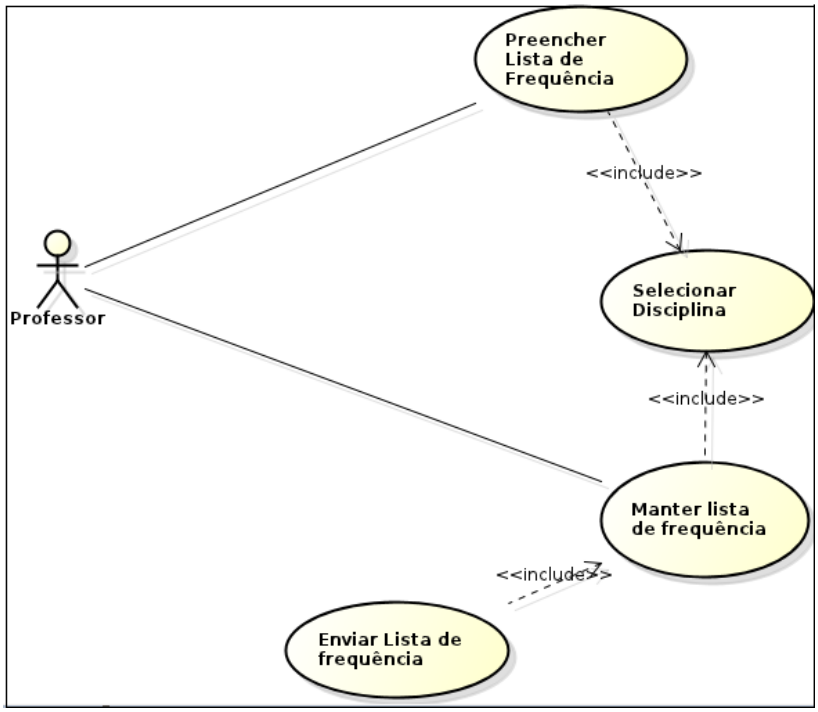

Figura 11 Funções do aplicativo Presence

O professor é único ator que opera as funções do sistema como pode ser visto na Figura 11. Suas funções incluem Preencher lista de frequência, Escolher Disciplina, Manter as listas de frequência e Enviar lista de frequência.

Basicamente o sistema consiste em oferecer ao usuário uma lista de presença virtual onde os dados referentes as disciplinas e os alunos são carregados de um arquivo xml localizado no cartão de memória do dispositivo. Esse arquivo xml precisa ser gerado para cada disciplina, para que o aplicativo consiga fazer a leitura.

O usuário consegue registrar uma aula e gerar um registro com os alunos presentes juntamente com os dados da disciplina como mostrado nos quadros 2 e 3:

Dia:	22/06/2012
Disciplina:	Empreendedorismo
Código da disciplina:	102
Professor:	José Silva

Alunos presentes:	Felipe Oliveira João da Luz Pedro Martins Marcos de Nobrega Luiza Maria a Silva

Quadro 2 Modelo de dados gerados pelo *Presence*

Dia:	01/06/2012
Disciplina:	Projeto Final II
Código da disciplina:	75
Professor:	Mario Ferreira
Alunos presentes:	Ruhan Carolino Paulo Tobias Ronaldo de Oliveira Amanda Silvestre

Quadro 3 Modelo de dados gerados pelo *Presence*

Tais dados são gravados no banco de dados do aplicativo.

Depois dos dados gerados é possível envia-los para uma conta de E-mail por meio de uma integração do aplicativo *Presence* com uma aplicação nativa.

A figura 12 ilustra arquitetura do aplicativo e seu ambiente execução.

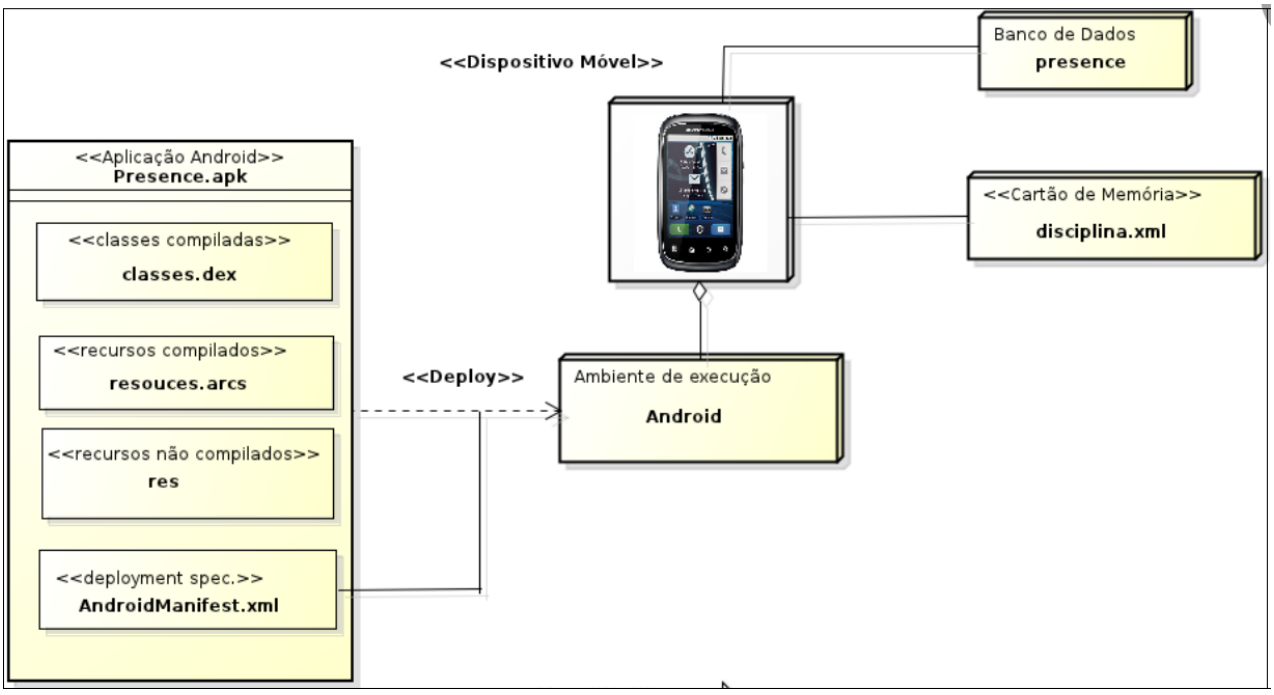

Figura 12Arquitetura do sistema

Como visto na figura 12 o aplicativo inclui um banco de dados, necessita de um cartão de memória no dispositivos para armazenar os arquivos disciplina.xml que representam cada disciplina ministrada pelo usuário contendo dados dos alunos e da disciplina. O aplicativo *Presence* também dispõem de recursos compilados da biblioteca simple-xml-2.5.2 bem como recursos não compilados, que serão por exemplo, imagens para ícone e background do aplicativo.

A arquitetura do aplicativo e do caso de uso apresentado, bem como a criação do arquivo disciplina.xml serão detalhados na seção seguinte.

4 DESENVOLVIMENTO DO SISTEMA PRESENCE

Nesse capitulo apresenta-se um descrição do sistema *Presence*, por meio da demonstração das modelagens do sistema (diagrama de casos de uso, descrição dos casos de uso, diagrama de atividades, arquitetura), o levantamento de requisitos e implementação.

4.1 ANÁLISE DO SISTEMA

4.1.1 Descrição

O produto executável gerado no desenvolvimento é um aplicativo para smartphones que possuam o sistema operacional *Android*. O aplicativo desenvolvido cujo nome é *Presence* faz o controle da frequência de todos os alunos matriculados nas disciplinas ao qual o professor ministra.

O aplicativo deve, basicamente, armazenar o registro de todas as aulas ministradas pelo professor e permitir que o professor preencha uma lista virtual de presença para cada aula. Para registrar uma aula é necessário informar uma data e preencher a lista de presença virtual.

Para utilizar o sistema é necessário antes carregar alguns arquivos XMLs no cartão de memória do dispositivo. Esses arquivos são referentes as disciplinas ministradas e serão melhor explicados no tópico implementação dos sistema.

Na tela inicial da aplicação por meio do botão “menu” é possível acessar configurações do aplicativo, um guia de ajuda e informações estarão disponíveis para consulta.

O aplicativo foi projetado para ser executado a partir da plataforma *Android* 2.2 até *Android* 2.x.

4.1.2 Declaração do Problema

O problema é	Organizar as informações relacionadas ao controle de frequência. Armazenamento de compartilhamento das informações
Afeta	Professores e sistema acadêmico
Cujo impacto é	Falta de consistência dos dados, atraso para alimentar o sistema acadêmico.
Uma boa solução seria	Implementação de um sistema que gerencie o controle de frequência
Para	Instituições de ensino
Quem	Professores

4.1.3 Levantamento de requisitos

Para a realização desta tarefa foi utilizada a técnica de entrevista. Abaixo são mostrados os requisitos encontrados para a construção do aplicativo *Presence*.

Requisitos

- Manter lista de frequência
- Enviar lista para E-mail
- Escanear código
- Gerar Lista de frequência
- Registrar aula

Ator

- Professor

4.1.4 Necessidades e Funcionalidades

Necessidade	Prioridade	Lançamento Planejamento
Manter lista de frequência	Alta	Meio do Projeto
Enviar Lista de Frequência	Alta	Final do projeto
Escanear código disciplina	Média	Inicio do Projeto
Registrar aula	Alta	Inicio do Projeto

4.1.5 Relação dos casos de uso

Especificação de casos de uso abaixo.

Caso de Uso	Ator	Descrição
Preencher Lista de Frequência	Professor	Seleciona os alunos presentes em uma lista e registra a aula escolhendo uma data.
Selecionar Disciplina	Professor	Escolhe uma disciplina para ser carregada no sistema.
Manter Lista de Frequência	Professor	Exclui, altera data de um registro.
Enviar Lista de Frequência	Professor	Envia um registro para um E-mail.

Descrição dos casos de uso

Caso de uso: Selecionar Disciplina

Ator: Professor
Descrição: Seleção de uma disciplina para identificar os alunos matriculados
Fluxo Básico: 1. O usuário clica no botão disciplinas. 2. O sistema exibe as disciplinas disponíveis no cartão de memória (arquivos.xml).

3. O usuário seleciona a disciplina desejada. 4. O sistema carrega os dados da disciplina selecionada.
Fluxos Alternativos: 2 a.1. O sistema não encontra o arquivo.xml referente a disciplina no cartão de memória. 2 a.2. O sistema retorna uma mensagem.

Caso de uso: Preencher lista de frequência

Ator: Professor
Descrição: Registro da aula preenchendo a lista de frequência.
Requisito: Execução do caso de uso Selecionar Disciplina
Fluxo Básico: 1. O usuário clica no botão listar 2. O sistema exibe informações sobre a disciplina e alista de alunos matriculados. 3. O usuário preenche a lista de frequência clicando sobre o nome dos alunos presentes. 4. O usuário clica no botão salvar 5. O sistema retorna uma mensagem: “salvando...”.
Fluxos Alternativos: 4.a.1. O usuário clica no botão alterar data. 4.a.2. O sistema retorna um *Popup* para o usuário entrar com a nova data. 4.a.3. O usuário entra com a data. 4.a 4. O sistema retorna uma mensagem. 4.b.1. O usuário clica no botão visualizar. 4.b.1 O sistema retorna uma mensagem com o nome dos alunos selecionados no passo 3.

Caso de uso: Manter Lista de Frequência

Ator: Professor
Descrição: Gerenciamento dos registros das aulas /listas de frequência
Requisito: Execução do caso de uso Selecionar Disciplina
Fluxo Básico: 1. O usuário clica no botão frequências salvas. 2. O sistema exibe os registros das aulas referentes a disciplina selecionada. 3. O usuário clica sobre um registro. 4. O sistema apresenta uma lista de opções (visualizar, excluir, alterar data). 5. O usuário clica em visualizar. 6. O sistema exibe o nome dos alunos registrados com presença.
Fluxos Alternativos: 5.a.1. O usuário clica em alterar data. 5a.2. O sistema exibe um *PopUp* solicitando que o usuário entre com a nova data. 5.a.3 O usuário entra com a nova data. 5.a.4. O sistema retorna uma mensagem. 5.b.1 O usuário clica na opção Excluir. 5.b.2 O sistema retorna uma mensagem “excluindo...”. 6.c.1 O usuário clica na opção Enviar por E-mail. 7.c .2 O caso de uso Enviar Lista de Frequência é invocado.

Caso de uso: Enviar Lista de Frequência

Ator: Professor
Descrição: Envio de um registro contendo uma lista de frequência para um E-mail.
Requisito: Execução do caso de uso Manter lista de Frequência
Fluxo Básico: 1. O sistema exibe uma lista de opções 2. O usuário clica na opção enviar por E-mail. 3. O sistema envia uma mensagem para o SO invocando uma aplicação de envio de E-mail. 4. O usuário clica no botão enviar. 5. O sistema retorna uma mensagem.
Fluxos Alternativos: 2.a.1. O sistema não encontra uma aplicação de E-mail. 2.a.2. O sistema exibe uma mensagem.

4.2 MODELAGEM DO SISTEMA

Neste tópico será apresentado os modelos que foram gerados pelos requisitos para a implementação do sistema.

4.2.1 Diagrama de caso de uso

O levantamento dos requisitos geraram o diagrama de casos de uso, representado pela Figura 13 que já foram descritos no capitulo anterior.

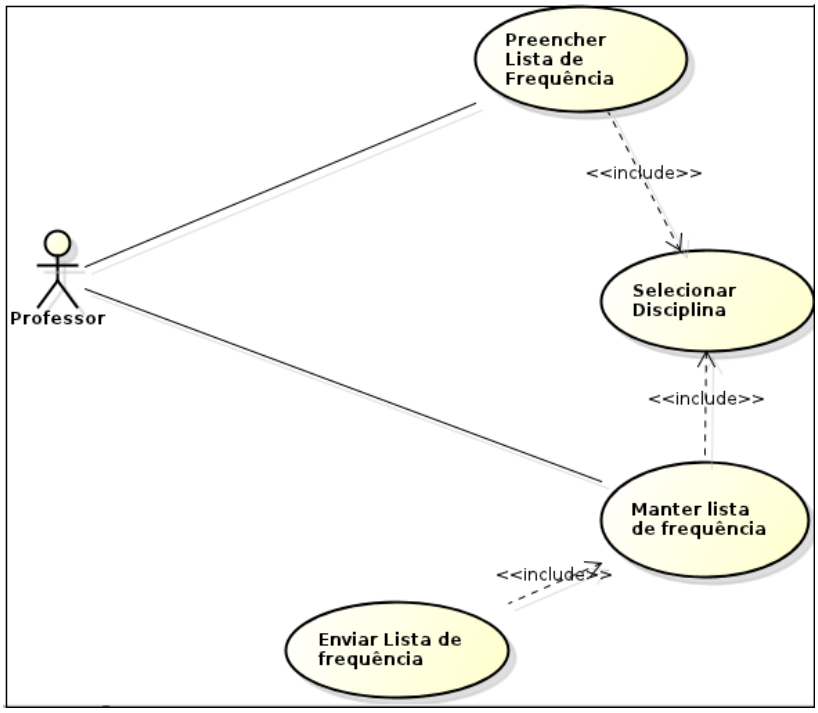

Figura 13 Diagrama de caso de uso do Presence

O diagrama de caso de uso foi modelado utilizando a ferramenta – Astah - Professional. Disponível para *download* no seguinte endereço http://astah.net/download.

O diagrama demostra de forma visual as funções do aplicativo *Presence*.

Através da figura é possível perceber as dependências entre os casos de uso.

Para a execução do caso de uso Preencher lista de frequência é necessário iniciar o caso de uso Selecionar Disciplina, ou seja, antes do usuário preencher uma lista e registrar uma aula é preciso indicar qual disciplina se deseja obter os dados.

O caso de uso disciplina é responsável por carregar os dados contidos no arquivo disciplina.xml do cartão de memória para o aplicativo.

O caso de uso Manter Disciplina também depende do caso de uso Selecionar Disciplina pois é necessário informar ao *Presence* qual disciplina se deseja gerenciar.

Por fim o caso de uso Enviar Lista de frequência depende do caso de uso Manter Lista de frequência, pois a única forma de visualizar as listas de frequência salvas é por meio do caso de uso Manter lista de frequências.

4.2.2 Diagrama de Entidade Relacionamento

O Diagrama de Entidades Relacionamento (DER) descreve o modelo de dados do sistema, expressa graficamente a estrutura lógica do banco de dados. Abaixo o DER do sistema *Presence.*

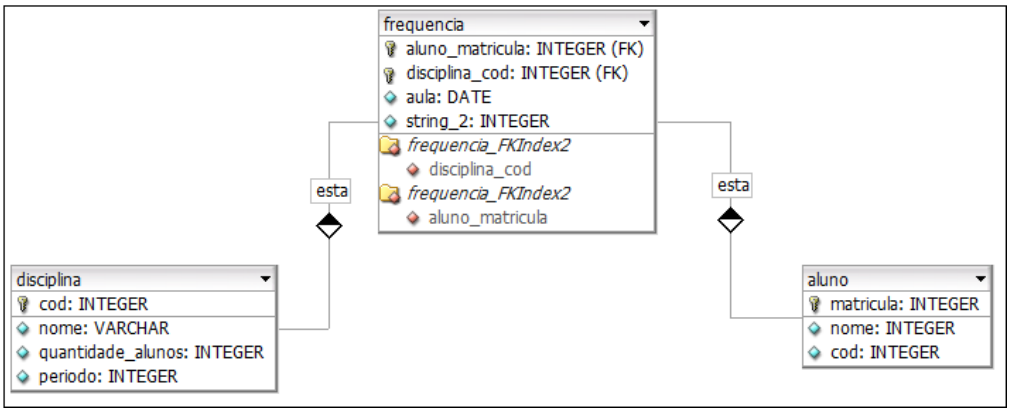

Figura 14 Diagrama de Entidade Relacionamento (DER)

O diagrama visto na figura 14 foi gerado utilizando uma ferramenta *Open Source* – DBDesigner para designer visual de banco de dados, disponível para *download* em: http://www.fabforce.net/dbdesigner4/.

O banco de dados do *Presence* é formado por apenas 3 tabelas que armazenam os dados que serão lidos dos arquivos XMLs referentes as disciplinas que devem ficar no cartão de memória. O tópico implementação do sistema descreve como o banco foi inserido no *Presence*.

4.2.3 Diagrama de atividades

O diagrama de atividades apresenta todas as possíveis atividades do *Presence* organizadas de forma lógica. O objetivo desse diagrama é mostrar o fluxo de uma atividade para outra como pode ser visto na figura 15.

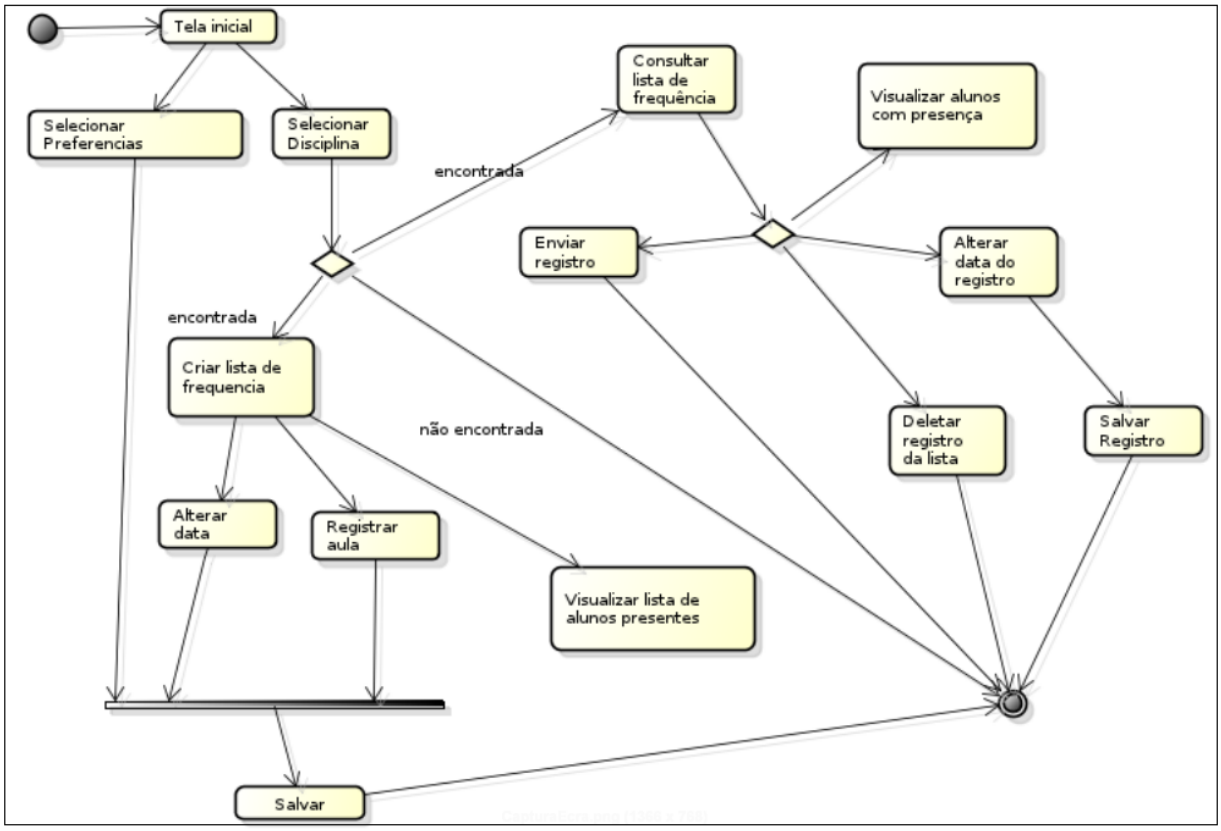

Figura 15 Diagrama de atividades

Seguindo a linha de desenvolvimento do Processo Unificado a base da implementação do aplicativo *Presence* foram os casos de uso.

Outros artefatos gerados durante o desenvolvimento como a Lista de Risco, o Plano de Iteração e o Plano de Desenvolvimento estão disponíveis nos APÊNDICES A, B e C, respectivamente.

4.2.4 Diagrama de arquitetura

A finalidade desse diagrama é apresentar uma visão de alto nível de todos os componentes envolvidos com o ambiente de execução e os recursos do aplicativo *Presence*, na figura 16 abaixo.

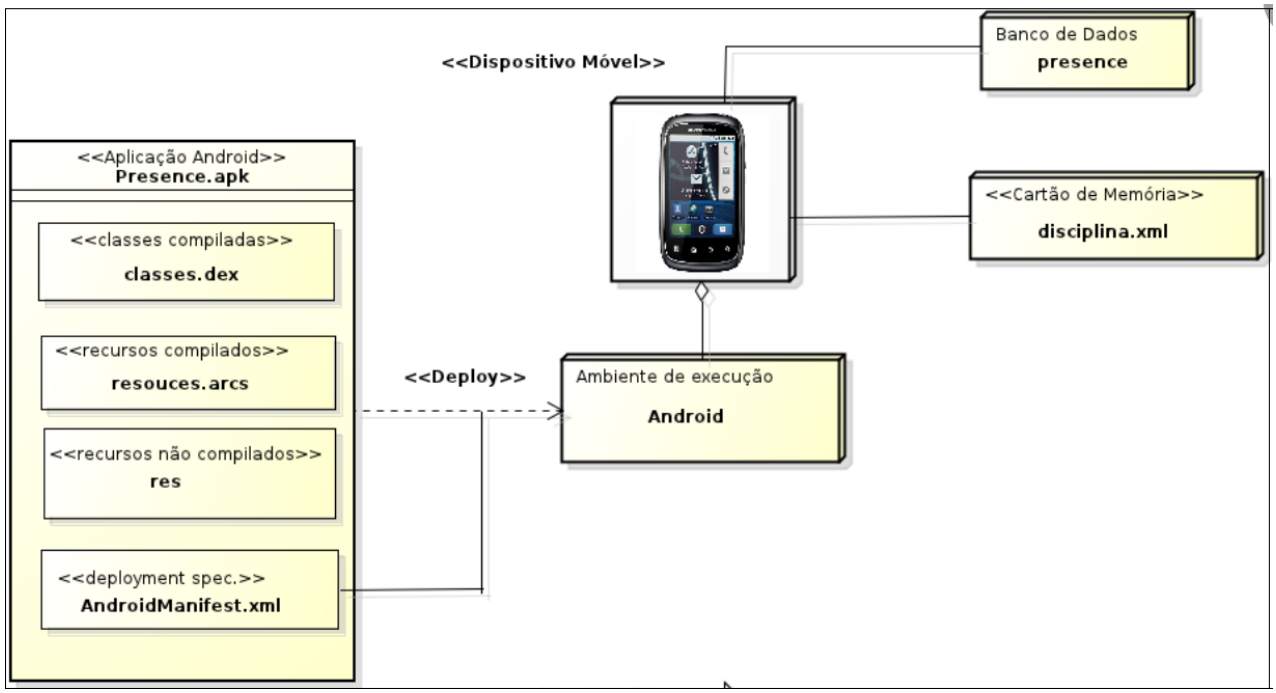

Figura 16 Arquitetura do sistema

Como visto na figura o sistema esta organizado da seguinte forma o aplicativo é formado por um conjunto de classes e recursos. As classes são previamente compiladas e dessa compilação são geradas as classes.dex, para cada classe compilada uma classe.dex é gerada.

As classes devem ser compiladas para a extensão .dex (Dalvick Executable Format) para que a maquina virtual Dalvick consiga fazer uso delas. O recurso compilado será a biblioteca simple-xml-2.5.2.jar. Para que a máquina virtual consiga interpretar esse recurso a biblioteca também é compilada e assim as classes.dex são geradas.

Os recursos não compilados serão as imagens.png utilizadas como background e ícones da aplicação. Na especificação do aplicativo esta localizado o arquivo AndroidManifest, esse arquivo contem todas as configurações da aplicação, tais como o mapeamento das *Activitys* utilizadas, versão do aplicativo entre outras informações.

O ambiente de execução do aplicativo será a plataforma *Android* 2.2. Um banco de dados deve ser implementado para que o aplicativo possa utiliza-lo. No cartão de memória ficam os arquivos disciplina.xml que serão lidos pelo aplicativo.

No tópico a seguir serão descritos os detalhes do desenvolvimento do sistema, bem como a implementação dos casos de uso e do arquivo disciplina.xml.

4.2.5 Implementação do Sistema

Nesse tópico serão demonstradas as implementações mais relevantes do aplicativo *Presence*, tais como, a implementação das classes responsáveis por gerar o arquivo disciplina.xml, a inserção do banco de dados no aplicativo *Presence*, integração do aplicativo com aplicação nativa do *Android*, bem como os casos de uso envolvidos com as implementações.

Arquivo Disciplina.XML

O arquivo com a lista de alunos de cada disciplina em XML é a interface de entrada do sistema. Sendo assim, em um primeiro momento, a ideia foi gerar um arquivo XML que pudesse ser lido pelo aplicativo *Presence*. O arquivo XML gerado fica no cartão de memória do emulador ou dispositivo ao qual o aplicativo foi instalado.

Esse arquivo deve conter seguintes dados para cada disciplina, nome do professor, período, quantidade de alunos e a lista dos alunos matriculados na disciplina (Figura 17).

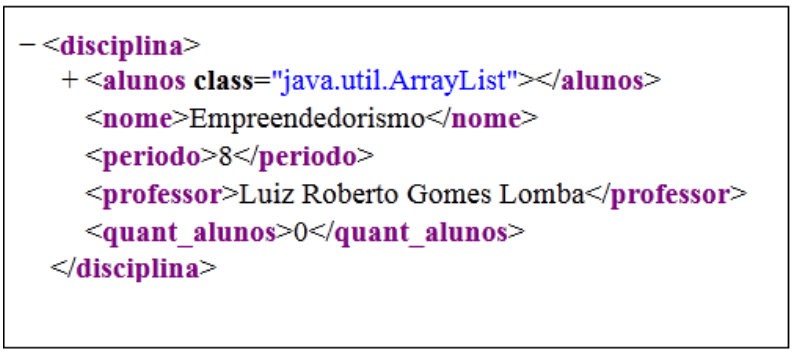

Figura 17 Estrutura do arquivo XML

Na Figura 17 a tag <disciplina> contem as seguintes tags: <alunos>, <nome>, <período>, <professor> e <quantidade_alunos>.

A tag <alunos> representa uma lista de alunos matriculados na disciplina. Na Figura 18 é possível observar o atributo class dentro da tag alunos, esse atributo faz referência a classe ArrayList localizada no pacote “java.util”.

Um objeto da classe ArrayList representa uma lista de objetos em java, isto significa que a tag <alunos> após ser lida pelo aplicativo será transformada em uma lista de objetos da classe Aluno.

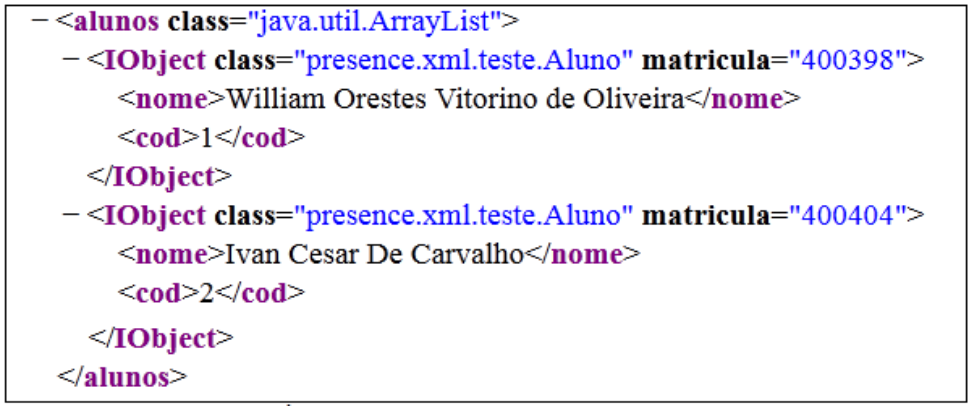

Figura 18 Tag Alunos

A tag<IObject> possui os atributos “matricula” como pode se observado na Figura 18 e representa a classe Aluno, em seguida serão dados mais detalhes sobre a estrutura das classes Aluno e Disciplina. O atributo “*class*” da tag <IObject> faz referência a uma classe Interface implementada pela classe Aluno, que esta localizada no pacote “presence.xml”. Já o atributo “matricula“ representa um código numérico, para satisfazer a implementação adotada, o código é representado pelo mesmo numero da matricula do aluno.

Por fim, a tag <cod> também representa um código numérico mas para fins de integração com um banco de dados de um sistema acadêmico ou outros tipos de sistemas. Essa tag é reservada para ser configurada conforme a necessidade do sistema, esse código não é utilizado em momento algum pelo aplicativo *Presence*.

A tag<cod> poderia ser utilizada, por exemplo, se um sistema acadêmico ou um sistema que faça a interface com o banco de dados também desejasse fazer a leitura desse arquivo XML, a tag poderia ser um código utilizado pelo sistema para controle interno. Analogamente, um sistema poderia se criado apenas para gerar esse arquivo XML para cada disciplina e a tag serviria somente para que o sistema pudesse manter um controle sobre os arquivos XMLs gerados.

Antes de apresentar os métodos que geram esse arquivo XML bem como a configuração necessária para isso vamos observar a estrutura das classes Disciplina Aluno e IObject.

Como explicado na Seção 2.2 do Capítulo 2, todos os aplicativos Android são implementados na linguagem Java. As seções a seguir descrevem os detalhes de implementação mais relevantes para entendimento técnico dos leitores.

Classe Aluno

A Figura 19, a seguir, mostra detalhes de implementação da Classe Aluno. É possível observar na linhas 1 e 2 as importações das classes Attribute e Element do pacote “org.simpleframework.xml” que será explicado posteriormente.

```
1  package presence.xml.teste;
2  import org.simpleframework.xml.Attribute;
3  import org.simpleframework.xml.Element;
4
5  public class Aluno implements IObject{
6      @Element //Notação Elemento
7      public String nome;
8      @Element
9      private  int cod;
10
11     private final int matricula;
12     //Construtor
13     public Aluno(@Attribute(name = "matricula") int id){//Notação Atributo
14         super();
15         this.matricula = id;
16     }
17
18     @Attribute(name = "matricula")
19     @Override
20     public int getObjectId() {
21         return matricula;
22     }
23
24     public String getNome() {
27
28     public void setNome(String nome) {
31
32     public int getCod() {
35
36     public void setCod(int cod) {
```

Figura 19 Estrutura da classe Aluno.

Na linha 5 da Figura 19 é possível perceber que a classe Aluno implementa a interface IObject, a implementação dessa interface serve apenas para facilitar a implementação e organização dos métodos de leitura que transformam as tags do arquivo XML em objetos no Java.

Da linha 6 a linha 9, é feito o mapeamento dos atributos “nome” e “cod” utilizando a notação @Element. A notação @Element faz com que os atributos “nome” e “cod” sejam convertidos em tags como se pode observar na Figura 20.

```
<nome>William Orestes Vitorino de Oliveira</nome>
<cod>1</cod>
```

Figura 20 Tags nome e cod no arquivo XML

Já na linha 13 da Figura 21, no construtor da classe Aluno surge a notação @Attribute, a função dessa notação é fazer com que o atributo matricula localizado na linha 11 faça referência ao atributo matricula da tag <IObject>.

```
10
11      private final int matricula;
12      //Construtor
13      public Aluno(@Attribute(name = "matricula") int id){//Notação Atributo
14          super();
15          this.matricula = id;
16      }
```

Figura 21 Construtor da classe Aluno

Como a classe Aluno implementa a interface IObject o método getObject Id() deve ser implementado Figura 22.

```
18      @Attribute(name = "matricula")
19      @Override
20      public int getObjectId() {
21          return matricula;
22      }
23
24      public String getNome() {
27
```

Figura 22 Método getObjectID()

Classe IObject

A classe IObject é uma Interface e possui um único método, getObjectId() como pode ser observado na Figura 23.

Interface em Java é como um contrato ao qual a classe que implementa tal interface deve implementar todos os métodos definidos na mesma. Uma Interface possui apenas a assinatura de seus métodos, ou seja, um método não pode ser implementado em uma interface. A assinatura de um método é composta por um modificador de acesso, *public, private, protected* ou *default*, um retorno, por exemplo void ou int, o nome do método e por fim os parâmetros recebidos pelo método. Então quando uma classe implementa uma interface ela se compromete a implementar seus métodos (LECHETA, 2010).

```
1  package presence.xml.teste;
2
3  public interface IObject {
4
5          int getObjectId();
6  }
7
```

Figura 23 Interface IObject

Classe Disciplina

A classe disciplina também foi mapeada da mesma forma que a classe Aluno, porém possui mais duas notações, importadas do pacote “org.simpleframework.xml”, como pode ser observado na linha 8 e na linha 23 da Figura 24. A notação @Root é utilizada para fazer com que a tag <disciplina> representada pela Classe Disciplina seja a tag que abrange todas as outras tags, isto é, disciplina é a tag raiz no arquivo XML.

Todas as outras tags devem ficar dentro da tag <disciplina>, esta é a função da notação @Root.

A tag @ElementList faz o mapeamento do objeto ”alunos” que é um objeto da classe ArrayList. Essa tag é utilizada para mapear listas, no caso o objeto “alunos” é uma lista de objetos que implementam a interface IObject. Basicamente o mapeamento é feito para que o arquivo XML em especifico a tag <alunos> consiga representar uma lista da mesma forma que um objeto da classe ArrayList representa.

```java
package presence.xml.teste;
import java.util.ArrayList;
import java.util.List;
import org.simpleframework.xml.Element;
import org.simpleframework.xml.ElementList;
import org.simpleframework.xml.Root;

@Root
public class Disciplina {

    @Element
    private String nome;

    @Element
    private String professor;

    @Element
    private int quant_alunos;

    @Element
    private String periodo;

    @ElementList
    public List<IObject> alunos = new ArrayList<IObject>();

    public String getPeriodo() {
        return periodo;
    }

    public void setPeriodo(String periodo) {
        this.periodo = periodo;
```

Figura 24 Classe Disciplina

Configuração necessária

Antes de apresentar os métodos que geram o arquivo XML, será demonstrada a configuração necessária para que se possa utilizar as notações @Root, @Element, @ElementList e @Attribute.

Para usar essas notações é preciso fazer a importação das classes as quais tem o mesmo nome das notações, Element, Root etc..Tais classes pertencem biblioteca simple-xml-2.5.2.jar que esta disponível para download no endereço: http://www.java2s.com/Code/Jar/s/Downloadsimplexml252jar.htm

Para importar a biblioteca simple-xml-2.5.2.jar em um projeto no Eclipse IDE os seguintes passos foram executados:

Clique com o botão direito sobre o Projeto Android -> Import -> File System -> simple-xml-2.5.2.jar

Clique com o botão direito sobre o Projeto *Android-> Properties* -> Java *Build Path -> Libraries -> Add* Jar -> simple-xml-2.5.2.jar

Após a importação da biblioteca será possível utilizar todas as classes necessárias para mapear as classes.

Classe WriteDisciplina XML

Para criar o arquivo disciplina.xml a classe WriteDisciplina foi implementada. A implementação da classe é relativamente simples contendo poucas linhas de código.

O construtor da classe WriteDisciplina é apresentado na Figura 25. Ele é responsável por gerar o arquivo disciplina.xml no cartão de memória do dispositivo. As importações são apresentadas na Figura 26.

```
12      //Construtor
13      public WriteDiciplina() {
14
15           diciplina = new Diciplina();
16           criarTurmaXML();
17
18          Serializer serial = new Persister();
19          File sdcardFile = new File("/sdcard/auditoria_e_seguranca_de_sistemas.xml");
20
21          try {
22              serial.write(diciplina, sdcardFile);
23          } catch (Exception e) {
24              e.printStackTrace();
25          }
26      }
27
```

Figura 25 Construtor da classe

```
1  package com.simpletest.test;
2
3  import java.io.File;
4  import org.simpleframework.xml.Serializer;
5  import org.simpleframework.xml.core.Persister;
6  import presence.xml.teste.Aluno;
7  import presence.xml.teste.Diciplina;
```

Figura 26 Importações

Classe ReadDisciplina XML

Essa classe é utilizada no caso de uso selecionar disciplina. A classe foi implementada com o propósito de fazer a leitura do arquivo disciplina.XML gerado pela classe Write Disciplina. O método adicionarMateria() recebe como parâmetro um objeto do tipo File contendo o endereço do arquivo XML no cartão de memória do dispositivo. O objeto “serial” da classe Serializer possui o método de leitura read() que recebe como parâmetro uma classe e um objeto do tipo File.

Após a execução da linha 66 da Figura 27 todos os dados contidos no arquivo XML são recuperados e transferidos para o objeto Disciplina disci.

Desta forma o objeto disciplina pode ser manipulado novamente pelo aplicativo *Presence* conforme a necessidade.

Após fazer a leitura do aplicativo os dados são gravados no banco de dados do aplicativo *Presence*.

```
62    public void adicionarMateria(File file) {
63
64        try {
65
66            Diciplina disci= serial.read(Diciplina.class, file);
67            //array.add(String.valueOf(disci.getNome()));
68            diciplinas.add(disci);
69
70        } catch (Exception e) {
71
72            e.printStackTrace();
73        }
74    }
```

Figura 27 Método adicionar disciplina

Integração com aplicativo de E-mail nativo

A implementação dessa função esta relacionada ao caso de uso Enviar Lista de Frequência. Para o envio dos dados uma integração com o aplicação nativa de E-mail do *Android* é realizada através de um método chamado enviarParaEmail demonstrado na figura 28.

```
182
183    public void enviarParaEmail(String email,String disciplina,String corpoEmail){
184
185        //mensagem de comunicação
186        final Intent emailIntent = new Intent( android.content.Intent.ACTION_SEND);
187        //tipo dados
188      emailIntent.setType("plain/text");
189        // add endereço de E-mail
190      emailIntent.putExtra(android.content.Intent.EXTRA_EMAIL,
191              new String[] { email});
192       //add nome da disciplina
193      emailIntent.putExtra(android.content.Intent.EXTRA_SUBJECT,
194              "PRESENCE-[ "+disciplina+" ]");
195       // add lista de frequencia e dados da disciplina
196      emailIntent.putExtra(android.content.Intent.EXTRA_TEXT,
197              corpoEmail);
198
199      startActivity(Intent.createChooser(
200              emailIntent, "Enviar E-mail..."));
201
202    }
```

Figura 28 Método Enviar para E-mail

O método recebe os parâmetros, endereço de E-mail, nome da disciplina e corpo da mensagem. O corpo da mensagem é uma *string* contendo todos os dados referentes a uma lista de frequência salva no banco de dados, isto é, data e alunos presentes em tal data e dados da disciplina.

Basicamente o aplicativo envia uma mensagem em linha de código para o *Android* que é criada na linha 186 na Figura 28 e enviada na linha 199 dizendo que deseja utilizar a aplicação nativa de E-mail.

Os parâmetros recebidos no método são repassados para o objeto “emailIntent” que representa uma mensagem a ser enviada para o *Android* nesse contexto.

Integração com aplicativo para leitura de código de barras

A ideia da implementação dessa função é de poder explorar ao máximo os recursos oferecidos pelo *Android*, tal como a facilidade de integração com outras aplicações sejam elas nativas ou não. A integração descrita a seguir é entre o *Presence* e um aplicativo não nativo chamado Barcode Escaner.

O Barcode Escaner é um aplicativo gratuito para leitura de código de barras tradicional e do tipo QRCode disponível para download no portal de aplicativos *Android*, Google Play.

Figura 29 Código de barras tradicional e código de barras QRCode

O QRCode da figura 29 ao ser escaneado leva até a pagina de donwload do BarcodeScanner.

O aplicativo funciona da seguinte forma, ele permite realizar a leitura de um código de barras por meio da câmera do dispositivo, basta apontar a câmera para um código e esperar alguns segundos. Os códigos QRCodes podem ser gerados no seguinte endereço http://qrcode.kaywa.com/.

A ideia é gerar um código para cada disciplina para que quando o código for escaneado o *Presence* entre no caso de uso Selecionar disciplina.

Abaixo na Figura 30 o método implementado para fazer a integração com o BarcodeScan:

```
233
234    public void escanearQRCode(){
235        //mensagem para o SO
236        intent = new Intent("com.google.zxing.client.android.SCAN");
237        //tipo do código a ser escaneado
238        intent.putExtra("SCAN_MODE", "QR_CODE_MODE");
239        //intent.putExtra("item_selecionado",combo.getSelectedItemId());
240
241        //envio da mensagem para o Android
242         startActivityForResult(intent, 0);
243    }
```

Figura 30 método escanearQRCode( )

O método é relativamente simples, na figura 30 na linha 236 uma mensagem é criada passando como parâmetro uma *string* que identifica o aplicativo *Barcode Scanner*, na linha 242 a mensagem é enviada para o *Android*.

Inserção do banco de dados no aplicativo *Presence*

Depois da criação do banco de dados com a Ferramenta SQL Expert Personal o arquivo gerado (banco de dados) foi inserido no projeto do aplicativo Presence utilizando a ferramenta File Explorer do Eclipse IDE, como demonstrado na Figura 31.

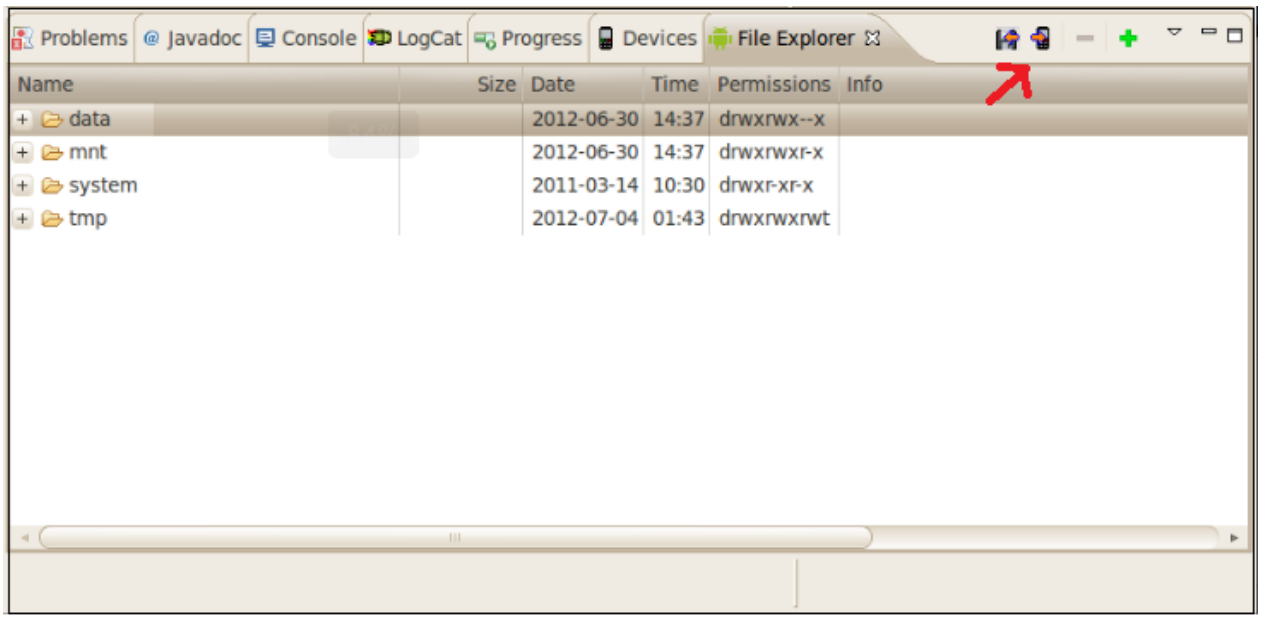

Figura 31 inserindo banco de dados no dispositivo

O banco de dados deve ser inserido dentro da pasta do aplicativo *Android* no dispositivo, no caso de *Presence* o caminho selecionado foi /data/data/com.presence/databases.

Para isso é preciso clicar no botão “inserir no dispositivo” marcado na Figura 31. O dispositivo foi conectado ao computador por meio de um cabo USB.

4.3 PRESENSE- GERENCIADOR DE FREQUÊNCIA ACADÊMICA

Tela Inicial

E possível visualizar a tela inicial do presence na Figura 32.

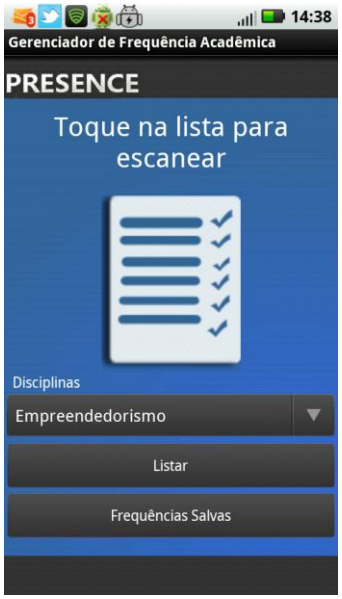

Figura 32 Tela inicial do aplicativo

O aplicativo final cujo nome é *Presence* ou em português Presença possui em sua tela inicial com um display mostrando algumas mensagens para auxiliar o usuário na utilização do aplicativo. As mensagens apresentadas são:

1- Bem vindo;
2- Toque na Lista para escanear;
3- Ou selecione a disciplina abaixo.

Essas mensagens tem como objetivo dar boas vindas ao usuário e instrui-lo rio a selecionar uma disciplina.

A mensagem 2 alerta o usuário para usar uma segunda forma de selecionar uma disciplina que é por meio do escaneamento de um código de barras clicando na imagem da lista localizada no centro da tela inicial.

As mensagens apresentadas são alternadas em um intervalo de 4s. Também é possível visualizar na tela inicial do aplicativo Figura 32 os botões Listar e Frequências Salvas.

O botão listar leva a tela Preencher lista de frequência onde o usuário pode selecionar os alunos presentes e registrar uma aula que será explicado nos tópicos posteriores.

O botão Frequências Salvas leva a tela Manter frequências onde é possível gerenciar as aulas registradas na tela Preencher lista de frequência.

4.3.1 Funcionalidade Selecionar Disciplina

Para o usuário selecionar uma disciplina é necessário clicar no botão disciplina apresentado na tela inicial. Um lista de disciplinas é exibida para cada arquivo disciplina.xml lido no cartão de memória como apresentado na figura 33.

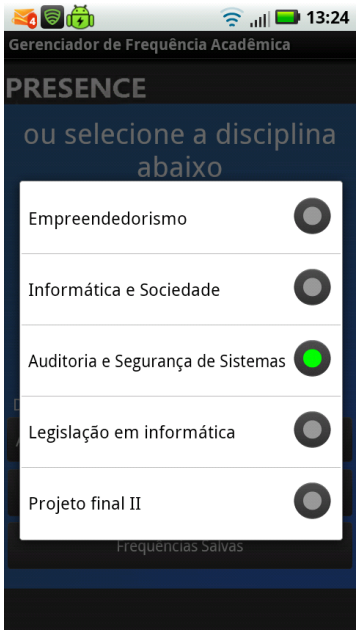

Figura 33 Funcionalidade Selecionar Disciplina

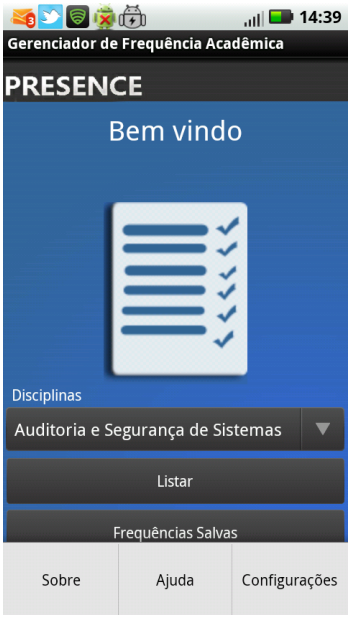

Figura 34 Opções da Tela Inicial

A opção “Sobre” apresentada no menu da figura 34 apresenta informações do desenvolvimento do aplicativo e sua descrição.

Em ajuda é possível visualizar passo a passo instruções de como utilizar o sistema figura 35. Em configurações é possível inserir um endereço de E-mail para que o aplicativo possa enviar as lista de frequências para o E-mail configurado.

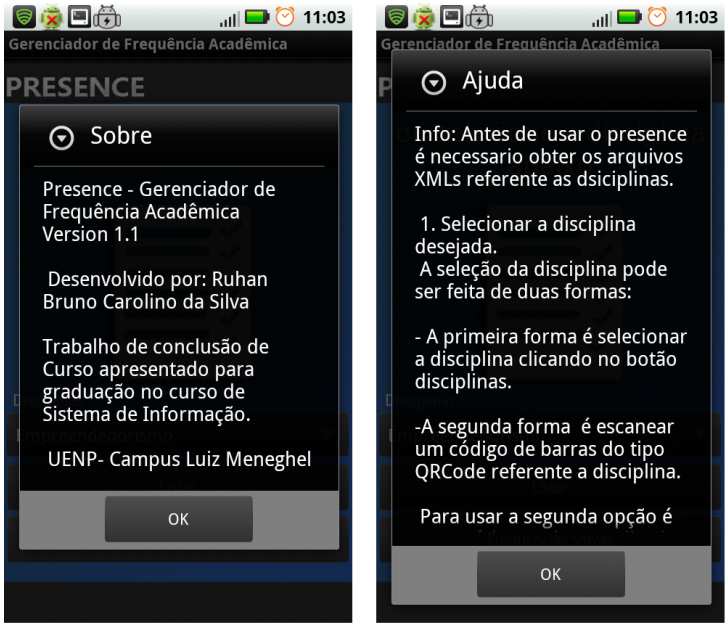

Figura 35 opções de Sobre e Ajuda

4.3.2 Funcionalidade Preencher lista de Frequência

A função dessa tela é permitir que o usuário consiga visualizar os alunos matriculados na disciplina e preencher as lista com os alunos presentes em sala de aula clicando sobre o respectivo nome de cada aluno.

Um display é exibido para o usuário com informações sobre a disciplina, tais como nome da disciplina, período, e principalmente uma data. As mensagens exibidas são alternadas em um intervalo de 3s.

Mensagens exibidas:

1. Nome da disciplina
2. Professor
3. Período
4. Data Atual

Também é possível alterar a data em que se deseja registrar a aula clicando no botão alterar data.

Outra opção disponível oferecida pelo botão visualizar é a exibição de uma mini lista com o nome dos alunos marcados com presença.

E por fim o botão salvar registra a aula salvando a lista de alunos com presença bem com a data da aula no banco de dados do Presence figura 36.

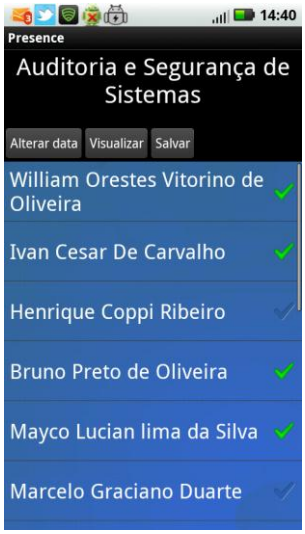

Figura 36 Preencher Lista de Frequência

4.3.3 Funcionalidade Manter Lista de frequência

Essa funcionalidade permite ao usuário realizar o controle de todos os registros referentes as listas salvas no banco de dados na funcionalidade Preencher lista de Frequência organizados por data da aula figura 37.

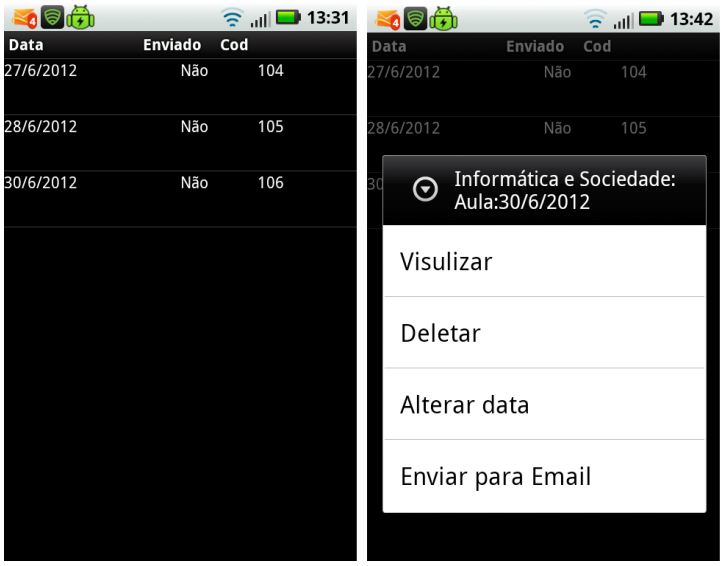

Figura 37 Funcionalidade Manter lista de frequência

Como demonstrado na figura 37 ao selecionar uma data (registro de aula) sobre a lista exibida é possível visualizar o nome dos alunos presentes na data selecionada clicando na opção Visualizar.

Cada data representa um registro de aula contendo os dados da disciplina e a lista de frequência dos alunos que é recuperada do banco de dados do *Presence*.

No tópico seguinte será apresentado a tela da ultima funcionalidade que pode ser acessada clicando na opção Enviar para E-mail como demonstrado na Figura 37.

4.3.4 Funcionalidade Enviar Lista de Frequência

Ao clicar no botão enviar para E-mail localizado na funcionalidade Manter lista de Frequência do tópico anterior as seguintes telas serão exibidas figura 38.

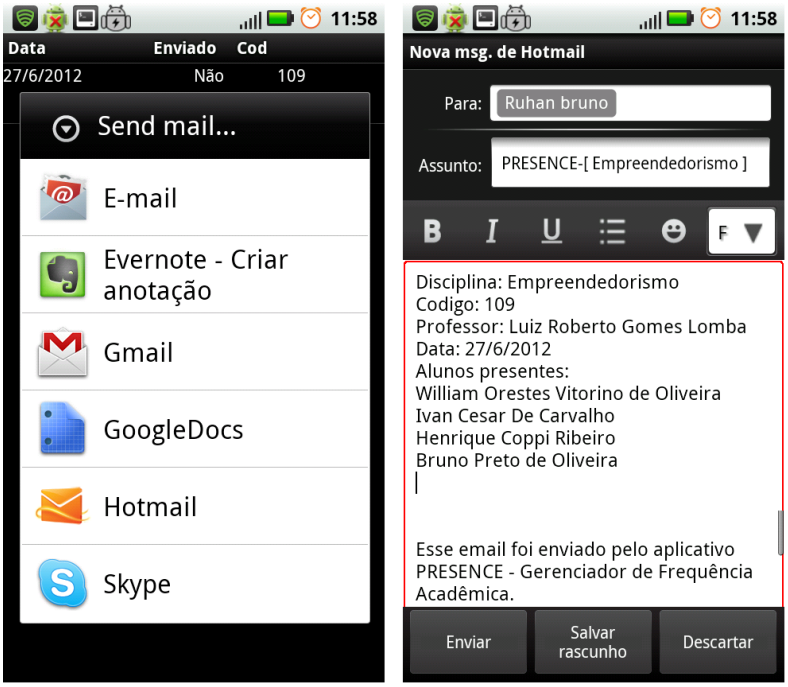

Figura 38 Funcionalidade Enviar para E-mail

Essa função demonstra a comunicação do aplicativo *Presence* com uma aplicação nativa para envio de E-mail do *Android*.

Como pode ser visto na figura ao selecionar a aplicação de E-mail, os dados referentes ao registro de frequência escolhidos são passados para o aplicativo de E-mail. No aplicativo de E-mail o destinatário pode ser alterado bem como o texto referente aos dados repassados pelo *Presence*.

5 CONSIDERAÇÕES FINAIS

O trabalho apresentado envolveu pesquisas sobre o crescimento do mercado móvel até o atual momento fazendo uma analise mais focada na perspectiva do desenvolvedor de aplicativos móveis. Foram analisadas as potencialidades da plataforma *Android* no desenvolvimento de aplicativos.

Um fator que justifica a liderança do SO *Android* é justamente a plataforma ser de código aberto e com um rico e poderoso ambiente de desenvolvimento.

Por meio da pesquisa sobre Software Livre foi possível esclarecer as vantagens e desvantagens no uso de uma plataforma livre de desenvolvimento.

Tendo em vista a pesquisa realizada foi possível a implementação do *Presence* – Gerenciador de Frequência acadêmica.

O aplicativo foi desenvolvido com foco para o meio acadêmico, entretanto por meio dos estudos realizadas pode se perceber que o foco poderia ser ampliado para outros modelos de instituições de ensino, como por exemplo, um colégio de ensino médio ou fundamental.

Dentre as vantagens de se utilizar o *Presence* no meio acadêmico, o que mais se destaca é a capacidade de organização que aplicativo oferece na manipulação dos dados relacionados ao gerenciamento de frequência, o aplicativo também possibilitou uma alternativa de compartilhamento por E-mail das informações manipuladas. O *Presence* oferece ao professor uma forma mais dinâmica de gerenciamento de frequência, diferente do que ocorria com o procedimento tradicional de gerenciamento de frequência.

Os arquivos xmls referentes as disciplinas foram implementados para futuras integrações com outros sistemas, como por exemplo um sistema acadêmico ou um servidor com um banco de dados.

A maior dificuldade do trabalho foi encontrar padrões de engenharia de software voltados para o desenvolvimento de aplicativos móveis.

Na falta de um modelo de engenharia foi adotado o Processo unificado para o desenvolvimento do aplicativo.

O trabalho demonstrou a utilização clara da modelagem que de origem aos casos de uso que serviram como base fundamental para a implementação do *Presence.*

Foi possível testar e integrar aplicativos nativos da plataforma Android complementando as funcionalidades do aplicativo.

Como continuação desse trabalho sugere-se a implementação de uma interface que faça a comunicação direta com o sistema acadêmico e também a implantação do Gerenciado de Frequência Acadêmica Presence.

REFERÊNCIAS

ANASTASIOS, A. Economides; GROUSOPOULO, Amalia.
Students thoughts about the importance and cost of their mobile devices features and services. University of Macedonia, Information Systems Department, Egnatia 156, 54006, Thessaloniki, Greece, 6 de Janeiro 2008.

ANDROID, **Android Developers**. Disponível em:< http://developer.android.com/guide/basics/what-is-android.html>.Acesso em: 5 mar.2012.

CARDOSO, Bruno P. **Framework para Android baseado em Metadados.** ITA - Instituto Tecnológico de Aeronáutica – Comando Geral de tecnologia Aeroespacial, 2011.

CLEMENTE, Fabiane *apud* GIL, A. C. (2007). ***Pesquisa qualitativa, exploratória e fenomenológica: Alguns conceitos básicos*. Disponível** em: <http://www.administradores.com.br/informe-se/artigos/pesquisa-qualitativa-exploratoria-e-fenomenologica-alguns-conceitos-basicos/14316: Acesso em 5 mar. 2012.

FILGUEIRAS, Guilherme B. **3DViewer: Visualizador de Imagens Tridimensionais para o Sistema Operacional Android**. Universidade Federal do Rio de Janeiro – Departamento de Eletrônica e Computação – Rio de Janeiro 2009.

GARTNER, **Technology Research**, Disponível em:< http://www.gartner.com/ >.Acesso em: 5 mar. 2012.

HEXSEL, Roberto A**. Proposta de Ações de Governo para incentivar o Uso de Software Livre,** Universidade Federal do paraná Departamento de Informática – Curitiba, 2002.

HOLZER, Adrian; ONDRUS, Jan; **Mobile Application Market: A developer's perspective;** Information Systems Institute, University of Lausanne, CH-1015 Lausanne, Switzerland; ESSEC Business School, 95021 Cergy, France. 15 de junho de 2009.

IDC Acesso: <http://www.idc.com > em 20/12/2011.

LECHETA, Ricardo R. **Google Android – Aprenda a criar aplicações para dispositivos móveis com Android SDK** . 2.ed. São Paulo : Novatec, 2010. 592p.

MARTINS, Rafael J. Werneck A. **Desenvolvimento de Aplicativo para Smartphone com a Plataforma Android.** PUC - Pontifícia Universidade Católica do Rio de janeiro. Rio de Janeiro, Dezembro de 2009.

MELO, Viviane P. **Impacto da implantação do Sistema Integrado de Gestão Acadêmica – SIGA.** Universidade Federal dos vales do Jequitinhonha, 2011.

OHA, **Open Handset Alliance**, Disponível em:< http://www.openhandsetalliance.com/ >.Acesso em: 5 mar. 2012.

RAYMOND, Eric S. **The Cathedral & The Bazaar**. O'Reilly, 2001. 241p.

RUSSO, Eduardo; FRÈ, Guilherme Almeida; NAGAMACHI, Helio Kazuo; FERES, Tarik Savietto. Universo: **Plataforma Universal de Desenvolvimento de Aplicações Móveis.**
Universidade de São Paulo para a Graduação em Engenharia de Computação. São Paulo, 2010.

SABINO, Vanessa; KON, Fabio**. Licença de Software Livre História e Características.** Universidade de São Paulo - Departamento da Ciência da Computação – Instituto de Matemática e Estatística, Março de 2009.

STALLMAN, Richard; **Free as in Freedom: Richard Stallman's Crusade for Free Software**: O' Reilly Media , 2002.240p.

UZEJKA, Guilherme. M. **Modelando um Cliente Voip Inteligente para a Plataforma Android.** Universidade Federal do Rio Grande do Sul – Departamento de Informática – Porto Alegre, 2011.

APENDICE A – Plano de desenvolvimento

Introdução

Esse documento descreve as estratégias que serão utilizadas no desenvolvimento do projeto, tais como datas de inicio e término de cada iteração e fase, com a finalidade de determinar um cronograma para o desenvolvimento, além disso, mostrar quais serão os membros que participarão do desenvolvimento do projeto e seus respectivos cargos.

Organização do Projeto

Seguindo os princípios do Processo Unificado, cada iteração tem por objetivo gerar uma parte funcional do sistema, cada parte funcional é também um produto final, desta forma, podendo ser executado ou utilizado.

O projeto *PRESENCE* será organizado por funções do sistema e dividido por iterações de modo que as funcionalidade possam ser integradas.

Para a implementação de cada fase do projeto serão definidos os membros envolvidos e seus papéis.

Estrutura Organizacional

A estrutura organizacional adotada no desenvolvimento desse projeto será basicamente seguindo a as informações disponíveis pelo PU, teremos os papéis que serão exercidos pela equipe de desenvolvimento, lembrando que essa estrutura poderá ser adaptada conforme a necessidade do projeto.

Interfaces Externas

O projeto não faz interface com nenhum outro grupo de desenvolvimento, pois as tarefas de todas as fases incluindo a de elaboração e de construção são desempenhadas pelas pessoas envolvidas internamente ao processo do software.

Funções e Responsabilidades

Pessoa	Função do Processo Unificado
Ruhan Bruno Silva– Engenheiro de Software	Coordenador do Projeto Gerente de Implementação Revisor de Requisitos Revisor de Arquitetura Gerente de Configuração Gerente de Controle de Mudanças Revisor de Projeto
Ruhan Bruno Silva– Engenheiro de Software	Revisor de Requisitos Analista de Sistema Especificador de Requisitos Designer da Interface com o Usuário Arquiteto de Software Revisor de Design Gerente de Teste Analista de Teste Designer Implementador Revisor de Código Integrador Designer de Teste Testador Criador Técnico

Planejamento do Projeto

Fase	Iteração	Objetivo Primário (riscos/casos de uso endereçados)	Início/Parada Planejada	Estimativa de Esforço (dias por pessoa)
Iniciação	I 0	Elaborar os artefatos de gerenciamento de projetos.	Início: 07/03/2012 Termino: 09/04/2012	10 dias
Elaboração	E 1	Definir a arquitetura para o projeto, e disponibilizar pelo menos 1 iteração.	Início: 10/04/2012 Termino: 21/04/2012	8 dias
Construção	C 1	Expandir e programar os casos de uso visando todos os requisitos que o Stake Holder solicitou. Disponibilizar a 2º	Início: 22/04/2012 Termino: 11/05/2012	10 dias

		iterações.		
	C 2	Terminar a implementação, e lançar a versão de teste beta.	Início: 12/05/2012 Termino: 26/05/2012	12 dias
Transição	T 1	Preparar o ambiente do domínio e começar a implantação do sistema.	Início: 27/05/2012 Termino: 7/06/2012	7 dias
	T 2	Corrigir os erros de implantação, e terminar a transição com sucesso.	Início: 8/06/2012 Termino: 16/06/2012	8 dias

Recursos do Projeto

O projeto será composto por um único membro, que deverá desempenhar todas as funções e cargos do processo de desenvolvimento.

Para desenvolvimento do projeto foram adquiridos 2 computadores (*notebook*) e um telefone móvel (*smartphone*).

Alguns livros de programação para dispositivos móveis foram adquiridos para treinamento.

Orçamento

O projeto não visará lucro por parte da equipe de desenvolvimento, de forma que o mesmo será desenvolvido sem gerar nenhum custo para o cliente.

APENDICE B – Lista de Riscos

Histórico da revisão

Data	Versão	Descrição	Autor
22/03/2012	1.0	Versão Inicial	Integração de Contexto

Introdução

Finalidade

Este documento descreve os riscos conhecidos no projeto PRESENCE Gerenciamento de Frequência Acadêmica.

Escopo

Esta lista de riscos endereça o projeto inteiro do PRESENCE Gerenciamento de Frequência Acadêmica.

Visão Geral

Os riscos conhecidos na data da publicação deste documento estão listados abaixo, juntamente com as estratégias de mitigação para cada risco.

Riscos

Risco de Escopo: Escala de tempo de desenvolvimento do projeto inflexível.

Gravidade do Risco: Danos Maiores

Descrição

Sabendo que o *PRESENCE* Gerenciamento de Frequência Acadêmica deve estar pronto em 10/06/2012 (dez de junho de dois mil e doze) não havendo a possibilidade de estender o prazo, é necessário que se elabore estratégias para garantir que esse prazo seja cumprido, de forma que o sistema seja entregue com todas as suas funcionalidades em operação.

Impactos

Sistema não funcional ou com funcionalidades limitadas, provavelmente não sendo aprovado pelo cliente.

Indicadores

Atraso das tarefas que deverão ser desenvolvidas, de forma que isso comprove que o tempo planejado não será o suficiente para executar todas as tarefas.

Estratégias de Mitigação

Desenvolver um plano de desenvolvimento do projeto condizente com o grau de conhecimento dos membros da equipe de desenvolvimento, podendo assim ter uma estimativa mais realista do tempo necessário para desenvolver o projeto e qual estratégia deveria ser adotada para que o mesmo seja entregue na data definida. Além disso, é necessário que seja realizado um acompanhamento constate do cronograma para que o mesmo possa ser reajustado conforme a necessidade.

Plano de Contingência

Nenhum

Risco Técnico: Tecnologias não experimentadas

Gravidade do Risco: Média

Descrição

O Sistema de Gerenciamento de Frequência Acadêmica (PRESENCE) fará o uso da rede local da faculdade pra envio de Mensagems. Para execução dessa tarefa será necessário um estudo especifico da rede da universidade e das respectivas tecnologias.

Impactos

O sistema apresentará suas funcionalidades limitadas, levando a dificuldade de comunicação entre o sistema acadêmico.

Indicadores

Falha ou dificuldade de programar o serviço no Sistema de Gerenciamento de Frequência Acadêmica (PRESENCE) .

Estratégias de Mitigação

A equipe de desenvolvimento deverá estudar sobre como implantar essa funcionalidade no sistema de forma que o serviço seja implantado e funcione como proposto.

Plano de Contingência

Nenhum

APÊNDICE C – Plano de Iteração

Histórico da revisão

Data	Versão	Descrição	Autor
08 de Abril de 2012	1.0	Versão Inicial	Integração de Contexto

Índice

Introdução

Finalidade

Este documento traz toda perspectiva do cronograma e das atividades que serão realizadas durante o processo da construção do software. Ele tem por finalidade impor datas que irá servir para conduzir o projeto durante a fase inicial, ou seja, esse plano de iteração conduzirá a uma análise completa sobre o caso de negócios para o sistema, tendo como objetivo principal uma melhor organização de quando as tarefas devem ser executadas.

Escopo

O plano de iteração da fase inicial se aplica ao projeto (PRESENCE) Gerenciamento de frequência Acadêmica. Esse plano será utilizado para orientar os envolvidos no processo de software, ou seja, este documento será utilizado pelo Gerente de Projetos e pela equipe do projeto.

Referências

1. Plano de Visão PRESENCE 1.0

Plano

A fase de iniciação desenvolverá o caso de negócio, os casos de uso do produto PRESENCE. Os principais casos de uso serão desenvolvidos bem como o Plano de Projeto superior. Ao final dessa iteração, será decidido se é necessário financiar e continuar com o projeto com base no caso de negócios.

Tarefas de Iteração

A tabela a seguir ilustra as tarefas com as datas de início e de encerramento planejadas.

Tarefa	**Inicio**	**Fim**
Iniciação	Qua 07/03/2012	Seg 09/04/2012
Elaboração do artefato de visão	Qui 08/03/2012	Sab 10/03/2012
Elaboração do artefato de Caso de Negócio	Seg 12/03/2012	Qua 21/03/2012
Criação do artefato de lista de Riscos	Qui 22/03/2012	Qui 22/03/2012
Criação do artefato Plano de Desenvolvimento de Software	Sex 23/03/2012	Sab 24/03/2012
Elaboração do Artefato Software Architecture Document	Seg 26/03/2012	Sex 30/03/2012
Elaboração do Relatório Sintético do Modelo de Caso de Uso	Qua 28/03/2012	Sab 31/03/2012
Desenvolvimento do diagrama de Casos de Uso	Qui 05/04/2012	Qui 05/04/2012
Desenvolvimento do Plano de Iteração	Seg 08/04/2012	Seg 09/04/2012

Conjunto de artefatos	**Distribuível**	**Proprietário Responsável**
Conjunto de requisitos	Documento de Visão Relatório Sintético de Caso de Uso	Ruhan
Conjunto de Gerenciamento	Plano de Iteração Lista de Riscos de projeto Plano de Desenvolvimento de Software Caso de Negócio	Ruhan
Conjunto de Analise e Design	Documento de arquitetura do Software	Ruhan
Conjunto de Elementos Modelo	Diagrama de Caso de Uso	Ruhan

Recursos

Recursos Financeiros

Não tem custo financeiro previsto para esta fase, já que o projeto não visa lucro.

Casos de Uso

Durante a iteração inicial, todos os casos de uso e agentes importantes são identificados. Os fluxos básicos e fluxos alternativos chaves de cada caso de uso, serão determinados no relatório sintético de casos de uso. O design e a implementação de caso de uso serão iniciados na próxima iteração.

Critérios de Avaliação

Não possui critérios de avaliação nesse plano de iteração, pois não será disponibilizada nenhuma iteração, somente será disponibilizado os artefatos produzidos na fase de Iniciação.