Universidade Estadual do Sudoeste da Bahia
Centro de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação

PresentiaMobile: um Software para Automatização de Lista de Frequência

Rafael Cardoso Oliveira

Vitória da Conquista - BA
Novembro - 2018

Universidade Estadual do Sudoeste da Bahia
Centro de Ciências Exatas e Tecnológicas
Curso de Ciência da Computação

PresentiaMobile: um Software para Automatização de Lista de Frequência

Rafael Cardoso Oliveira

Trabalho de conclusão de curso apresentado junto ao curso de Ciência da Computação da Universidade Estadual do Sudoeste da Bahia, na área de concentração de Ciências Exatas, como requisito parcial à obtenção do título de Bacharel.

Orientador:
Prof. Dr. Marlos Marques

Vitória da Conquista - BA
Novembro - 2018

Pensar, analisar e inventar não são atos anômalos,
mas a normal respiração da inteligência.

Pierre Menard, escritor francês

Agradecimentos

Agradeço a Deus por ter me concedido força, saúde e perseverança para enfrentar cada obstáculo encontrado ao longo dessa jornada. Aos meus pais, Eduardo Amorim e Adelaide Cerqueira, que me motivaram e ajudaram nos momentos mais difíceis. A todos os meus professores, pois sem eles eu não teria chegado à reta final desta trajetória. Em especial, ao meu orientador Prof. Dr. Marlos Marques pelo compartilhamento do seu tema de pesquisa comigo, além da sua generosa ajuda, fundamental na implementação deste trabalho. E, finalmente, a todas as pessoas que estiveram ao meu lado, me proporcionando um convívio proveitoso e alegre no decorrer da minha graduação e da minha vida.

Sumário

Lista de Figuras

Lista de Tabelas

Lista de Símbolos e Abreviaturas

ACK (*Acknowledgments*) - Confirmação positiva.

ADT (*Android Development Tools*) **-** Ferramentas de Desenvolvimento do Android

AIDL (*Android Interface Definition Language*) – Linguagem de Definição de Interface do Android.

API (*Application Program Interface*) **-** Interface de Aplicação do Programa.

ARPA (*Advanced Research Projects Ag*ency) – Agencia

CPD - Centro de Processamento de Dados.

CSV (*Comma Separeted Values*) - Valores Separados por Vírgula.

GUI (*Graphical User Interface*) – Interface Gráfica do Usuário.

IDE (*Integrated Development Environment*) – Ambiente Integrado de Desenvolvimento.

LDB - Lei de Diretrizes e Bases da Educação.

MVC (*Model View Controller*) - Controlador de Exibição de Modelo.

OHA (*Open Handset Alliance*) - Aliança Aberta de Telefonia.

OS (*Operating System*) – Sistema Operacional.

SAV - Sala de Aula Virtual.

SDK (*Software Development Kit*) – Kit de Desenvolvimento de Software.

UML (*Unified Modeling Language*) - Linguagem de Modelagem Unificada.

XML (*Extensible Markup Language*) - Linguagem Extensível de Marcação Genérica.

XP (*Extreme Programming*) – Programação Extrema.

Resumo

O processo de verificação da presença do aluno em sala de aula é obrigatório de acordo a Lei 9.394/96 da LDB (Lei de Diretrizes e Bases da Educação). Contudo a realização deste procedimento na forma como é feita atualmente é falho, maçante e dispendioso em termos de tempo para o docente, a ponto de efetivamente não ser realizada por alguns destes. O desenvolvimento de algum tipo de *software* que ajudasse nesta tarefa incentivaria o cumprimento desta exigência, como ficou constatado pela pesquisa realizada junto aos professores do curso de Ciência da Computação da UESB. O presente trabalho propôs uma classificação para os tipos de solução encontradas na literatura e desenvolveu um aplicativo para dispositivos móveis com o intuito de automatizar o processo de aquisição e contabilização das faltas e presença dos discentes em sala de aula, tornando-o mais eficiente, menos burocrático e menos sujeito a erros, uma vez que a arquitetura aqui proposta possui o mínimo de intervenção por parte dos agentes envolvidos (docentes e discentes).

Palavras Chaves: Lista de Frequência, Desenvolvimento de Aplicações Móveis, Android.

Abstract

The process of verifying the student's presence in the classroom is obligatory according to Law 9.394/96 of the LDB (Law of Guidelines and Bases of Education). However, performing this procedure as it is currently done is flawed, dull and expensive in terms of time to the teacher, to the point of effectively not being realized by some of these. The development of some kind of software that would help in this task would encourage the fulfillment of this requirement, as evidenced by the research carried out with the teachers of the course of Computer Science of UESB. The present work proposed a classification for the types of solution found in the literature and developed an application for mobile devices in order to automate the process of acquisition and accounting of the absences and presence of the students in the classroom, making it more efficient, less bureaucratic and less subject to errors, since the architecture proposed here has the minimum the intervention of the agents involved (teachers and students).

Keywords: Frequency List, Mobile Application Development, Android.

1 Introdução

1.1 Motivação

A lista de presença, ou chamada, é um mecanismo obrigatório para que o professor realize o controle efetivo dos alunos em sala de aula. A maneira tradicional, ou analógica, de registrar a presença do aluno em sala de aula é algo maçante e ineficiente, pois, parte do tempo que poderia ser utilizado para tirar dúvidas dos alunos ou ministrar novos conteúdos, é usado na execução do processo de verificação da presença. Além disso, ao fim de cada semestre (ou unidade), o professor precisa se ocupar em fazer manualmente a contabilização das faltas.

Este processo pode ser otimizado através da utilização de algum tipo de *software* que auxilie o preenchimento do diário de classe. Existem vários trabalhos que sugerem a implementação de aplicativos que visam agilizar o processo de controle de presença discente nas aulas. No entanto, a maioria desses aplicativos apenas automatizam o processo analógico, contribuindo somente no que diz respeito à contabilização dos dados armazenados pelo sistema. Obviamente, por se tratar de um processo automatizado, também ocorrerão melhorias com relação a confiabilidade e a eficácia no gerenciamento da informação.

Um possível avanço neste tipo de sistema seria o desenvolvimento de algum mecanismo que permitisse, não somente a contabilização automática dos dados, mas também um aprimoramento com relação à sua aquisição. Desta forma, o sistema implementado não apenas replicaria o modelo analógico em um meio digital, ou seja, uma simples cópia da lista de presença na tela de um dispositivo, mas promoveria uma automatização completa do processo. O aplicativo proposto neste trabalho se enquadra nesta categoria e será desenvolvido na plataforma Android para dispositivos móveis, considerando que os *smartphones* estão cada dia mais presentes na vida das pessoas.

1.2 Objetivos

O projeto tem como propósito desenvolver um aplicativo para dispositivos móveis que atenda as necessidades do professor, durante o processo de efetuar o controle da frequência dos alunos em sala de aula, de modo que a aquisição de todos os dados referentes ao aluno e à contabilização do número de faltas, sejam feitas de maneira automática, tornando assim esse processo simples e eficiente. Além disso, o desenvolvimento deste trabalho compreende o estudo da linguagem Android e suas ferramentas, a definição do problema a ser resolvido, a apresentação da solução de forma detalhada e a implementação de um aplicativo que resolve o problema.

1.3 Estrutura do Trabalho

O presente trabalho possui a seguinte estrutura de organização. No capítulo 2 - **Referencial Teórico** - será abordada toda a fundamentação teórica, a fim de esclarecer os pressupostos teóricos que dão sustentação à pesquisa; uma breve introdução à plataforma Android, à ferramenta de desenvolvimento Android Studio e à metodologia utilizada para alcançar com êxito os objetivos deste trabalho. No capítulo 3 - **Estado da Arte** - é feita uma análise de outros sistemas correlatos, que também desenvolveram aplicativos para otimizar o controle de frequência discente de maneira diferente do *software* que pretende-se desenvolver neste trabalho. No capítulo 4 - **PresentiaMobile -** é feita uma contextualização à respeito do trabalho que foi desenvolvido, os processos e os procedimentos utilizados na implementação do *software*, de modo que possa realizar de maneira satisfatória a obtenção dos dados gerados no controle da presença dos seus usuários e a contabilização correta desta. No capítulo 5 - **Avaliação -** será apresentada uma avaliação do desenvolvimento do projeto como um todo e as conclusões obtidas após o término de todas as suas etapas. Por fim, o sexto e último capítulo – **Conclusão e Trabalhos Futuros** - apresenta as conclusões e as perspectivas de trabalhos futuros.

2 Referencial Teórico

A LDB (Lei de Diretrizes e Bases da Educação), através da Lei 9.394/96, diz que uma vez dentro do território brasileiro, qualquer que seja a instituição de ensino, ela é obrigada a exigir um mínimo de 75% de frequência do aluno para que este possa ser aprovado.

> Art. 24, inciso IV. O controle de frequência fica a cargo da escola, conforme o disposto no seu regimento e nas normas do respectivo sistema de ensino, exigida a frequência mínima de setenta e cinco por cento do total de horas letivas para aprovação. (BRASIL LDB, 1996)

A obrigação de fiscalizar a assiduidade do aluno nas disciplinas, normalmente fica a cargo do professor. Entretanto, a grande maioria dos docentes não faz uso de mecanismos automatizados para realizar esse procedimento, de modo que cada um cria uma abordagem própria para agilizar esse processo. De uma maneira geral, essa informação fica centralizada em folhas de papel ou no próprio computador do professor, sem estar disponível para o acesso do aluno.

Para adquirir, armazenar, gerenciar e disponibilizar estas informações, as instituições de ensino foram desenvolvendo, ao longo do tempo, mecanismos e ferramentas para dar suporte a esta exigência legal. Esses mecanismos e ferramentas serão analisados nas subseções seguintes.

2.1 Diário de classe

Moretto (2004) afirma que o diário de classe é um mecanismo dispendioso e sujeito a erros de informações; isso devido ao trabalho repetitivo e demorado, que leva o professor a trancrever todas as notas e presenças registradas no diário ao longo do semestre letivo, para fins de um relatório final, que deve ser entregue ao coordenador do curso para, só então, fazer o lançamento do resultado final no sistema.

Diante desse desperdício de tempo, uma solução viável foi o desenvolvimento do diário de classe digital, disponível em *softwares* para dispositivos móveis e *laptops*.

Esse mecanismo permite que cada professor apenas preencha os dados referentes ao diário no início do semestre letivo e ao fim do período letivo, transfira todos esses dados ao sistema evitando todo o trabalho manual e reduzindo a possibilidade de erros.

O diário de classe digital não só proporciona mais facilidade e eficácia na realização das atividades semestrais do professor, como também vem a substituir aspectos inconvenientes do antigo diário, no que diz respeito à quantidade de papel necessária para a confecção das cadernetas impressas, o que corresponde a um grande número de árvores derrubadas, além de se considerar a facilidade de perda, bem como danos e esquecimento dos dados, pois eles estão contidos em folhas de papel; e ainda a falta de acesso dos alunos às informações em tempo real, já que as informações contidas no diário somente estão disponíveis ao professor.

Assim, todo o aprimoramento neste utensilio essencial, de uso obrigatório do professor, durante todo período letivo, surge da iniciativa de desenvolvedores que buscam criar *softwares* no intuito de facilitar e tornar mais eficazes não só atividades como esta, mas outras práticas educacionais que fazem parte do diário de classe, a exemplo da lista de frequência, analisada na subseção seguinte.

2.2 Lista de Frequência

De acordo com Ghodekar *et al.* (2013), o registro da frequência dos alunos em sala de aula nas instituições de ensino é de grande importância, porém, com o método adotado para realizar a checagem dos alunos que estão presentes ou ausentes através de uma lista em folha de papel, perdem-se, pelo menos, 10 minutos.

Isso demonstra que tal processo demanda tempo e é falho, pois deve ser considerado que essa lista será facilmente perdida e/ou danificada, o aluno pode esquecer de assinar o seu nome ou até mesmo aproveitar as falhas para falsificar a assinatura de outro estudante que está ausente. Desta forma, com o intuito de aprimorar o combate a essas deficiências na contabilização e aquisição da frequência dos alunos, fez se necessário informatizar todo o processo tradicional de execução da lista de

frequência, através do desenvolvimento de alguns *softwares*, no propósito de modernizar e solucionar as deficiências mencionadas anteriormente.

Os *softwares* que estão sendo desenvolvidos apresentam abordagens diferentes, e os autores do presente trabalho propuseram-se a criar uma classificação que faz a divisão em modelo digital e digital automatizado. Isto, porque ambos os modelos são distintos no método de aquisição dos dados referente á frequência dos alunos, o que será explicado nas subseções seguintes.

2.2.1 Modelo Tradicional

O registro da frequência na forma tradicional, através do papel e da caneta, ainda é uma opção adotada pelos professores dentro de várias instituições de ensino, mesmo sabendo que tal processo demanda tempo para ser realizado em sala de aula e que esse tempo poderia ser melhor empregado no método de ensino-aprendizagem dos alunos durante as aulas (WEISER, 1991).

O modelo tradicional apresenta duas possíveis formas de execução, a primeira utilizada normalmente em ambientes escolares dos primeiros anos do ensino fundamental e médio, em que o professor fica responsável por fazer uma chamada oral pelo nome do estudante e por meio de identificadores documentam a presença ou falta do aluno (Figura 2.1a). A segunda forma é feita habitualmente no ambiente acadêmico, em que uma lista é passada entre os discentes da turma e eles mesmos assinam seu nome, confirmando a sua presença (Figura 2.1b).

UNIV. ESTADUAL DO SUDOESTE DA BAHIA

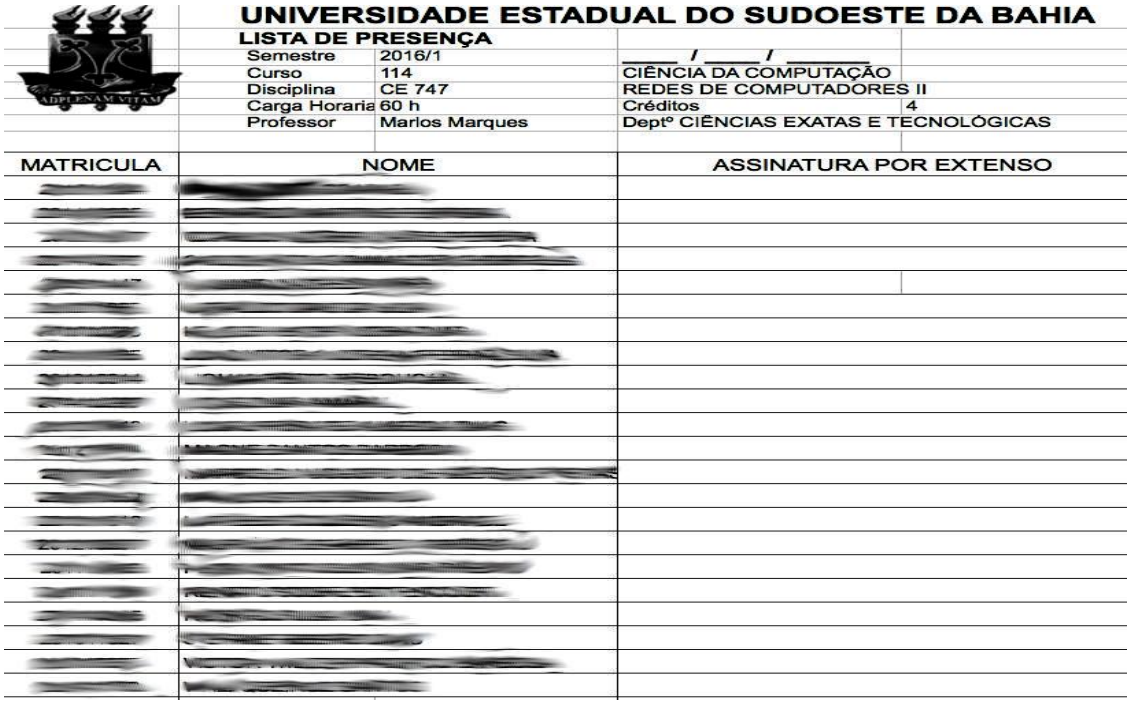
FOLHA DE FREQUÊNCIA DO ALUNO

Vitória da Conquista

CE 755	SISTEMAS DISTRIBUÍDOS	Deptº CIÊNCIAS EXATAS	CIÊNCIA DA COMPUTAÇÃO	Turma T01
2014/2	Professor(a) [redacted]	Carga Horária 60 h	4 créditos	Folha 8

MATRÍCULA	CURSO	NOME	REGISTRO DE FALTA																												TOTAL DE FALTAS
			AULA	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	
			DIA	20	21	27	28	03	04	10	11	17	18	24	25	01	02	08	09	15	16	22	23	29	30	05	06	12	13	26	27	03	04	
			MÊS	08	08	08	08	09	09	09	09	09	09	09	09	10	10	10	10	10	10	10	10	10	10	11	11	11	11	11	11	12	12	
[redacted]	114	[redacted]		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	2	2	•	•	•	•	•	•	•	•	•	•	•	•	4
[redacted]	114	[redacted]		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	2	•	2	•	•	•	•	•	4
[redacted]	114	[redacted]		•	•	•	•	•	•	•	•	•	•	•	2	•	2	•	•	•	•	•	•	2	2	•	•	•	•	•	•	•	•	8
[redacted]	114	[redacted]		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	2	2	•	•	•	•	•	•	•	•	•	•	•	•	4
[redacted]	114	[redacted]		2	2	•	•	•	•	•	•	•	•	•	•	•	2	•	•	•	•	2	•	•	•	•	2	•	•	•	•	•	•	10
[redacted]	114	[redacted]		•	•	•	2	•	•	•	•	•	•	•	•	•	•	•	•	2	•	•	•	•	2	2	2	2	2	2	2	2	2	22
[redacted]	114	[redacted]		2	•	•	•	•	•	•	•	•	•	2	•	•	•	•	•	•	2	•	•	•	•	•	•	•	•	2	•	•	•	8
[redacted]	114	[redacted]		2	•	•	•	•	•	•	•	•	•	•	2	•	•	•	•	2	2	•	•	•	•	•	2	2	•	•	•	2	•	14
[redacted]	114	[redacted]		•	•	•	•	•	•	•	•	•	•	2	•	•	2	•	•	2	2	•	•	2	•	•	•	•	•	•	•	•	•	10
[redacted]	114	[redacted]		2	2	•	2	•	2	•	•	•	•	2	•	•	2	•	•	•	•	2	•	2	•	•	•	2	2	2	2	2	2	26
[redacted]	114	[redacted]		•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	•	2	•	•	•	•	•	2

Figura 2.1a – Exemplo da lista de chamada analógica: lista de presença com marcação



UNIVERSIDADE ESTADUAL DO SUDOESTE DA BAHIA

LISTA DE PRESENÇA

Semestre	2016/1	____ / ____ / _____
Curso	114	CIÊNCIA DA COMPUTAÇÃO
Disciplina	CE 747	REDES DE COMPUTADORES II
Carga Horaria	60 h	Créditos 4
Professor	Marlos Marques	Deptº CIÊNCIAS EXATAS E TECNOLÓGICAS

MATRICULA	NOME	ASSINATURA POR EXTENSO
[redacted]		

Figura 2.1b – Exemplo da lista de chamada analógica: lista de presença com assinaturas

Apesar do modelo tradicional ser simples e de fácil execução, ele apresenta deficiências que há 50 anos atrás não poderiam ser solucionadas, pois os computadores da 4ª geração, com a capacidade de armazenar e contabilizar dados, somente vieram a surgir em 1975 e foram sendo aprimorados ao longo dos anos.

Hoje em dia, os professores já fazem o uso de *softwares* em *laptops*, com o objetivo de agilizar e aprimorar todo o processo equivalente à contabilização do número de faltas dos discentes durante o período letivo, ocasionando assim a transição do modelo tradicional para o modelo digital, que será mencionado na próxima subseção.

2.2.2 Modelo Digital

Os autores deste trabalho afirmam que o modelo digital é sem dúvida um início do aprimoramento no modelo tradicional de execução da lista de frequência, pois todo procedimento cansativo de contagem das faltas e presenças dos alunos, que antes era feito manualmente pelos professores, hoje já está sendo substituído pelo uso de *softwares* disponíveis em *desktops* e *laptops*.

Segundo Gleiser (2011), o uso de computadores já faz parte do cotidiano escolar e acadêmico dos professores, pois em pesquisa realizada nas instituições de ensino do Distrito Federal, 70% desses profissionais já fazem uso de *laptops* em suas aulas e afirmam que a ferramenta, juntamente com os *softwares*, tem agregado qualidade e dinamismo às suas atividades, a exemplo da execução do processo da lista de frequência dos discentes.

O *software* que geralmente é utilizado pelos docentes durante o processo de verificação da presença do aluno é o Microsoft Excel. Isto porque, por meio dele, eles fazem o uso de planilhas para inserir os dados dos alunos no programa, como também as faltas e presenças que, após todo o procedimento, são contabilizados e, ao fim do período letivo, gera-se o resultado final com o total de faltas de cada aluno (Figura 2.2). No entanto, mesmo substituindo todo esse trabalho de contagem que antes era feito pelos docentes, a abordagem digital ainda possui deficiências no que diz respeito à aquisição dos dados, pois o processo de obtenção das informações dos alunos ainda continua sendo feito manualmente.

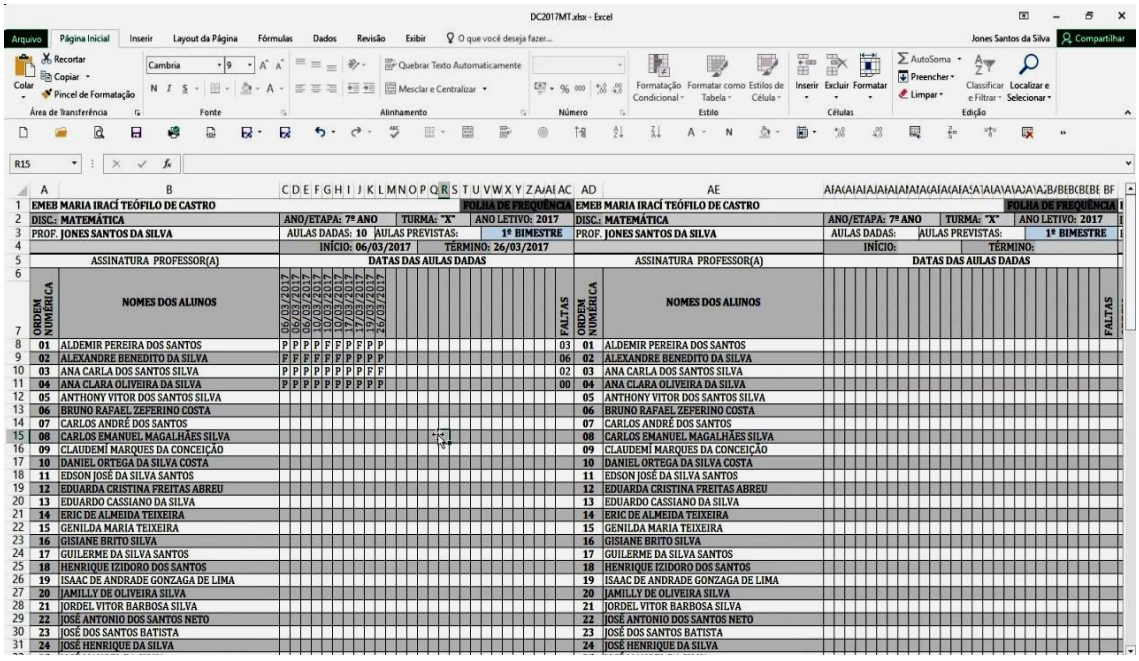


Figura 2.2 – Exemplo da lista de chamada feita no modelo digital usando Microsoft Excel

Deste modo, para solucionar esta deficiência no modelo digital estão sendo desenvolvidos aplicativos para dispositivos móveis que realizam a aquisição dos dados de maneira automática. Esses aplicativos, a exemplo do que vem sendo desenvolvido

neste trabalho, fazem parte da classificação digital automatizada, modelo que será explicado a seguir.

2.2.3 Modelo Digital Automatizado

De acordo com os autores deste trabalho, o modelo digital automatizado tem como perspectiva aprimorar por completo o obrigatório processo de execução da lista de frequência e oferecer outros benefícios, como por exemplo mais segurança para as crianças e jovens que frequentam as instituições de ensino. Este modelo também é capaz de monitorar a entrada e saída dos estudantes e enviar essas informações para os seus pais.

Para chegar a esse nível de aperfeiçoamento, uma opção seria fazer com que os professores substituam a utilização de *softwares* nos *laptops*, durante a verificação da frequência dos alunos, pelo uso de aplicativos em dispositivos móveis, pois esse tipo de aplicação possui um número maior de ferramentas e funcionalidades necessárias para alcançar o objetivo desejável.

Nesse contexto, existem trabalhos correlatos a este que usam outros tipos de tecnologias para chegar ao mesmo objetivo, como por exemplo o uso do *bluetooth* na comunicação entre dispositivos móveis (*smartphones* e *tablets*); a utilização do cartão eletrônico e da biometria, uma tecnologia atual e segura, que, através de impressão digital ou reconhecimento facial, é capaz de identificar o estudante. Por fim, existe também o uso de *web-service*, um portal de acesso exclusivo aos alunos e professores que estão vinculados a alguma instituição de ensino, a exemplo do portal Moodle Acadêmico.

Dessa forma, é notável que as tecnologias de informação são cada vez mais essenciais na vida das pessoas e, hoje, a utilização de dispositivos móveis tem crescido bastante juntamente com a implementação de aplicativos, como os que fazem o registro da frequência dos alunos. Este trabalho se enquadra nesta categoria, pois tem como objetivo implementar um aplicativo *mobile* que faz todo processo de aquisição e contabilização dos dados referente à frequência dos alunos, utilizando o sistema operacional Android, que será detalhado na próxima seção.

2.3 Plataforma Android

O Android é um sistema operacional inicialmente desenvolvido pela Google e atualmente é mantido pela *Open Handset Alliance* (OHA), grupo de empresas cujo objetivo é a criação de plataformas únicas e *open source* (código aberto), que podem ser utilizadas em dispositivos móveis para o desenvolvimento de aplicações coorporativas. Baseado no núcleo do Linux, os seus aplicativos são desenvolvidos utilizando a linguagem Java, o que permite uma maior facilidade de adaptação dos fabricantes de dispositivos móveis e desenvolvedores de aplicativos (LECHETA, 2016).

Sua primeira versão, denominada Alpha, foi lançada em 2008 e é considerada o ponto de partida para o início evolutivo do Android, o que deu origem a outras versões com a peculiar característica de ser denominada com um nome de um doce (Figura 2.3). Em abril de 2009, foi lançada a versão Cupcake que trouxe diversas melhorias para o sistema operacional, como câmera, GPS, *upload* de fotos e vídeos para o YouTube, além do surgimento dos *widgets* (mini aplicativos executados na tela inicial). Logo em seguida, surgiu a versão Donut, lançada em setembro de 2009, que inovou o Android com a conversão de texto em voz e vice versa. No ano de 2013, a versão KitKat trouxe aperfeiçoamentos no *bluetooth*, *print framework*, sensores e criação da API Transitions, usada pelos desenvolvedores na criação e animação das interfaces gráficas. Com uma nova política de privacidade, a versão Marshwallow, lançada em 2015, tem como novo recurso o Runtime Permissions (permissão em tempo de execução), que concede ao usuário a liberdade de permitir ou negar a permissão de algum recurso. Por fim, surgiu a versão mais atual, chamada Oreo, lançada em 2017, que se destacou pela economia do consumo de bateria e por dar suporte a outros idiomas além do nativo do sistema.

Figura 2.3 - Linha do tempo das versões Android (QUEIRÓS, 2017)

Atualmente, o sistema operacional Android é o mais utilizado pelos usuários de dispositivos móveis em comparação com seus concorrentes, o iOS e o Windows Phone. De acordo com a empresa de pesquisa e consultoria de tecnologia da informação Gartner (GARTNER, 2016), no ano de 2015, o Android representava 76% do número de dispositivos adquiridos no mercado, enquanto o seu concorrente, o iOS, detinha 20,4%, e o Windows Phone apenas 2,8% do mercado.

Outra pesquisa realizada em 2017 pela International Data Corporation (IDC, 2017) revela que o Android vem crescendo cada vez mais nos últimos anos, pois entre o período de 2014 até 2017, a plataforma conseguiu passar de 69,3% para 82,8% no número de dispositivos utilizando o seu sistema operacional (Figura 2.4).

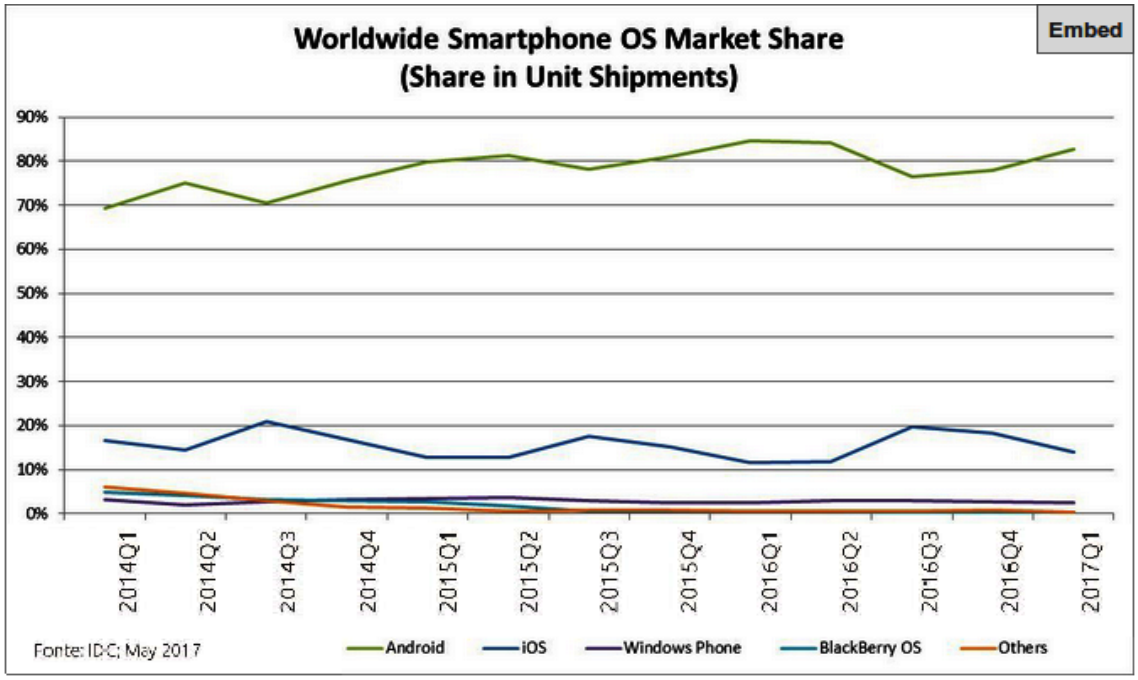


Figura 2.4 – Gráfico comparativo entre sistemas operacionais móveis (IDC, 2017)

Desta forma, devido aos índices de crescimento no número de usuários da plataforma Android apresentados nas pesquisas acima, esta arquitetura foi escolhida para o desenvolvimento deste trabalho.

2.3.1 Arquitetura Android

Conforme Wall (2008), a plataforma Android não é considerada apenas um sistema operacional, ela é um conjunto de ferramentas e bibliotecas acopladas a um *kernel* do Linux, formando um sistema mais completo e robusto. Essas ferramentas interagem entre si em um formato de pilha, onde cada camada oferece recursos para a camada que se encontra acima dela (Figura 2.5).

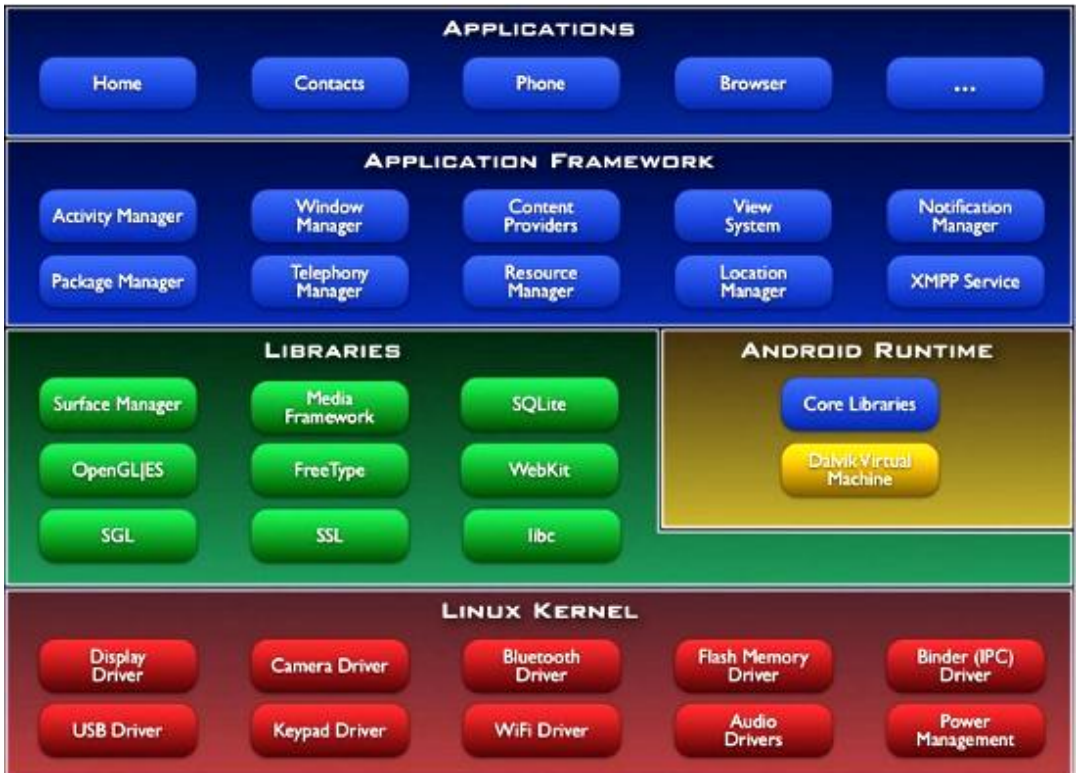


Figura 2.5 – Arquitetura Android (RABELLO, 2009)

No seu núcleo, está um *kernel* do Linux, na versão 2.6, com algumas alterações, responsáveis pelo gerenciamento de memória e processos, além do sistema de arquivos. Junto a ele estão os *drivers* de mais baixo nível, que controlam tela, teclado, dispositivos de rede, gerenciamento de energia, entre outros. Estas estruturas fornecem a base sobre a qual a plataforma é construída.

Na camada logo acima, encontram-se bibliotecas escritas em C e C++, que são responsáveis pela parte gráfica (Surface Manager e OpenGL), multimídia, banco de dados (SQLite), suporte a *browsers* (WebKit), além de camadas de rede e

gerenciamento de fontes. Estas bibliotecas prestam serviços à camada seguinte, que pode ser chamada de *framework* de aplicações. Nesta camada, encontram-se as APIs de suporte ao desenvolvimento, que simplificam o acesso ao sistema de localização, de telefonia, de notificações e outros, além de prover a estrutura dos programas do Android, gerenciando o sistema de *views*, os *content providers*, as *activities* e o sistema de janelas.

Por fim, encontra-se a camada de aplicações básicas do Android, composta por *softwares* básicos do sistema como: a opção telefone, utilizada para realizar ligações no aparelho; a opção agenda contatos, útil para salvar todos os contatos no aparelho; e o *browser*, que possibilita navegar pela internet no aparelho. Estes *softwares* podem ser substituídos em caso de preferência do usuário, sem que seja comprometido o funcionamento do sistema. Também existe a possibilidade de serem instalados inúmeros outros *softwares* que agreguem todo o tipo de funcionalidade ao aparelho.

2.3.2 Android SDK

Os desenvolvedores de aplicativos Android dizem que não é possível desenvolver nenhum tipo de aplicativo sem a ferramenta principal chamada Android SDK (*Software Development Kit*), pois ela é a engrenagem fundamental para a implementação de qualquer tipo de aplicação em Android. De acordo com Lecheta (2016), a ferramenta Android SDK é essencial no desenvolvimento de aplicações para dispositivos móveis, pois possui recursos para construir, testar e depurar código, além de incluir documentação e utilitários que auxiliam no desenvolvimento de aplicações.

A criação desse ambiente de desenvolvimento em Android, utilizando a linguagem de programação Java, é de fundamental importância, independente do fato de a IDE utilizada ser o Eclipse ou Android Studio. O fato é que, para começar a programar, é necessário que o ambiente de desenvolvimento esteja pronto com todos os pacotes e bibliotecas instalados, para, só então, posteriormente, começar a inserir códigos. Se a ferramenta escolhida para o desenvolvimento Android for o Eclipse, é necessário adicionar o *plug-in* ADT (*Android Development Tools*) à sua instalação, pois somente ele permite que a IDE compile aplicativos para o Android, inicie os emuladores

e se conecte aos seus serviços de depuração, além de permitir a edição e compilação de arquivos XML do Android, através da AIDL (*Android Interface Definition Language*). Mas, se a ferramenta escolhida for o Android Studio, IDE, lançada em 2013 pela Google e eleita a preferida para desenvolver aplicativos móveis, a instalação do plug-in ADT torna-se desnecessária e, assim, o processo de desenvolvimento da linguagem de programação torna-se mais fácil.

2.3.3 Android Studio

O Android Studio é uma ferramenta que foi lançada na conferência Google I/O, em 16 de maio de 2013, e se tornou a IDE padrão para o desenvolvimento Android. Ela oferece gratuitamente todo o suporte necessário para a criação de aplicativos voltados para dispositivos móveis, TVs ou *tablets.* Sua arquitetura é composta por várias janelas que permitem ao usuário executar múltiplas tarefas durante o processo de desenvolvimento do *software* (Figura 2.6).

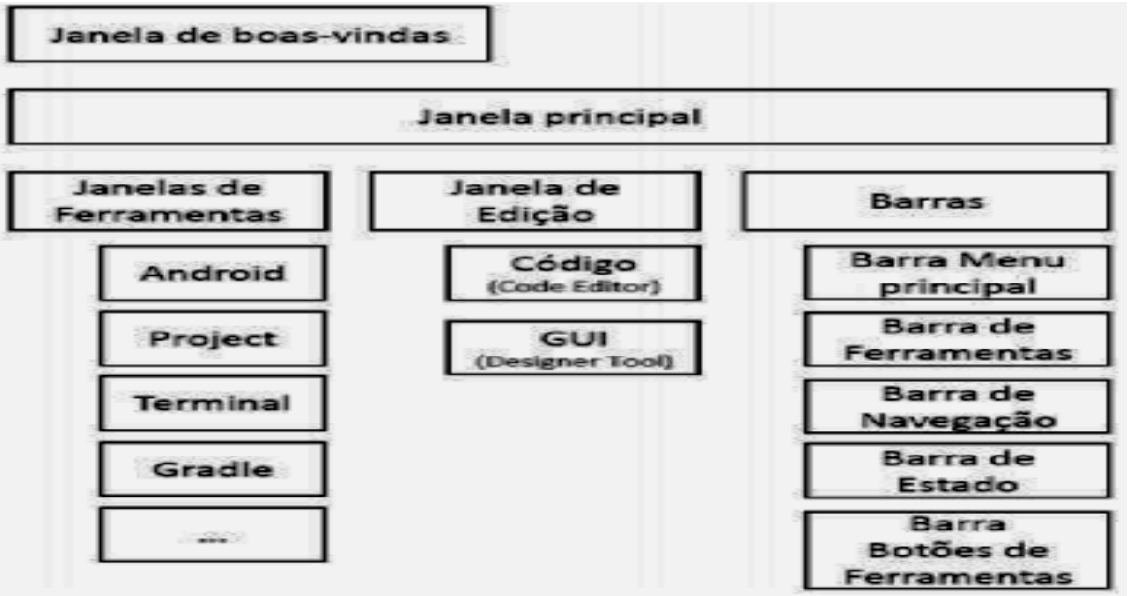


Figura 2.6 – Organização da arquitetura da IDE Android Studio (QUEIRÓS, 2016)

Antes de iniciar todo o processo de desenvolvimento, é necessário fazer o *download* das versões do Android para o qual deseja desenvolver juntamente com as respectivas bibliotecas e pacotes na janela Android SDK Manager do *software* (Figura 2.7).

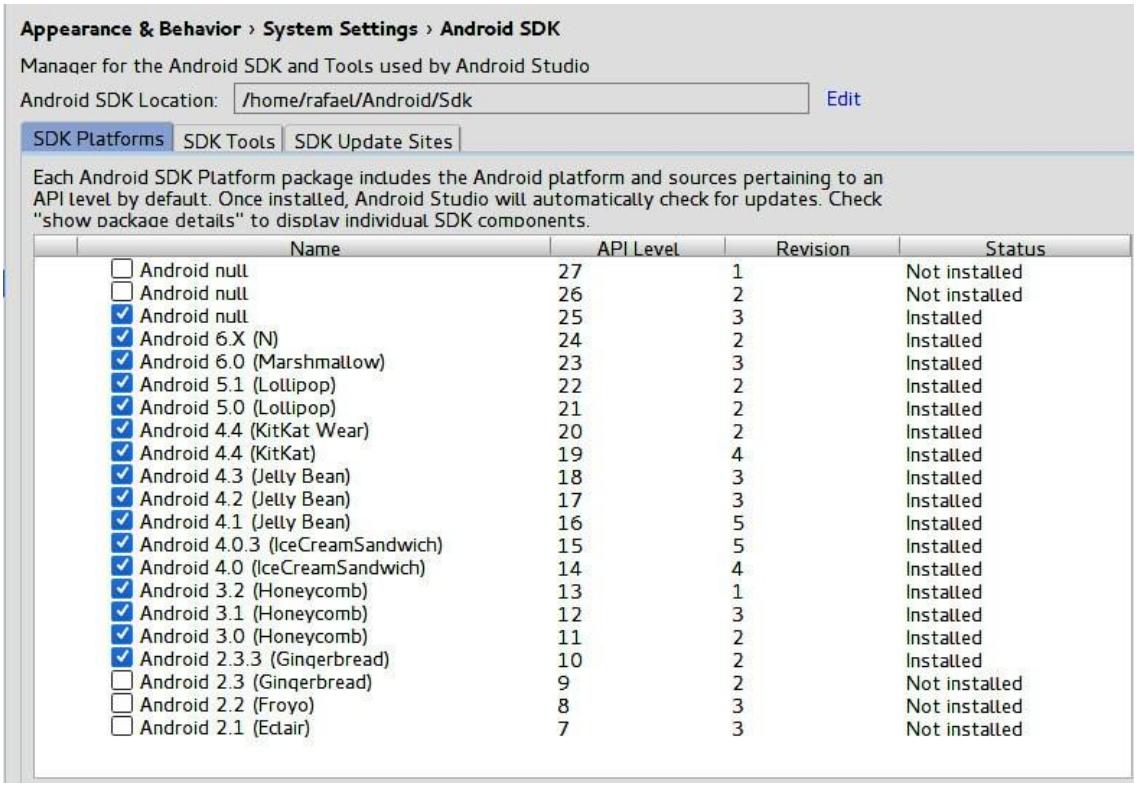


Figura 2.7 - Janela do Android SDK Manager

Quando o primeiro projeto for criado na ferramenta, uma janela principal constituída por barras e painéis é exibida (Figura 2.8). Nesta janela, cada uma das barras e painéis enumerados possui um conjunto de funções e ferramentas. A primeira barra, chamada barra de menus, contém um conjunto de ferramentas que podem ser usadas para executar tarefas no Android Studio. A seguinte, denominada barra de ferramentas, possui vários botões para permitir ações frequentes dentro da IDE. A terceira é conhecida como barra de navegação, ela permite aos usuários navegarem por ficheiros e pastas do projeto. As barras de botões e ferramentas presentes no lado esquerdo, direito e inferior da janela principal, possuem botões para ativar ou desativar cada uma das opções de ferramentas. A quinta e ultima é denominada barra de estado, responsável por exibir mensagens informativas sobre o projeto e as atividades no Android Studio. Por fim, existem dois painéis, o de edição, que permite ao usuário editar códigos; e o de ferramentas do projeto, que demostra toda a hierarquia dos ficheiros do projeto.

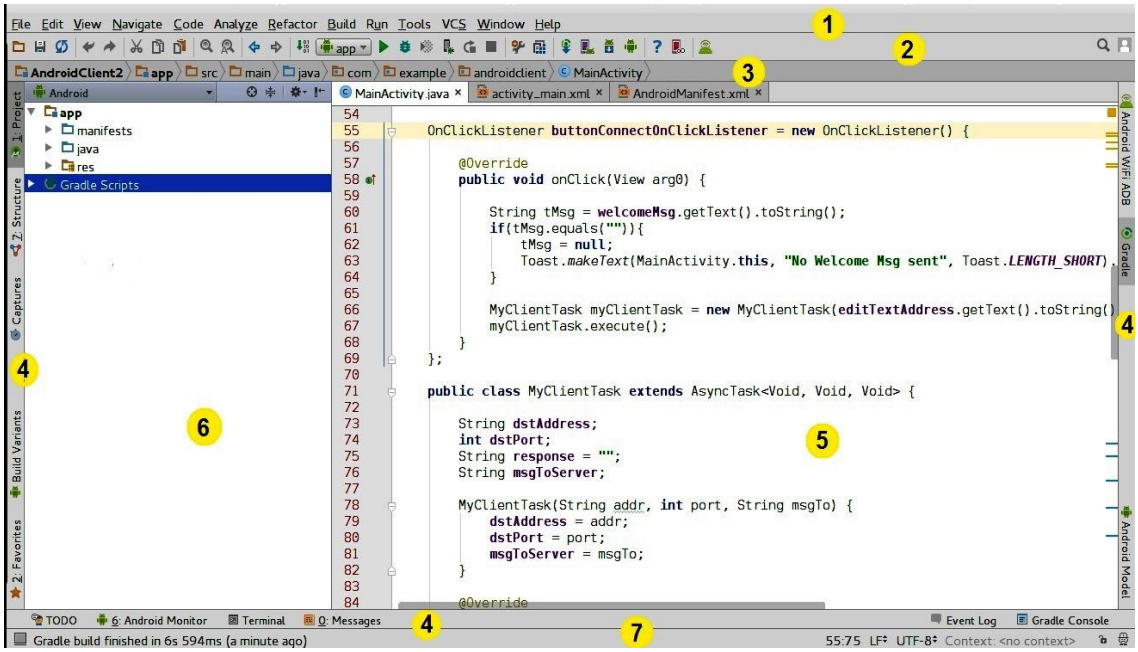


Figura 2.8 – Janela principal do Android Studio

Durante o processo de desenvolvimento dos aplicativos é necessário emular a aplicação, pois o programador deve observar como está ficando a interface e se foi encontrado algum erro. Para isso, o Android Studio oferece um emulador chamado AVD (*Android Virtual Device Manage*r), exibido na Figura 2.9 e que permite fazer este procedimento em uma máquina virtual no próprio *laptop* ou diretamente no dispositivo móvel, visto que, para isso, é necessário conectar um cabo no celular e na porta USB do computador e executar os aplicativos diretamente do aparelho (Figura 2.10). Isso, sem dúvida, facilita os testes durante o desenvolvimento e torna-o bem mais produtivo.

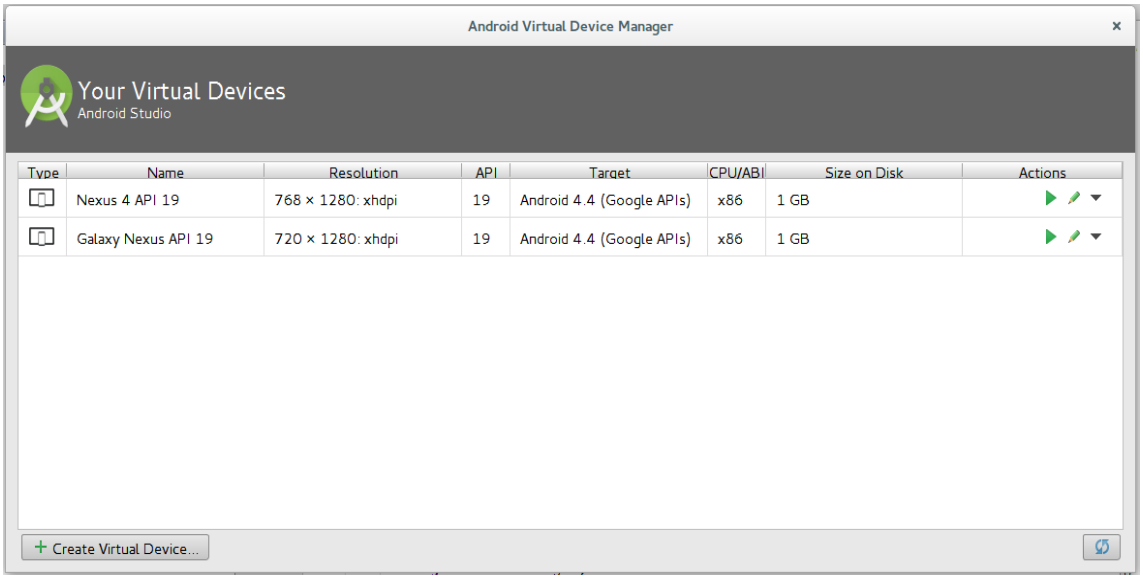


Figura 2.9 – Janela do *Android Virtual Device Manager*

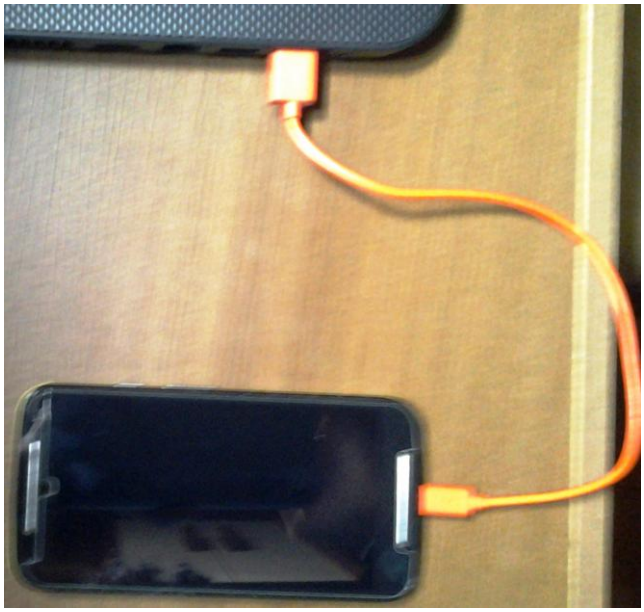


Figura 2.10 – Exemplo da emulação do *software* via dispositivo móvel

2.3.4 Interface Gráfica

A IDE Android Studio possui uma grande variedade de componentes que podem ser usados na construção do *layout* da interface gráfica dos aplicativos. Uma das classes principais responsáveis é a View (Figura 2.11), considerada a classe mãe geradora dos elementos visuais do Android, em conjunto com suas subclasses, responsáveis por criar os componentes gráficos das telas. No entanto, para fazer isso, é preciso implementar o método *onDraw* (Canvas) em cada subclasse durante a fase de desenvolvimento do aplicativo.

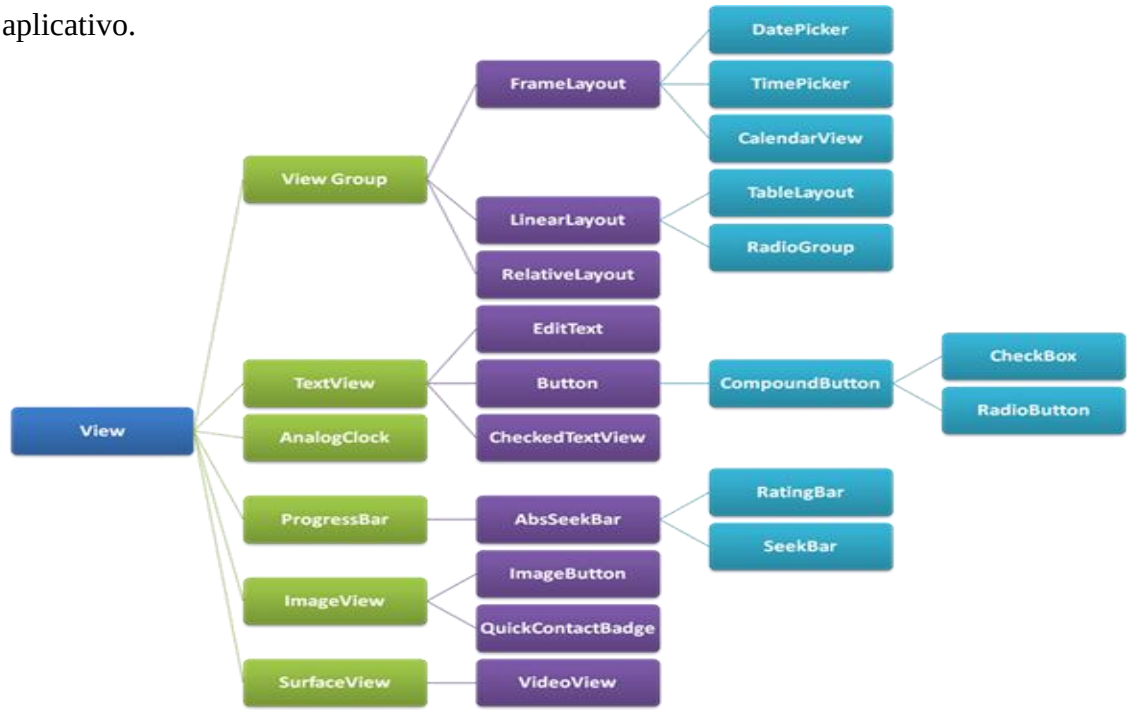


Figura 2.11 – Hierarquia da classe *View* do Android

Outra classe muito importante na criação, organização dos botões e imagens da GUI é a ViewGroup, que também aparece na Figura 2.11 acima e apresenta uma hierarquia de subclasses. Em tal figura, estão presentes ainda duas das principais: a LinearLayout e a RelativeLayout, cujo a primeira tem a função de organizar uma sequência de componentes como imagens e campos de texto presentes no *layout* de aplicações para que eles fiquem posicionados corretamente na horizontal ou vertical. E a segunda, a finalidade de posicionar os componentes presentes na interface gráfica ao lado, abaixo ou acima dos já existentes, dado que para isso é necessário definir um id para cada componente, pois o posicionamento de um depende do outro. Portanto, a utilização da classe View e de suas subclasses no desenvolvimento de uma interface gráfica atraente e acessível ao usuário é um requisito muito importante em um aplicativo, pois a primeira coisa observada pelos usuários de dispositivos móveis na interface de um aplicativo é a boa aparência e a organização dos elementos ali presentes.

2.3.5 Arquitetura Cliente/Servidor

Na comunicação em Rede de Computadores é necessário o uso de estruturas que permitem que dois ou mais dispositivos troquem informações entre si. Estas estruturas são chamadas de *sockets* e representam um ponto de conexão para uma rede TCP/IP. Esta mesma estrutura faz uso de uma arquitetura muito conhecida pela sua simplicidade e relevância na literatura de rede de computadores denominada arquitetura Cliente/Servidor que funciona da seguinte forma: o servidor escolhe uma determinada porta e fica aguardando suas conexões. O cliente deve saber previamente qual dispositivo irá fornecer o (HOST ou IP) e a porta de que o servidor está aguardando conexões, para solicitar uma conexão. Se nenhum problema ocorrer, o servidor aceita a conexão, gerando um *socket* em uma porta qualquer do lado servidor, criando assim um canal de comunicação entre o cliente e o servidor (Figura 2.12).

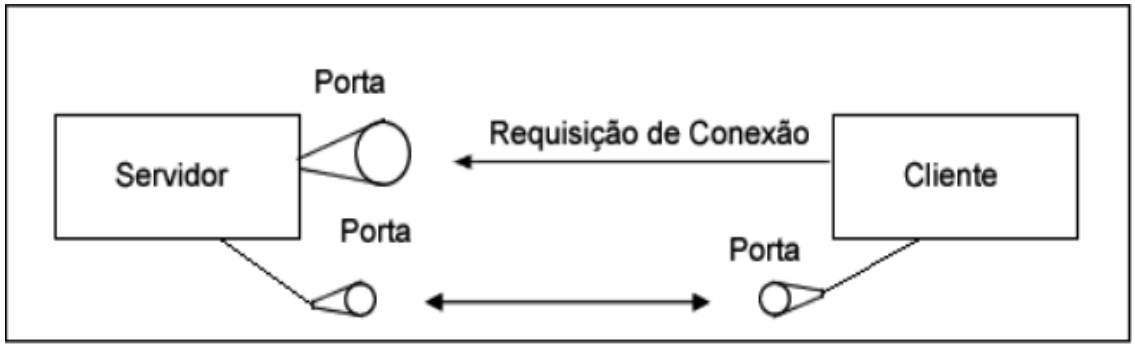


Figura 2.12 - Criação do canal de comunicação via arquitetura Cliente/Servidor

Geralmente, o comportamento do servidor é ficar constantemente aguardando novas conexões e gerar *sockets* para atender as solicitações de clientes. A arquitetura Cliente/Servidor apresenta ainda dois protocolos de comunicação e transporte de dados chamados TCP e UDP que fazem em comum o encapsulamento dos segmentos e o envio desses em pacotes de dados, porém existem diferenças entre eles, pois o UDP realiza o envio dos datagramas IP encapsulados sem que seja necessário estabelecer uma conexão, diferente do protocolo TCP, que possui controle sobre a conexão realiza esse envio de maneira confiável e sequencial.

2.4 Extreme Programming

Para a construção do aplicativo proposto neste trabalho, será usado o método ágil de desenvolvimento de *software* chamado Extreme Programming (XP). Diferente dos modelos em cascata e incremental que primeiro planeja todo ciclo de processos encadeados entre si, antes de começar de fato a implementação, o XP parte logo para a fase de desenvolvimento do *software*, em consequência do seu dinamismo, pois a todo o momento ele está sendo modificado por um dos membros da equipe de desenvolvedores de *softwares*.

Desta forma, como todos os modelos de processo de *software* possui suas vantagens e desvantagens, o método XP não é diferente e suas vantagens são: a facilidade de adaptar rapidamente eventuais mudanças nos requisitos, visto que esse recurso ao ser realizado nas metodologias tradicionais requer um pouco mais de tempo e recurso; permitir a comunicação entre os membros da equipe, para saber se está tudo ocorrendo conforme foi pensado durante a criação do *software* ou existe algum impasse; e o fato do cliente poder acompanhar as fases de desenvolvimento do *software* e não precisar esperar até o final de todo o processo para ver o *software* funcionando, como ocorre nas metodologias tradicionais. As desvantagens dizem respeito ao fato de não existir uma avaliação de riscos, o que impossibilita criar uma análise e estratégia que busque diminuir os erros e o fato da análise de requisitos ser informal, o que pode não

ser bem visto pelos clientes, pois eles poderão se sentir inseguros quanto ao bom funcionamento do sistema.

Sommerville (2007) descreve a metodologia do Extreme Programming, onde os programadores que optam pelo modelo XP trabalham em pares, pois durante a divisão de tarefas para facilitar na implementação de *softwares*, os dois membros da equipe devem esta cientes de cada fase do sistema e desenvolvem testes para cada tarefa antes da escrita do código. Todos esses testes devem ser executados com sucesso quando um novo código é integrado, pois é essencial que o programador compreenda todo o ciclo do processo Extreme Programming, para produzir o incremento do sistema que está sendo desenvolvido (Figura 2.13).

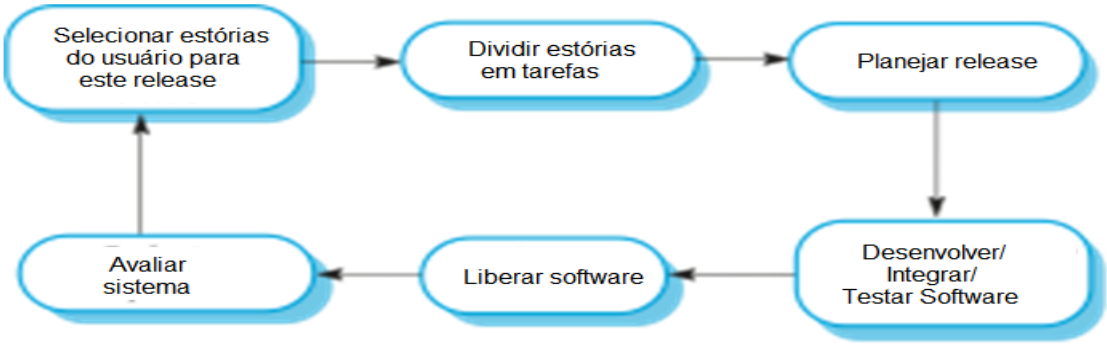


Figura 2.13 – Exemplo do ciclo de processos do XP (SOMMERVILLE, 2007)

2.5 UML

A UML (*Unified Modeling Language*) é hoje a base de estruturação para a fase de projeto de *software*. Como o próprio nome diz, a UML é uma metodologia, ou uma descrição das melhores práticas a serem seguidas durante a modelagem de um *software*. Segundo Bezerra (2007) a UML não se trata de uma linguagem de programação, e sim de uma notação, a qual tem como objetivo auxiliar durante todo o processo posterior ao desenvolvimento do *software* propriamente dito, influenciando no comportamento, estrutura e dinâmica dos processos.

Essa ferramenta permite a construção de diversos diagramas, desde os mais comuns como diagrama de caso de uso e o diagrama de classes até o mapeamento do modelo relacional do banco de dados. É importante descrevê-los e realizar o

levantamento de requisitos para melhor entendimento do projeto. Nesta seção serão apresentados alguns diagramas que dizem respeito ao funcionamento do software juntamente com o levantamento dos requisitos.

3 Estado da Arte

Com o surgimento de novas tecnologias de informação, hoje ficou fácil buscar soluções para atividades que ainda são feitas manualmente e gastam muitos recursos. No ambiente escolar e acadêmico, por exemplo, muitas instituições já substituem o uso do papel pela biometria, cartão eletrônico ou *bluetooth* no processo de verificação da assiduidade dos alunos. O uso da biometria através da impressão digital, como forma de efetivar a frequência dos alunos na instituição de ensino vem sendo adotada pelas Secretarias de Educação em estados e municípios. Segundo Morena Costa (2011), algumas instituições de ensino do Espírito Santo já adotaram o uso da digital como forma de efetivação da frequência em sala de aula, de tal modo que os estudantes têm 15 minutos para registrar a presença em um aparelho que fica logo na entrada da sala e que realiza o recolhimento das impressões digitais dos alunos.

Um outro recurso que vem sendo utilizado é o cartão eletrônico. De acordo com Brustolin (2010), a Secretaria de Educação do estado do Rio de Janeiro tem como objetivo adotar essa tecnologia nas escolas da rede estadual, de forma que o seu funcionamento ocorre do seguinte modo: o aluno ao chegar à portaria da instituição de ensino, deve passar o cartão no leitor da catraca eletrônica e esta através da rede sem fio (*Wi-fi*) envia a informação de confirmação da frequência juntamente com registro *online* do horário de entrada e saída do aluno para o sistema acadêmico.

Bem diferente dos dois tipos de tecnologia já mencionados é a utilização do *bluetooth*. Segundo Albiero (2017) a utilização dessa tecnologia ocorre de maneira simples, aonde o professor e os alunos devem acionar a função ligar *bluetooth* no dispositivo móvel e esperar até que os dispositivos sejam pareados e reconhecidos, após esse processo, os alunos já podem fazer o envio da confirmação da frequência para o aparelho do professor dentro da sala de aula.

Vários trabalhos correlatos foram implementados utilizando algumas das tecnologias descritas acima e são analisados a seguir. Vale ressaltar que estes trabalhos também podem ser classificados em uma das categorias apresentadas no capítulo anterior: Modelo Tradicional/Analógico, Modelo Digital e Modelo Digital Automatizado.

3.1 Sistema Móvel de Controle de Frequência

O aplicativo iPresence pretende modernizar o processo de inspeção das faltas dos alunos durante as aulas, através do uso de tecnologia móvel na forma de um aplicativo para *smartphones* e *tablet*, facilitando assim a tarefa de controlar a frequência dos alunos das disciplinas lecionadas pelos professores (Figura 3.1).

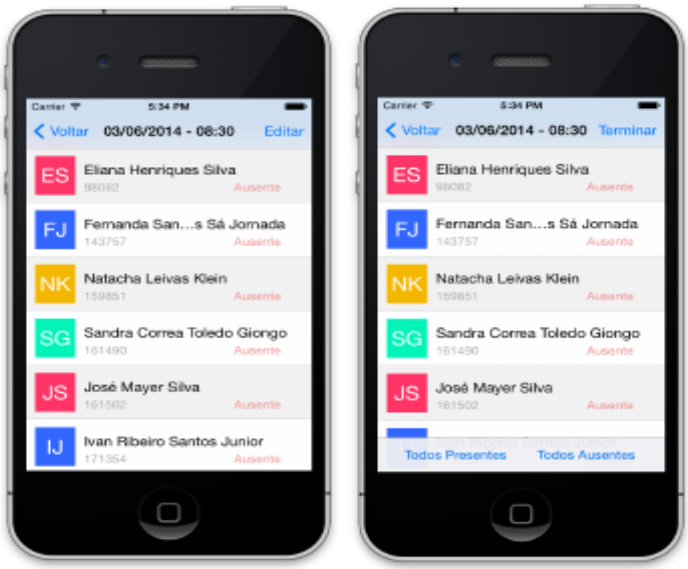


Figura 3.1 – Exemplo da interface do iPresence (HECK, 2013)

Um aspecto adicional do *software* é a integração existente com o sistema SAV (Sala de Aula Virtual), plataforma criada em 2011 pelo CPD (Centro de Processamento de Dados) da UFRGS (Universidade Federal do Rio Grande do Sul). Essa incorporação permite que, após realizada a chamada por parte do professor, o resultado esteja disponível no SAV, sem necessidade da interação humana.

O iPresence foi desenvolvido de forma nativa para a plataforma iOS e sua portabilidade somente atende à arquitetura alvo em que foi desenvolvido. No entanto, o

seu desempenho é muito melhor se comparado aos aplicativos Web. Isso porque ele tem acesso à maioria das funcionalidades do dispositivo, tais como: localização por GPS, câmera, sensores e sistema de arquivo.

A comunicação existente entre o iPresence e o SAV ocorre via um servidor Web, que interliga o aplicativo ao sistema de integração, de modo que o *software* possa atualizar o banco de dados do sistema SAV, com as demais chamadas feitas pelos professores (Figura 3.2).

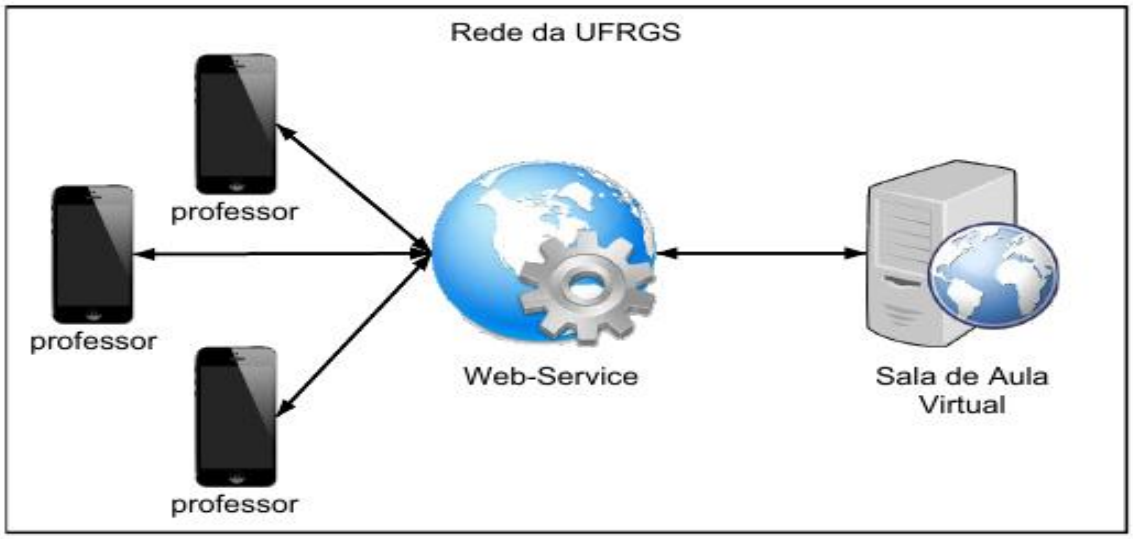


Figura 3.2 – Visão geral da integração ente o aplicativo iPresence e o SAV (HECK, 2013)

O banco de dados usado na modelagem do aplicativo iPresence foi o SQLite, em razão de ele dar suporte a dispositivos móveis iOS. O seu modelo de entidade relacionamento, apresenta as entidades herdadas do sistema de integração do sistema SAV e as relações que existem entre elas (Figura 3.3).

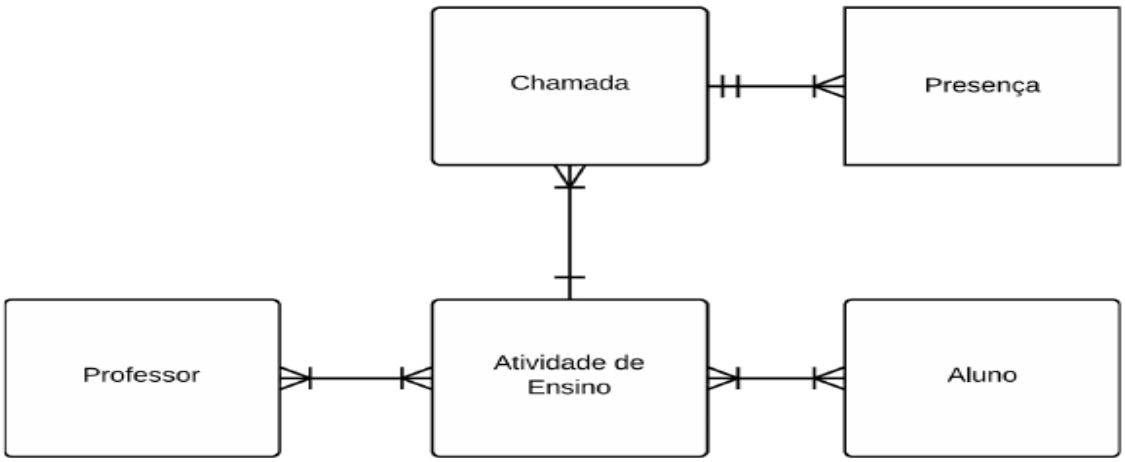


Figura 3.3 - Modelo Entidade Relacionamento do iPresence (HECK, 2013)

Para a organização do desenvolvimento do *software*, foi feito o uso do padrão de design chamado MVC (*Model View Controller*), no qual o conceito do aplicativo móvel foi dividido em três categorias de códigos, cada um agregando uma função diferente. A categoria View apresenta o código usado na apresentação dos dados; o Model contém o código que faz operação sobre os dados; e o Controller gerencia os dados. A pasta nomeada Storyboard contém os arquivos em que são implementadas as Views (Figura 3.4).

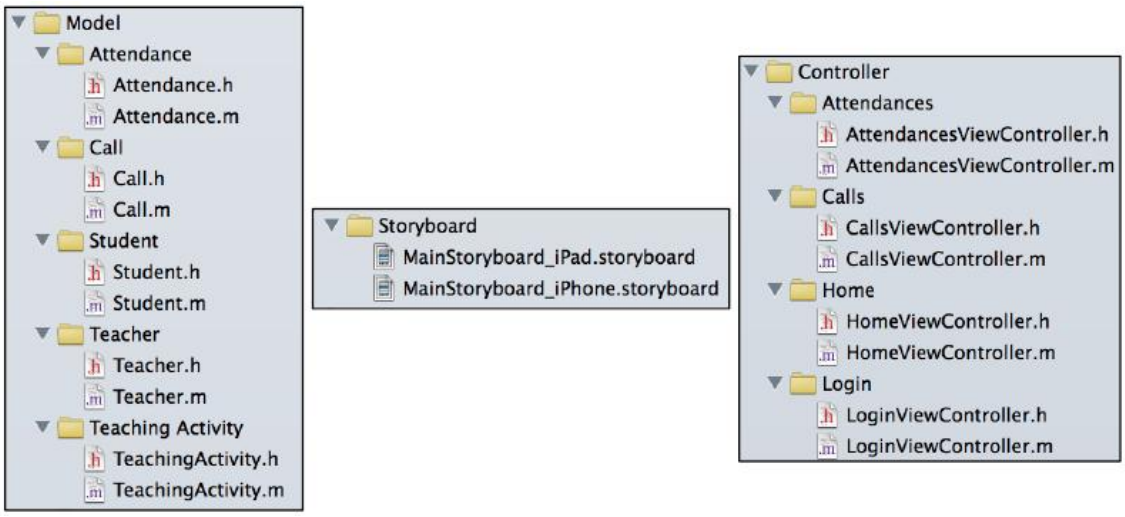
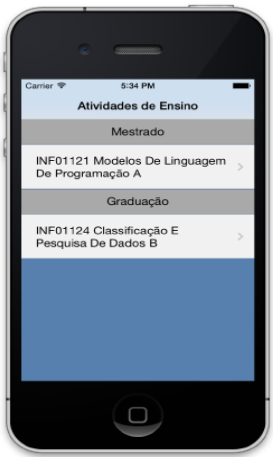


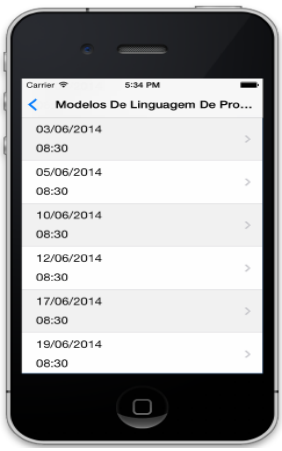
Figura 3.4 - Separação de conceitos usando MVC no aplicativo iPresence (HECK, 2013)

O *software* possui apenas quatro telas simples, cada uma com sua função. A primeira tela é a de *login* (Figura 3.5a); a segunda refere-se ás atividades de ensino (Figura 3.5b); a terceira apresenta a tela de chamadas feitas pelo professor (Figura 3.5c); e a quarta e última tela apresenta a lista de presença com informação de cada aluno em uma chamada (Figura 3.5d).

a) Tela de Login



b) Tela de Atividades de Ensino



c)Tela de Chamadas

d)Tela de Presenças

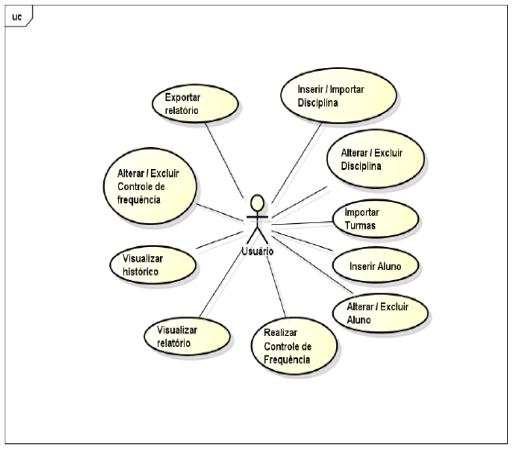
Figura 3.5 - Exemplo das quatro telas do aplicativo iPresence (HECK, 2013)

3.2 Controle de Frequência para Android

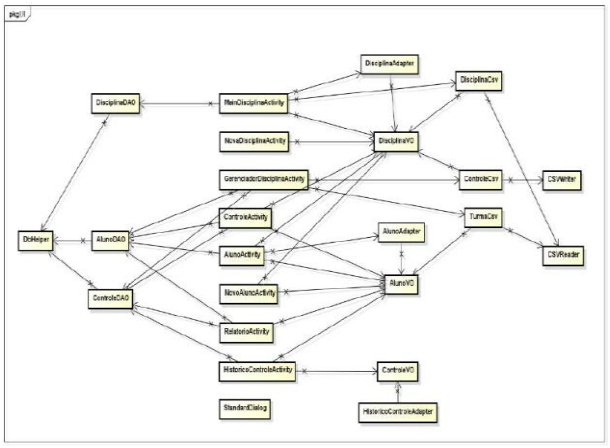
O controle de frequência para Android é um projeto desenvolvido por Vliet (2013), que consiste na criação de um aplicativo na plataforma Android, cujo objetivo é facilitar e agilizar o controle de frequência dos alunos durante as aulas, substituindo o uso tradicional do papel e o alto custo de tempo por algo mais eficaz, reduzindo as falhas na verificação das listas de presença, por meio da sincronização dos dados pelo dispositivo móvel (Figura 3.6).

Figura 3.6 – Exemplo da interface do *software* Controle de Frequência para Android (VLIET, 2013)

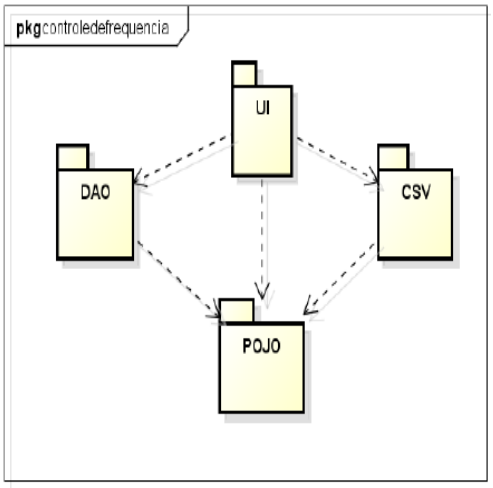
Antes deste sistema ser implementado, foi feito todo o estudo para definir qual o melhor processo de modelagem no desenvolvimento do aplicativo Controle de Frequência para Android. Assim, utilizando o padrão UML (*Unified Modeling Language*), foram criados os diagramas de caso de uso (Figura 3.7a), de classe (Figura 3.7b), de pacotes (Figura 3.7c) e de objetos (Figura 3.7d).



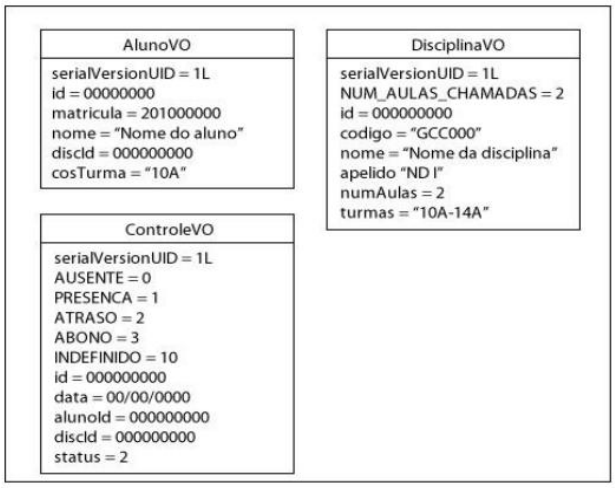
a)Diagrama de caso de uso



b)Diagrama de classe



c) Diagrama de pacotes



d) Diagrama de Objetos

Figura 3.7 – Diagramas do *software* Controle de Frequência para Android (VLIET, 2013)

O processo de engenharia de *software* utilizado no planejamento deste aplicativo foi o desenvolvimento iterativo, devido à facilidade de atualização e à possibilidade de alterações com o surgimento de novos requisitos.

O aplicativo Controle de Frequência para Android foi modelado no banco de dados SQLite, de modo que foram criadas as tabelas “alunos”, “disciplinas” e “controle_freq” no seu modelo de entidade e relacionamento. Cada uma dessas tabelas possui suas próprias características e se relacionam com as demais (Figura 3.8).

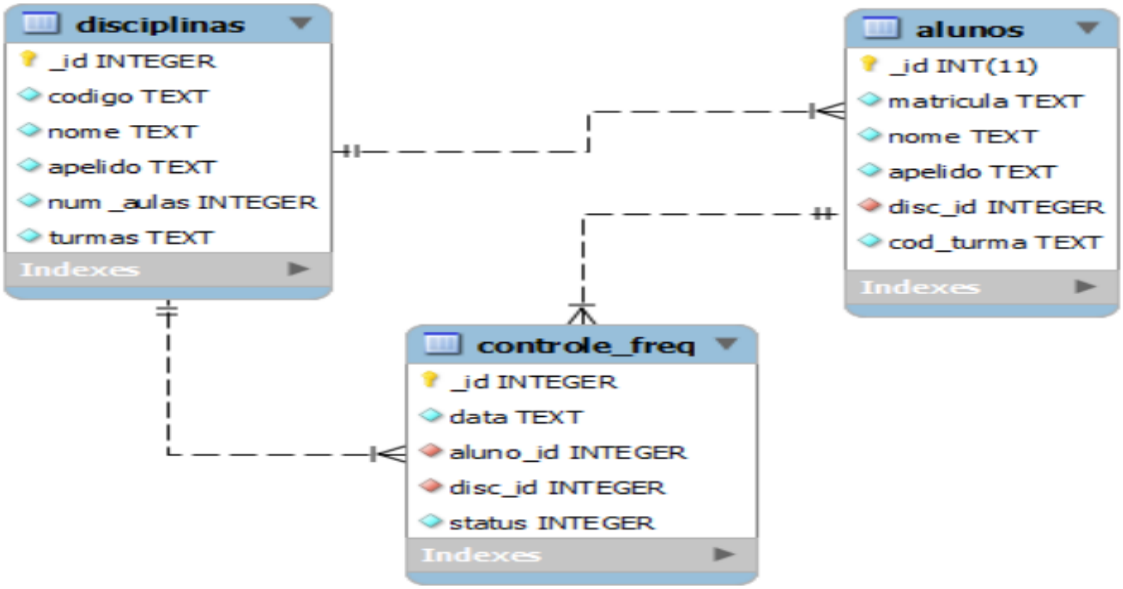


Figura 3.8 - Modelagem do banco de dados do aplicativo Controle de Frequência para Android (VLIET, 2013)

Finalizado o processo de realizar a frequência, que é feito pelos professores e alunos, são salvos arquivos no banco de dados do sistema de Controle de Frequência para Android, no formato CSV (*Comma Separeted Values*), de modo que eles são preenchidos seguindo um padrão, cuja primeira linha contém o código da disciplina; a segunda, o nome da disciplina e a turma separados por “-”; a terceira, o cabeçalho usado para informar o significado das linhas com nome e matrícula; e, logo em seguida, na quarta linha, seguindo a ordem da linha anterior, a lista de alunos que compõem a turma (Figura 3.9).

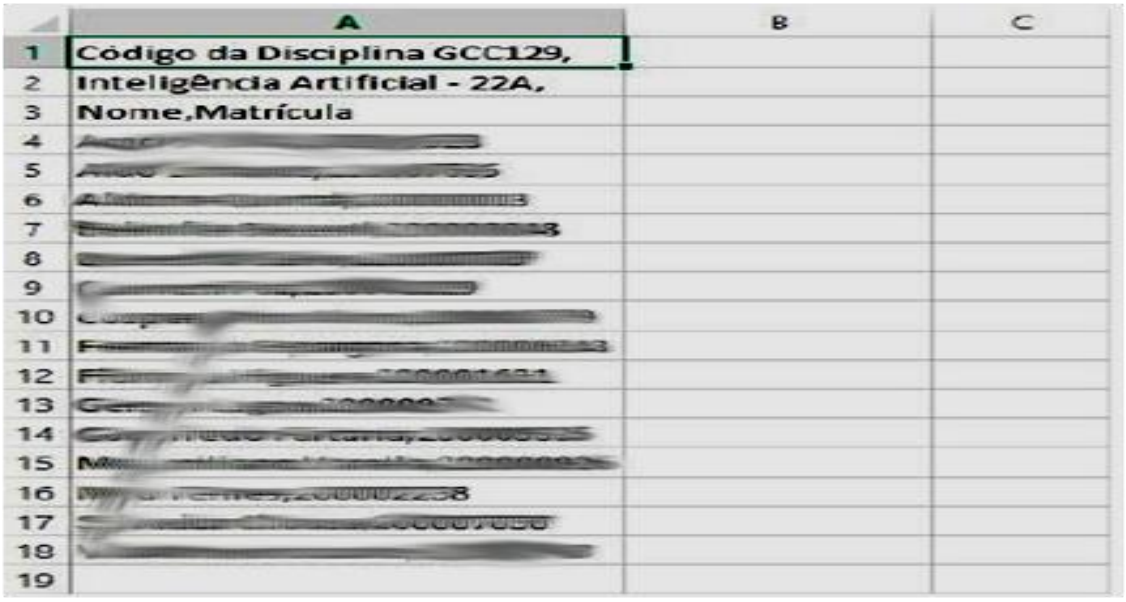
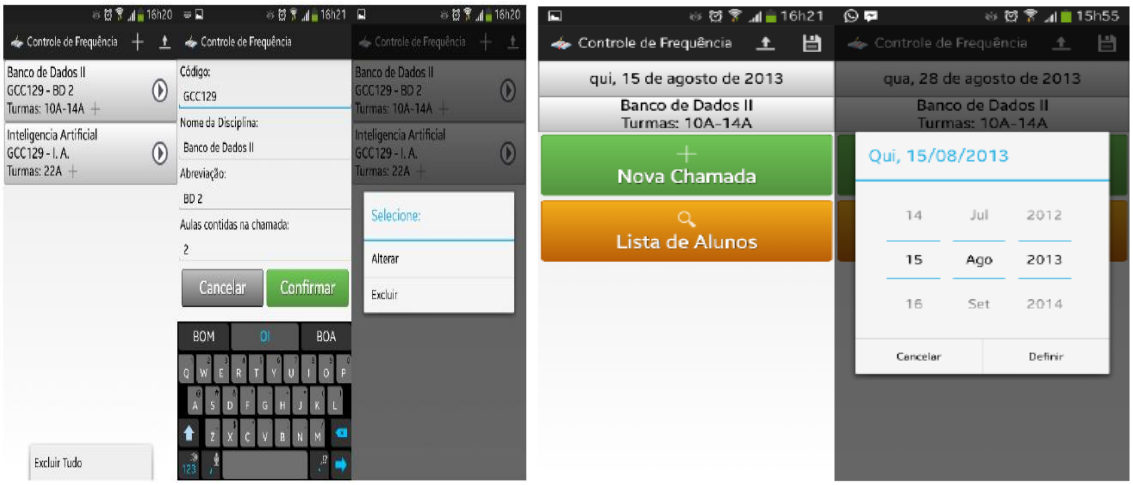


Figura 3.9 - Estrutura arquivo CSV utilizado pelo aplicativo Controle de Frequência para Android (VLIET, 2013)

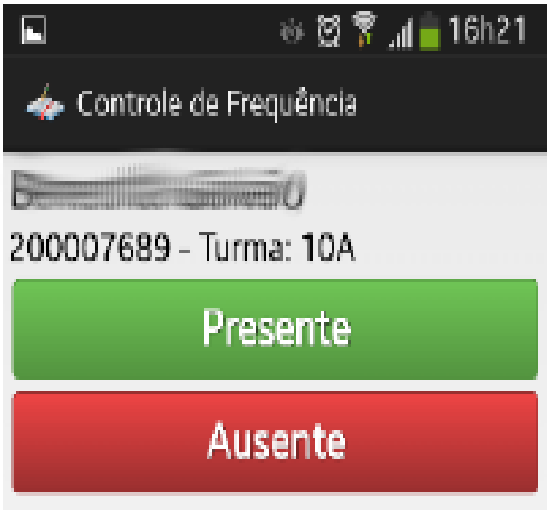
Vliet (2003) afirma que para desenvolver dispositivos móveis existe uma enorme preocupação com relação ao espaço e principalmente às informações que devem ou não ser apresentadas ao usuário em cada uma das interfaces de um aplicativo. Desta forma, o aplicativo Controle de Frequência para Android possui seis telas, cada uma com funções distintas uma das outras. Interface editar disciplina (Figura 3.10a), interface gerenciar disciplina (Figura 3.10b), interface controle de frequência (Figura 3.10c), interface lista de alunos (Figura 3.10d), interface relatório por aluno (Figura 3.10e) e a interface histórico detalhado (Figura 3.10f). As telas editar disciplina, gerenciar disciplina, controle de frequência e lista de alunos por se só são autoexplicativas. Diferente da tela de relatório por aluno, que permite o usuário visualizar a soma de todos os *status* armazenados para um único aluno até a data atual, visto que toda essa relação é mostrada de forma fácil, com o total de presenças, atrasos, abonos e ausências.

O uso dos botões Anterior e Próximo, ajudam o usuário a percorrer toda lista de alunos mais facilmente, mas se caso ele quiser ver mais detalhes do aluno, basta ele clicar no botão abaixo Ver Histórico. A última tela histórico detalhado, permite o usuário fazer alterações em um controle de frequência realizado numa determinada data. Isso é possível porque o *software* permite alterar o *status*, no qual um aluno pode passar de ausente para presente, ou de ausente para abono, um exemplo da mudança do *status* ausente para abono, ocorre quando o aluno consegue repor as aulas perdidas e o professor retira suas faltas.



a)Interface Editar Disciplinas

b) Interface Gerenciar Disciplinas



c) Interface Controle de Frequência

d)Interface Lista de Alunos

e)Interface Relatório por Aluno

f) Interface Histórico Detalhado

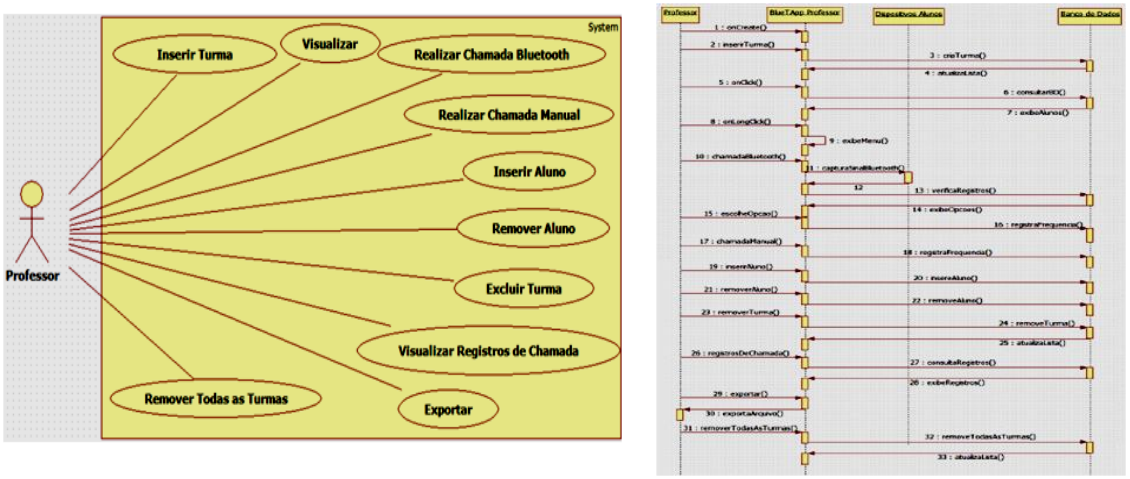
Figura 3.10 - Exemplo das seis telas do aplicativo Controle de Frequência para Android (VLIET, 2013)

3.3 Aplicativo Móvel para Registro Automático da Presença Acadêmica Via Bluetooth

BlueTApp é um aplicativo móvel para controle de frequência acadêmica, desenvolvido na plataforma Android, que apresenta uma evolução em relação ao modelo digital: ele não apenas gerencia o processo de contabilização dos dados, mas também busca fazer a aquisição destes dados automaticamente. A ideia central do

aplicativo é fazer a captura dos sinais, via *bluetooth*, presente nos dispositivos móveis dos alunos e, através das informações obtidas a partir desses sinais, verificar a presença ou ausência do aluno em sala de aula (ALBIERO, 2014).

O *software* apresenta duas versões: uma para o professor e outra para o aluno. Porém, como a versão para aluno é mais simples que à versão para professor, que realiza todo o gerenciamento do processo de efetuar a frequência de maneira automática, fez-se necessário construir a modelagem de diagramas de caso de uso (Figura 3.11a) e de sequência (Figura 3.11b), usando a ferramenta UML, para deixar mais claro quais seriam os requisitos da versão do professor.



a)Diagrama de caso de uso b)Diagrama de sequência

Figura 3.11 - Diagramas do Aplicativo Móvel para Registro Automático da Presença Acadêmica via Bluetooth (ALBIERO, 2014)

Todos os dados referentes à presença dos alunos estão contidos em um arquivo no formato CSV. Porém, antes de realizar a ativação do aplicativo em seu dispositivo móvel, o professor deve fazer o *download* do arquivo ".csv", tendo em vista que é somente através deste que pode ser feita toda a importação e o armazenamento dos dados referentes ao processo de efetivação da presença dos alunos (Figura 3.12).

nome,matrícula,data de inclusão na turma, curso,e-mail
ALAN PERALTA DUTRA,201222150,28/07/2014,Bacharelado em Sistemas de Informação,adutra@inf.ufsm.br
ANDRE LUIS GIACOMINI BRENDLER,201240234,28/07/2014,Bacharelado em Sistemas de Informação,andre_brendler@hotmail.com
CRISTOFER ROVIAN CLARO PEDROSO,2921130,25/08/2014,Curso de Engenharia de Computação,crisrcp@gmail.com
EMIR NATAL BRITO DE CAMPOS JUNIOR,2920918,28/07/2014,Bacharelado em Sistemas de Informação,emircampos@gmail.com
FERNANDO QUATRIN CAMPAGNOLO,201120061,11/08/2014,Bacharelado em Sistemas de Informação,fcampagnolo@inf.ufsm.br
HERSON FLACH NADALON,201040150,28/07/2014,Bacharelado em Sistemas de Informação,hnadalon@inf.ufsm.br
JUNIOR DE SOUZA,201020579,22/08/2014,Curso de Engenharia de Computação,junior.souza2102@gmail.com
MARCO ANTONIO DALCIN,201020318,25/08/2014,Curso de Engenharia de Computação,marco_dalcin@hotmail.com

Figura 3.12 - Formato do arquivo ".csv" importado pelo aplicativo BlueTApp (ALBIERO, 2014)

O BlueTApp mantém todos os arquivos referentes às frequências realizadas no seu banco de dados SQLite, já que essa base de dados é nativa da plataforma Android. Assim, os dados referentes à aplicação foram armazenados em duas tabelas. A tabela turmas (Tabela 3.1a), que possui as colunas Id, Nome, Matricula, Frequência, Bluetooth Address, Curso e Email, que são preenchidas a partir das informações importadas do arquivo .csv e a tabela registro de chamada (Tabela 3.1b).

Id	Nome	Matricula	Frequência	Bluetooth Address	Curso	Email
1	Fulano	123456	PPPPAA	0C:DF:A4:C0:9E:ED	Ciência da Computação	fulano @gmail .com

a) Tabela turmas

Id	Data	Presenças	Ementa	Posiçao Inicial	Posição Final
1	15/11/2014	28	Orientação a Objetos - Classes	2	3

b) Tabela registro de chamada

Tabela 3.1 - Tabelas que demonstram a organização das informações no banco de dados do aplicativo BlueTApp (ALBIERO, 2017)

Como o aplicativo possui duas versões, ou seja, uma versão para o aluno e outra para o professor, foi feita uma padronização no titulo das telas de modo que é possível identificar a interface pertencente a cada usuário. As telas que fazem parte da versão para docentes possui na parte superior do *layout* o nome "BlueTApp Professor" e as telas da versão para discentes "BlueTApp Aluno". A versão do aluno possui uma única tela inicial que solicita o preenchimento obrigatório do número de matricula, pois somente após o comprimento deste requisito o estudante possui a permissão para utilizar o adaptador *bluetooth* do aparelho, tornando-o visível para outros dispositivos (Figura 3.13a). As outras telas do aplicativo como: a tela inicial do aplicativo BlueTApp na versão para o professor (Figura 3.13b); a tela para inserir nova turma (Figura 3.13c), a tela que exibe a lista de alunos cadastrados (Figura 3.13d) e a que exibe o menu de opções (Figura 3.13e) todas pertencem a versão para professor e são autoexplicativas.

a) Tela inicial BlueTApp Aluno

b) Tela inicial BlueTApp Professor

c) Tela Inserir Nova Turma

d)Tela Lista de Alunos Cadastrados

e) Tela Menu de Opções

Figura 3.13 - Exemplo das cinco telas do Aplicativo Móvel para Registro Automático da Presença Acadêmica Via *Bluetooth* (ALBIERO, 2017)

4 PresentiaMobile

O presente trabalho consiste no desenvolvimento do PresentiaMobile, aplicativo Android para dispositivos móveis, que tem como finalidade fazer a aquisição automatizada e a contabilização dos dados referentes à frequência dos alunos. O *software* faz uso do armazenamento interno para armazenar os dados dos alunos e da comunicação *socket* via rede para conectar com o *desktop* do professor e enviar as informações necessárias para confirmar a presença do aluno.

4.1 Miniprojetos

Devido à complexidade da programação na arquitetura Android, o que requer um tempo a mais de estudo e conhecimento da sua sintaxe e ferramentas, o orientador Prof. Marlos Marques sugeriu uma estratégia de aprendizado baseado em miniprojetos. Cada miniprojeto abordou uma característica chave do PresentiaMobile, permitindo o aprendizado rápido e consistente das tecnologias necessárias para o desenvolvimento do projeto proposto. Estes miniprojetos são descritos a seguir.

4.1.1 Miniprojeto: *Hello World*

A criação do "*Hello World!*" (Olá Mundo!) é a primeira aplicação feita pelos programadores quando iniciam o desenvolvimento em qualquer linguagem de programação (Figura 4.1). O único componente gráfico usado na confecção da tela abaixo foi o TextView, de maneira bem simples, somente para imprimir na tela a expressão de boas vindas, "*Hello World!* ". A subclasse TextView é a primeira e a mais simples da classe View ela é utilizada para exibir um campo de texto na interface de aplicações e muitos programadores compara à com o JLabel da classe *swing* do Java, pois possui a mesma função.

Figura 4.1 – Tela "*Hello World,"* exemplo de aplicação, usando interface gráfica

4.1.2 Miniprojeto: *GUI*

Com o objetivo de conhecer mais a respeito do desenvolvimento de interface gráfica no Android, foi desenvolvida uma calculadora simples neste miniprojeto (Figura 4.2). A sua aplicação consiste no uso das classes EditText, utilizadas para exibir no *display* todos os números da calculadora e da String, denominada *operation*, que trabalha em conjunto com os métodos criados para cada operação matemática. Além dessas classes, foi usada também a classe Button, para criar os botões na interface da calculadora.

A classe EditText é uma subclasse da TextView e geralmente é utilizada para que o programador possa colocar na interface gráfica de aplicações informações em um campo de texto como texto normal ou números como foi feita na aplicação calculadora simples. A classe String trabalha com uma cadeia de caracteres, assim a criação da String *operation* foi feita com o objetivo de organizar o projeto e proporcionar uma melhor visualização dos métodos que realizam as quatro operações matemáticas no código-fonte do miniprojeto, pois ela identifica cada método com o seu respectivo sinal "+" , "-" , "*" e "/".

Figura 4.2 – Tela da aplicação calculadora simples, exemplo com uso da interface gráfica Android

4.1.3 Miniprojeto: *Sockets*

Antes de iniciar o estudo da utilização de *sockets* na arquitetura Android, utilizando a ferramenta Android Studio, foi conveniente criar uma aplicação simples do *socket* Cliente/Servidor na linguagem Java para *desktop* apenas como objeto de estudo e revisão do assunto de comunicação em redes de computadores. Vale ressaltar que a lógica e o código utilizado na estrutura do cliente foi amplamente aproveitado na implementação do PresentiaMobile, pois ambos fazem utilizados na classe Socket, enquanto que a estrutura do servidor serviu para os testes de comunicação e foi na sua maioria implementada pelo professor orientador.

4.1.3.1 Comunicação com sockets em Java para Desktop

Na criação do *socket* cliente a classe do projeto foi nomeada ClienteChamada. Ela estabelece um canal de comunicação entre o cliente e o servidor, através de um atributo chamado Socket, composto pelo IP e pela porta do servidor. As classes DataInputStream e DataOutputStream foram utilizadas para determinar uma lógica de leitura e escrita das mensagens enviadas e recebidas como também permitir o envio e o recebimento da confirmação da mensagem enviada. No miniprojeto ClienteChamada foi criado uma GUI simples (Figura 4.3), cujo único componente usado foi o Button da classe *swing*, para inserir na interface os botões "Enviar", que, ao ser pressionado, estabelece a conexão com o *socket* servidor e envia uma mensagem. E o "Sair" criado para encerrar a execução da aplicação.

Figura 4.3 - Interface Gráfica do miniprojeto ClienteChamada

O servidor utiliza um *socket* para abrir uma porta de conexão e aguardar até ser estabelecida uma conexão, foi usada a classe ServerSocket e, em seguida, o método *accept*, quando de fato ela for estabelecida. Como no aplicativo PresentiaMobile, vários alunos irão tentar estabelecer conexões com o servidor para efetivar sua presença, *threads* e semáforos foram utilizados para gerenciar o fluxo de comunicação. Na Figura 4.4 é mostrada a interface gráfica para teste do servidor, utilizando apenas um componente JTextArea da classe *swing*, para desenhar uma caixa de texto com o objetivo de imprimir as mensagens enviadas pelo cliente.

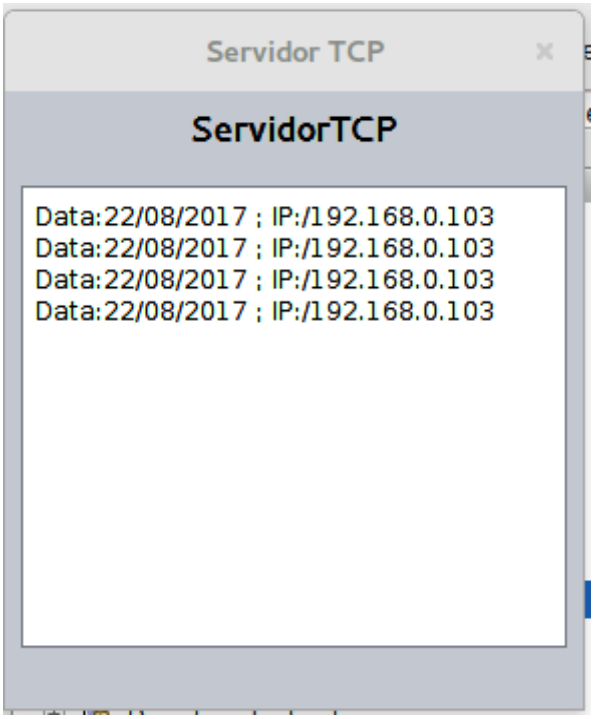


Figura 4.4 – Interface gráfica do servidor

4.1.3.2 Comunicação com sockets em Java para Android

Após revisar como é estabelecida uma comunicação em rede via *socket* através da aplicação Java para *desktop*, foi desenvolvido um miniprojeto semelhante para a arquitetura Android, mediante um miniprojeto simulando a arquitetura Cliente/Servidor (Figura 4.5). No desenvolvimento desta aplicação, foram usadas as classes EditText, TextView e a Socket. A primeira classe é responsável por criar o campo que recebe a mensagem do cliente e envia para o servidor; já a segunda, serve para imprimir a resposta de confirmação do recebimento da mensagem pelo servidor na tela do cliente; e

a última, em conjunto com as classes DataInputStream e DataOutputStream, é usada na abertura e fechamento da conexão e no envio dos dados, da mesma forma que é feita em uma aplicação Java normal.

A criação deste miniprojeto exigiu uma atenção maior se comparado os anteriores, pois foi a partir desta aplicação que obteve-se a princípio uma noção de como seria implementar uma conexão via *socket* na arquitetura Android, visto que isso é o requisito fundamental para o funcionamento do *software* PresentiaMobile desenvolvido neste trabalho. Foi feito também o uso das ferramentas e classes da interface gráfica do Android observados nos miniprojetos anteriores, de tal modo que os campos criados devem estar envolvidos de certa forma com envio e recebimento de dados.

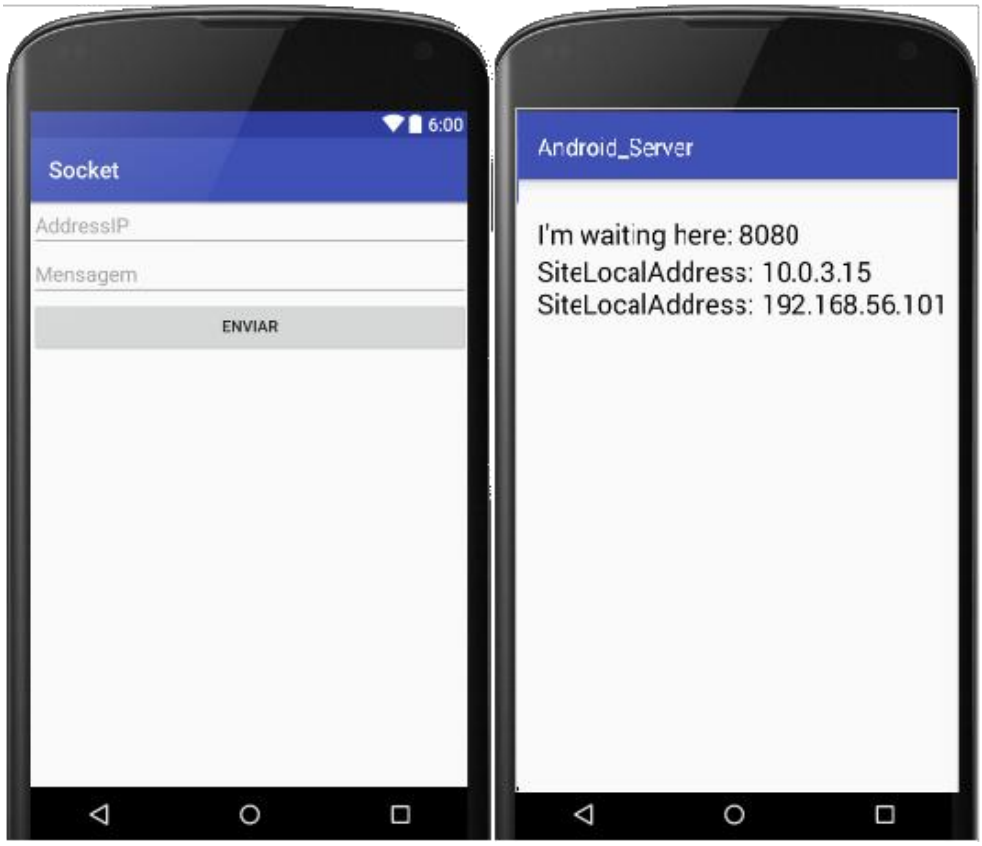


Figura 4.5 – Tela da aplicação *sock*et, exemplo com uso da comunicação via Socket Android

Desta forma, a lógica aplicada nesta aplicação é basicamente a mesma do miniprojeto realizado anteriormente, porém existe uma diferença entre aplicação *socket* para *desktop* e a *mobile* presente na parte do código que o servidor estabelece a conexão com o cliente, visto que para estabelecer esta conexão no miniprojeto ClienteChamada

basta apenas informar a classe ServerSocket no código do servidor o endereço da porta local do cliente, diferente da aplicação *socket* no Android que além de também informar esse endereço da porta local do cliente é necessário criar um método String privado chamado *getIpAddress*, responsável por obter o endereço IP das interfaces de rede local e *wifi* do dispositivo móvel (Figura 4.6).

```
private String getIpAddress() {
    String ip = "";
    try {
        Enumeration<NetworkInterface> enumNetworkInterfaces = NetworkInterface
                .getNetworkInterfaces();
        while (enumNetworkInterfaces.hasMoreElements()) {
            NetworkInterface networkInterface = enumNetworkInterfaces
                    .nextElement();
            Enumeration<InetAddress> enumInetAddress = networkInterface
                    .getInetAddresses();
            while (enumInetAddress.hasMoreElements()) {
                InetAddress inetAddress = enumInetAddress.nextElement();

                if (inetAddress.isSiteLocalAddress()) {
                    ip += "SiteLocalAddress: "
                            + inetAddress.getHostAddress() + "\n";
                }

            }

        }
```

Figura 4.6 – Código para obter o endereço IP do cliente na comunicação via Socket Android

4.1.4 Miniprojeto: SQLite

O miniprojeto Agenda Contatos (Figura 4.7) foi desenvolvido durante os estudos sobre o uso do banco de dados SQLite, nativo da plataforma Android. Apesar de não ter feito o uso deste recurso no *software* proposto neste trabalho, esta aplicação serviu como aprendizado para futuras aplicações.

Figura 4.7 – Tela da aplicação agenda contatos, exemplo usando o banco de dados Android

A aplicação é composta por duas telas. A primeira possui um botão adicionar, situado à direita do campo de texto, responsável por permitir o cadastro dos contatos na agenda. A tela seguinte apresenta os campos nome, telefone, e-mail e endereço, que devem ser preenchidos e depois salvos na opção que aparece na aba superior. Após fazer todo esse procedimento, o contato que foi salvo aparece na primeira tela, e o usuário pode verificar se os dados foram gravados corretamente.

As classes utilizadas no desenvolvimento do código foram o EditText, para criar os campos nome, telefone, e-mail e endereço; o ListView, para exibir os contatos salvos; o Spinner, para criar uma aba de opções que especifica o tipo de cada um dos dados na tela de cadastro dos contatos; e o pacote *android.database.sqlite*, responsável por armazenar os dados de cada contato no banco de dados da aplicação.

4.2 Desenvolvimento do PresentiaMobile

O PresentiaMobile é um aplicativo móvel desenvolvido de forma nativa para a plataforma Android através da IDE Android Studio. O objetivo do aplicativo é adquirir os dados referentes à frequência dos alunos, de maneira eficiente, e realizar também a contagem desses dados. A linguagem de programação utilizada foi o Java. Ele não faz uso do banco de dados, mas o substitui pelo armazenamento interno do Android, visto que, na primeira tela do aplicativo, os alunos devem preencher os seguintes dados

(matricula, nome e sobrenome), que são armazenados em um arquivo de texto na memória interna do programa. O *software* apresenta quatro telas simples que foram projetadas com o objetivo de direcionar o usuário o mais rápido possível para a tela de confirmação da presença dos alunos. A seguir, serão detalhados alguns aspectos relacionados ao desenvolvimento do projeto.

4.2.1 Análise de Requisitos

Antes de iniciar de fato o desenvolvimento do aplicativo PresentiaMobile, foi feito inicialmente o levantamento de requisitos durante as reuniões entre os autores do projeto. Através da análise de cada um deles, foi possível verificar quais realmente atendiam as expectativas do projeto. Esta etapa foi necessária para coletar dados para criar um sistema de acordo com as necessidades dos usuários (professores e alunos), pois somente após possuir toda a documentação detalhada, com as funcionalidades do sistema, foi possível dar início ao projeto e o planejamento do desenvolvimento do aplicativo com as atividades a serem realizadas.

Os requisitos que aparecem nas seções 4.2.1.1 e 4.2.1.2 fornecem as diretrizes básicas para o desenvolvimento do projeto, eles foram separados em dois tipos distintos, relacionados e com diferentes características essenciais do *software*. Primeiramente, são listados os requisitos funcionais, os quais são relativos às funcionalidades esperadas pelos usuários. Posteriormente são listados requisitos não funcionais, os quais determinam características essenciais para produção do *software*.

4.2.1.1 Requisitos Funcionais

RF01 - O professor usuário do sistema deve iniciar a frequência.

RF02 - O aluno usuário do sistema deve realizar o seu cadastro.

RF03 - O aluno usuário do sistema deve confirmar sua frequência.

RF04 - O professor usuário do sistema deve finalizar a frequência.

RF05 - O professor usuário do sistema deve visualizar os dados da frequência.

RF06 - O sistema de integração deve gerar um relatório com todos os dados da frequência do aluno.

RF07 - O professor usuário deve enviar um relatório de frequência, apenas para os alunos que estão propícios a serem reprovados por falta.

RF08 - O professor usuário do sistema deve imprimir o documento com todos os dados da frequência do aluno.

4.2.1.2 Requisitos Não Funcionais

RNF01 - O sistema somente funciona em dispositivos móveis com Sistema Operacional Android.

RNF02 – Aplicativo com suporte as versões 4.4 a 7.0 do Android.

Desempenho – Almeja-se que o aplicativo funcione da maneira mais ágil possível e ofereça o melhor desempenho em todas as versões do Android ao qual o aplicativo oferece suporte.

Usabilidade – O aplicativo deve atender a todos requisitos mínimos descritos anteriormente, oferecendo a maior usabilidade possível aos usuários, atendendo suas expectativas.

Modularidade – O *software* deve permitir que novos módulos sejam adicionados permitindo novas funcionalidades ao sistema de acordo com novas necessidades e atendendo a outros requisitos estabelecidos pelos usuários.

4.2.2 Diagrama de Caso de Uso

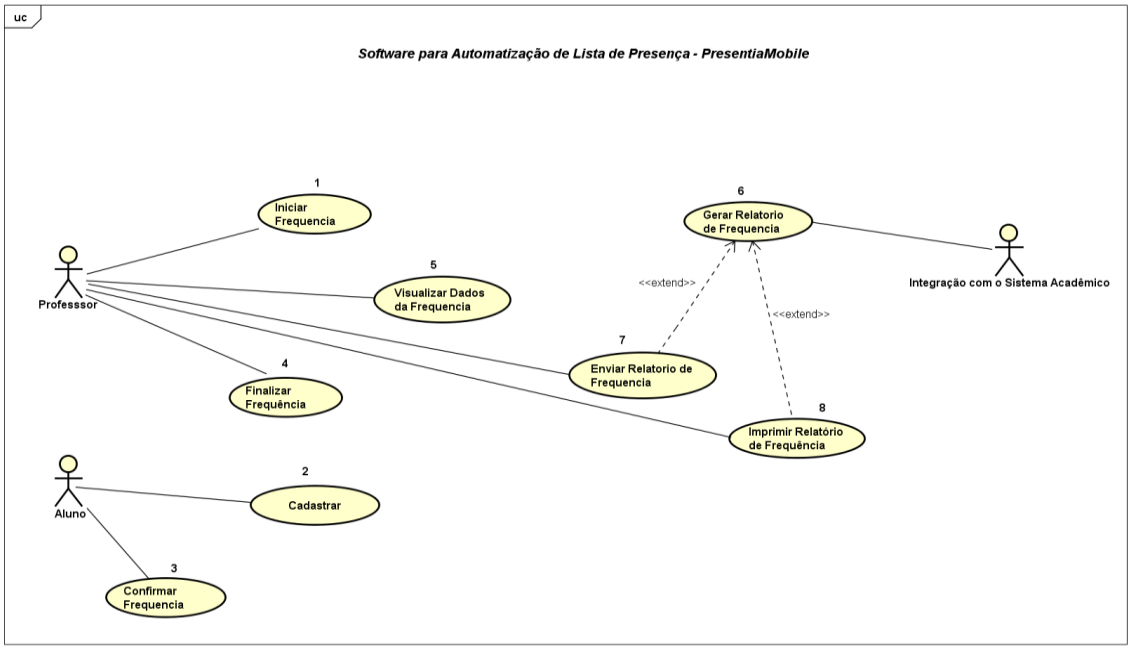


Figura 4.8 – Diagrama de casos de uso do PresentiaMobile

- **ATORES DO SISTEMA**

Nome	
Professor	Representa o usuário que possui acesso ao aplicativo PresentiaMobile. O professor é quem concede iniciar o processo de efetivação da frequência, finalizar a frequência e tem acesso a opção "Visualizar Dados da Frequência".
Aluno	Representa o usuário que também possui acesso ao aplicativo PresentiaMobile. O aluno tem acesso somente as opções "Cadastrar" e "Confirmar Frequência".
Sistema Acadêmico	Representa um sistema vinculado a uma universidade ou instituição de ensino, que embora não faça parte do desenvolvimento do projeto, o aplicativo interage com este para enviar os dados do aplicativo e gerar um relatório de frequência.

- **CASOS DE USO DO SISTEMA**

1. Iniciar Frequência
• DESCRIÇÃO DO CASO DE USO
1) DESCRIÇÃO
O caso de uso inicializa uma nova frequência com o uso do aplicativo, para verificar a assiduidade dos alunos em sala de aula.
2) ATORES
O ator principal é o Professor. Não existe ator secundário.
3) PRÉ-CONDIÇÕES
O agente professor deve verificar antes se todos os alunos presentes na classe possuem dispositivos móveis com sistema operacional Android e se o aplicativo PresentiaMobile está instalado. Caso exista alunos que não possuem o dispositivo, o

professor deve anotar a presença destes e posteriormente registrar no sistema de integração.
4) FLUXO DE EVENTOS
Esse caso de uso se inicia quando o agente Professor realizar a operação correspondente a de inicializar uma nova frequência durante as aulas.
4.1) FLUXO PRINCIPAL
1. O agente Professor avisa aos alunos o exato momento de realizar a validação da frequência.
2. O agente Professor espera os alunos inicializarem o aplicativo nos celulares.
3. O agente Professor inicializa o servidor no seu *desktop* e ordena que os alunos efetivem a sua frequência em sala de aula.
5) PÓS-CONDIÇÕES
Não existe pós-condições.

2. Cadastrar
• DESCRIÇÃO DO CASO DE USO
1) DESCRIÇÃO
O caso de uso permite que o agente Aluno se cadastre no aplicativo, informando os dados (matricula, nome e sobrenome).
2) ATORES
O ator principal é o Aluno. Não existe ator secundário.
3) PRÉ-CONDIÇÕES
O agente aluno deve, através do seu dispositivo móvel, fazer o *download* do aplicativo PresentiaMobile e em seguida instalá-lo.
4) FLUXO DE EVENTOS
Esse caso de uso se inicia quando o agente Aluno realizar a operação correspondente a de inicializar o aplicativo.
4.1) FLUXO PRINCIPAL
1. O agente Aluno inicializa o aplicativo em seu *smartphone*.
2. O agente Aluno preenche os campos com os dados referentes à matricula, nome e sobrenome.

3. O agente Aluno confirma o cadastro pressionando o botão “OK”.
4. O aplicativo confirma o cadastro e salva esses dados na memória interna do aplicativo.
5) PÓS-CONDIÇÕES
Não existe pós-condições.

3. Confirmar Frequência
• DESCRIÇÃO DO CASO DE USO
1) DESCRIÇÃO
O caso de uso permite que o agente Aluno confirme sua presença no aplicativo, e envie uma mensagem APDU com os seguintes dados confirmando sua presença.
2) ATORES
O ator principal é o Aluno. Não existe ator secundário
3) PRÉ-CONDIÇÕES
O agente aluno deve através do seu dispositivo móvel executar o aplicativo PresentiaMobile.
4) FLUXO DE EVENTOS
Esse caso de uso se inicia quando o agente Aluno realiza a operação correspondente de confirmar sua presença na aula.
4.1) FLUXO PRINCIPAL
1. O agente Aluno inicializa o aplicativo em seu dispositivo móvel.
2. O agente Aluno verifica se o professor deu aviso para os discentes confirmarem sua presença.
3. O agente Aluno confirma o cadastro pressionando o botão “PRESENTE”.
4. O aplicativo confirma o cadastro e salva esses dados no arquivo de texto no *desktop* do professor.
5) PÓS-CONDIÇÕES
Esses dados também são salvos no banco de dados do Sistema de Integração com o Sistema Acadêmico, cujo o aplicativo está integrado.

4. Finalizar Frequência
• DESCRIÇÃO DO CASO DE USO

1) DESCRIÇÃO
O caso de uso permite que o agente Professor encerre o processo de efetivação da frequência que foi inicializado durante a aula.
2) ATORES
O ator principal é o Professor. Não existe ator secundário.
3) PRÉ-CONDIÇÕES
O agente Professor deve verificar se todos alunos já confirmaram a presença.
4) FLUXO DE EVENTOS
Esse caso de uso se inicia quando o agente Professor realiza a operação correspondente a de encerrar a confirmação da frequência durante a aula.
4.1) FLUXO PRINCIPAL
1. O agente Professor verifica se todos os alunos já confirmaram sua presença.
2. Se sim, o agente Professor encerra o servidor em seu *desktop*.
3. Se não, ele espera até o restante dos alunos confirme sua presença e em seguida encerra-o.
5) PÓS-CONDIÇÕES
Não existe pós-condições.

5. Visualizar Dados da Frequência
• DESCRIÇÃO DO CASO DE USO
1) DESCRIÇÃO
O caso de uso permite que o agente Professor visualize todos os dados referentes a frequência dos alunos contidos no arquivo de texto.
2) ATORES
O ator principal é o Professor. Não existe ator secundário.
3) PRÉ-CONDIÇÕES
O agente Professor deve verificar se existe um arquivo com o nome "PresentiaArquivo.txt" dentro da pasta pessoal do seu *desktop*.
4) FLUXO DE EVENTOS
Esse caso de uso se inicia quando o agente Professor realiza a operação

correspondente a de visualizar todos os dados referentes a frequência dos alunos, contidos no arquivo de texto.
4.1) FLUXO PRINCIPAL
1. O agente Professor verifica se existe um arquivo chamado "PresentiaArquivo.txt" na pasta pessoal.
2. O agente Professor abre o arquivo em um *software* apropriado.
3. O agente Professor visualiza todos os dados referentes a frequência dos alunos na ordem em que foram salvos.
5) PÓS-CONDIÇÕES
Não existe pós-condições.

6. Gerar Relatório de Frequência
• DESCRIÇÃO DO CASO DE USO
1) DESCRIÇÃO
O caso de uso permite que o agente Sistema de Integração com o Sistema Acadêmico, analise todos os dados referentes a frequência dos alunos contidos no banco de dados do sistema e gere um documento com todos esses dados.
2) ATORES
O ator principal é o Sistema de Integração. Não existe ator secundário.
3) PRÉ-CONDIÇÕES
O agente Sistema de Integração com o Sistema Acadêmico deve estabelecer uma comunicação com o aplicativo e vice-versa.
4) FLUXO DE EVENTOS
Esse caso de uso se inicia quando o agente Sistema de Integração com o Sistema Acadêmico, realiza a operação correspondente a de gerar um documento com os dados referentes a frequência dos alunos, contidos no seu banco de dados.
4.1) FLUXO PRINCIPAL
1. O agente Sistema de Integração com o Sistema Acadêmico, faz uma busca por todos os dados referentes a frequência dos alunos, contidos no seu banco de dados.
2. O agente Sistema de Integração com Sistema Acadêmico cria um documento com todos os dados que diz respeito a frequência do aluno durante todo período letivo.

5) PÓS-CONDIÇÕES
Não existe pós-condições.

7. Enviar Relatório de Frequência
• DESCRIÇÃO DO CASO DE USO
1) DESCRIÇÃO
O caso de uso permite que o agente Professor envie um documento com todos os dados referentes a frequência dos alunos, contendo a quantidade de presenças e faltas, inclusive com um alerta se o aluno já atingiu o total de 20% ou 25% do total de faltas.
2) ATORES
O ator principal é o Professor. Não existe ator secundário.
3) PRÉ-CONDIÇÕES
O agente Professor deve ter acesso ao sistema de integração que mantém comunicação com o aplicativo.
4) FLUXO DE EVENTOS
Esse caso de uso se inicia quando o agente Professor realiza a operação correspondente a de enviar um documento com os dados referentes a frequência, apenas para os alunos que estão aptos a ser reprovados por falta.
4.1) FLUXO PRINCIPAL
1. O agente Professor acessa o sistema de integração.
2. O agente Professor verifica quais alunos estão propício a ser reprovado por falta.
3. O agente Professor envia apenas para esses discentes, um arquivo com os dados referentes a sua frequência e um alerta.
5) PÓS-CONDIÇÕES
O documento será enviado no formato PDF, pois é a extensão gerada pelo sistema de integração.

8. Imprimir Relatório de Frequência
• DESCRIÇÃO DO CASO DE USO
1) DESCRIÇÃO
O caso de uso permite que o agente Professor imprima o documento com todos os

dados referentes a frequência dos alunos.
2) ATORES
O ator principal é o Professor.
Não existe ator secundário.
3) PRÉ-CONDIÇÕES
O agente Professor deve ter acesso ao sistema de integração que mantém comunicação com o aplicativo.
4) FLUXO DE EVENTOS
Esse caso de uso se inicia quando o agente Professor realiza a operação correspondente a de imprimir um documento com os dados referentes a frequência dos alunos, armazenado no aplicativo e enviado para o sistema de integração.
4.1) FLUXO PRINCIPAL
1. O agente Professor acessa o sistema de integração.
2. O agente Professor seleciona o curso, o semestre, a disciplina e informa a data desejada.
3. O agente Professor solicita a impressão do documento.
5) PÓS-CONDIÇÕES
Não existe pós-condições.

4.2.3 Diagrama de Classes

O diagrama de classes é utilizado na construção do sistema desde o nível de análise até o nível de especificação. Segundo Bezerra (2007), de todos diagramas da UML, esse é o mais importante e o mais rico em termos de notação. Na Figura 4.9 logo abaixo é possível observar o diagrama de classes do aplicativo PresentiaMobile, formado pelas principais classes que compõem todo sistema de automatização da lista de frequência, as funcionalidades de cada uma delas e o seu relacionamento. Ele é importante pelo fato de demonstrar claramente onde cada funcionalidade do sistema está definida e as possíveis alterações que podem ser feitas, seja na retirada ou adição de funções.

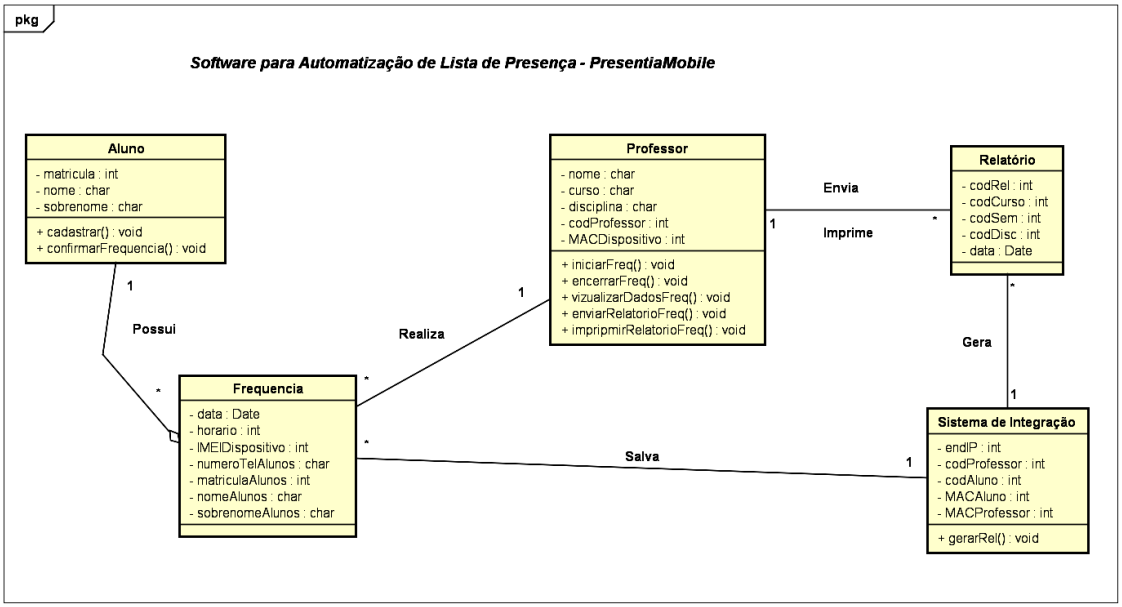


Figura 4.9 – Diagrama de classes do PresentiaMobile

4.2.4 Diagrama de Atividade

Esse diagrama é conhecido também como diagrama de estados, pelo fato de ser possível representar o grau de complexidade de uma funcionalidade específica, através da variação dos estados de acordo com a entrada fornecida. Ele é importante, pelo fato de demonstrar o comportamento de um caso de uso que sofre variações com o tempo ou na presença de diferentes valores de entrada. As Figuras 4.10, 4.11 e 4.12 apresentam respectivamente, o diagrama de atividade de alguns dos principais casos de uso do aplicativo PresentiaMobile.

No diagrama de atividade do caso de uso Cadastrar (Figura 4.10), essa atividade é realizada pelos alunos usuários do aplicativo, cuja função se inicia a partir do primeiro acesso ao *software*, que logo em seguida solicita que o usuário preencha os campos (matrícula, nome e sobrenome) na tela de cadastro e em seguida confirma o cadastro no aplicativo.

Já o caso de uso Confirmar Frequência (Figura 4.11), também realizado pelos alunos usuários, é um dos mais importantes e tem início quando o aluno inicializa o aplicativo, verifica se a frequência foi aberta, ou seja, o professor avisou a classe que pode ser feita a efetivação da presença e os discentes confirmam sua presença. Porém, caso o docente ainda não tenha aberto o servidor em seu *desktop* e o aluno tentar confirmar sua presença, ele vai perceber que não será possível e deve esperar um pouco.

O último diagrama de atividade, refere-se a dois casos de uso o Enviar e Imprimir Relatório de Frequência (Figura 4.12). Esses por sua vez dependem que o docente acesse o sistema integrado ao aplicativo e solicite a criação de um relatório com todos os dados da frequência, pois somente após ter gerado esse documento é possível optar pelo envio ou impressão do relatório de frequência.

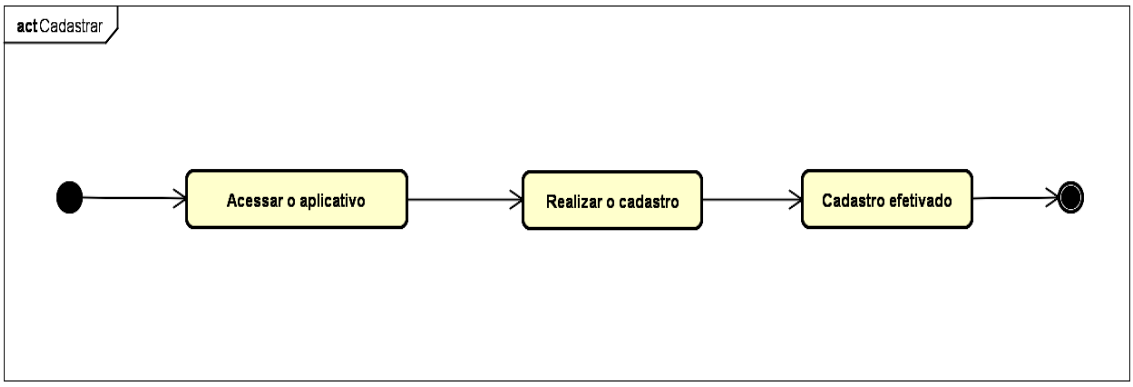


Figura 4.10 – Diagrama de atividade do caso de uso Cadastrar

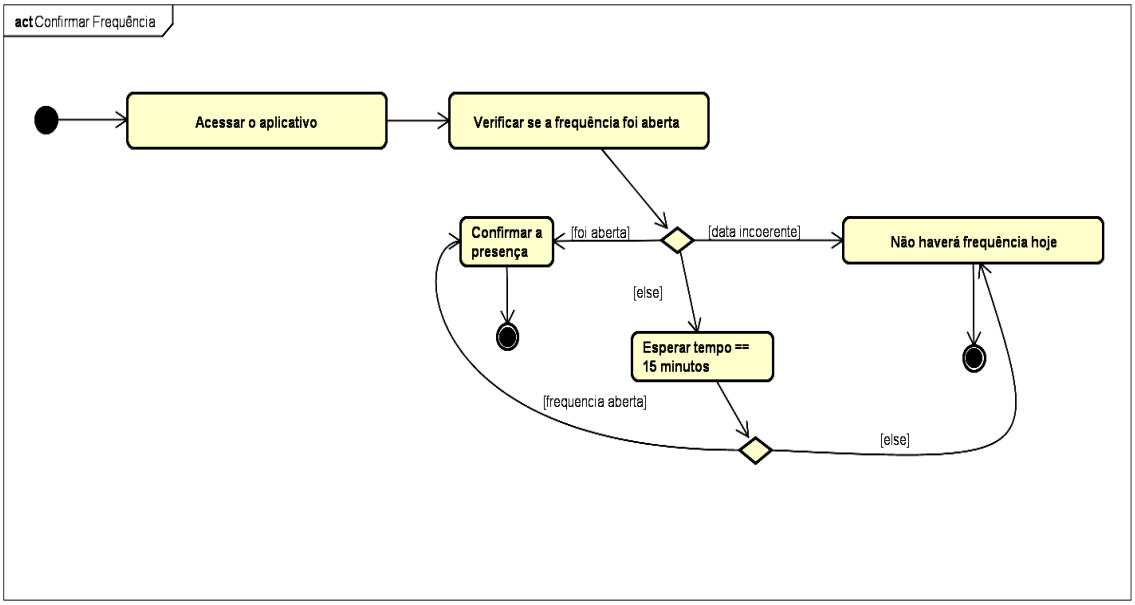


Figura 4.11 – Diagrama de atividade do caso de uso Confirmar Frequência

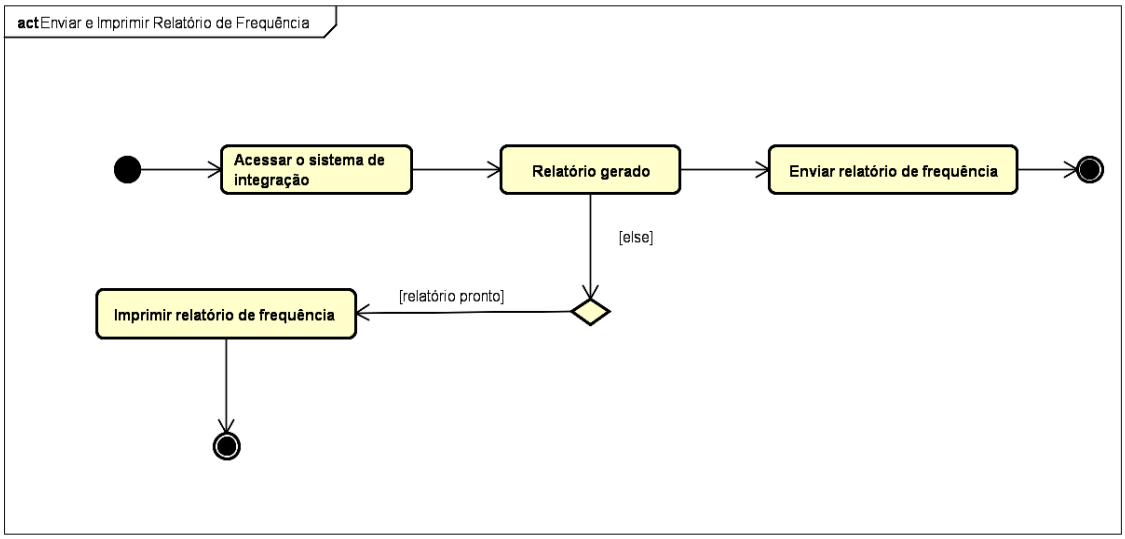


Figura 4.12 – Diagrama de atividade dos casos de uso Enviar e Imprimir Relatório de Frequência

4.2.5 Telas do PresentiaMobile

Nesta seção serão mostradas as funcionalidades e características presente em cada uma das quatro telas simples do aplicativo e o modo como ocorre a troca de mensagens no aplicativo PresentiaMobile.

A tela de inicialização do aplicativo, ou Splash Screen, como é conhecida em Android (Figura 4.13), foi criada através de um *handler* que exibe o logotipo do *software* subindo durante 3 segundos e ao final desse tempo, exibe a próxima tela.

Figura 4.13 – Tela de Abertura

A tela de cadastro dos alunos é a segunda a ser exibida. Nela os alunos devem preencher os campos matrícula, nome e sobrenome e, em seguida, pressionar o botão

“OK” para confirmar o cadastro e permitir a criação do arquivo de texto na memória interna do aplicativo (Figura 4.14).

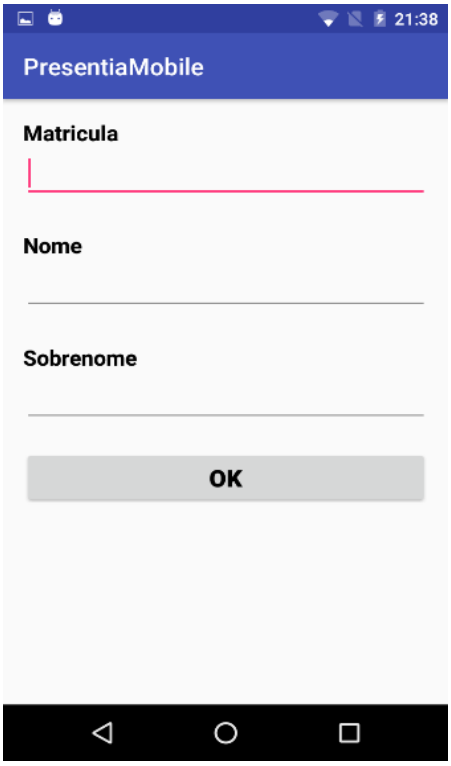


Figura 4.14 – Tela de Cadastro

Com objetivo de não permitir que qualquer informação fosse colocada nos campos matricula, nome e sobrenome e logo após fossem salvos, foi aplicado o processo de verificação de persistência dos dados, contanto que cada campo estabeleceu uma regra de como deve ser preenchido, cujo a única regra em comum entre os três campos foi não aceitar o cadastro com qualquer um dos campos em branco. Para preencher o espaço da matrícula deve ser inserido 9 dígitos, pois esse requisito não aceita ser preenchido com uma quantidade de dígitos inferior ou superior a isso. Os campos nome e sobrenome não aceitam todos os caracteres, apenas letras maiúsculas, minúsculas e acentuação (Figura 4.15).

```
//VALIDAR OS CAMPOS E CRIAR ARQUIVO PRESENTIACONFIG.TXT
public void Salvar_onClick() {

    //VERIFICAÇÃO CAMPO MATRICULA
    if (editMatricula.getText().toString().equals("")) {
        MessageBox.Alert(this, this.getString(R.string.matricula_obrigatoria));
        editMatricula.requestFocus();

    } else if (editMatricula.getText().length() < 9) {
        MessageBox.Alert(this, this.getString(R.string.matricula_incorreta));
        editMatricula.requestFocus();

    }

    //VERIFICAÇÃO CAMPO NOME
    else if (editNome.getText().toString().equals("")) {
        MessageBox.Alert(this, this.getString(R.string.nome_obrigatorio));
        editNome.requestFocus();

    }

    //VERIFICAÇÃO CAMPO SOBRENOME
    else if (editSobrenome.getText().toString().equals("")) {
        MessageBox.Alert(this, this.getString(R.string.sobrenome_obrigatorio));
        editSobrenome.requestFocus();
```

Figura 4.15 – Código da validação da persistência dos campos na tela de cadastro

Na tela seguinte, o usuário aluno, ao pressionar o botão “Presente” confirma sua presença, e envia uma mensagem APDU, composta pela Data, Hora, IMEI, Número do Celular e a mensagem complementar formada por matrícula, nome e sobrenome do aluno, do seu dispositivo para o *desktop* do professor, via classe DataOutputStream (Figura 4.16).

Figura 4.16 – Tela da Interface Aluno

A tela de confirmação da presença é a última tela do aplicativo. Ela aparece somente quando o processo de efetivação da presença foi bem sucedido, ou seja, se na interface do professor no *desktop* chegar uma mensagem APDU, confirmando a presença do aluno. Assim, o professor reenvia outra mensagem com o *feedback* de confirmação da presença, como demonstra a Figura 4.17.

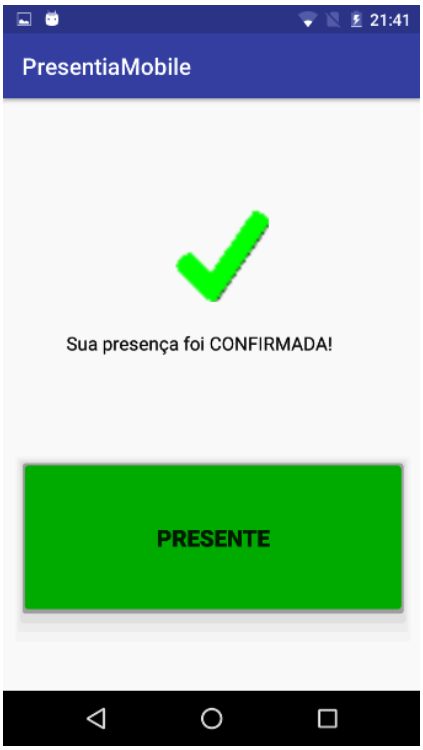


Figura 4.17 – Tela de Confirmação da Presença

O armazenamento de dados no aplicativo tem início, a partir do momento que o aluno confirma o preenchimento dos campos referentes ao número de matrícula, nome e sobrenome, e estas informações são salvas e armazenadas no armazenamento interno *(Internal Storage*) do Android, evitando que o aluno preencha os referidos campos toda vez que usar o aplicativo.

A troca de mensagens na aplicação desenvolvida neste trabalho, ocorre da maneira comum como é conhecida na literatura de redes de computadores, onde o cliente e o servidor fazem uso da classe DataOutputStream e DataInputStream para enviar e receber mensagens e criar um fluxo de entrada e saída. Esta parte do código foi baseada do miniprojeto correspondente, conforme descrito na seção 4.1.3.2 e está ilustrada na Figura 4.18.

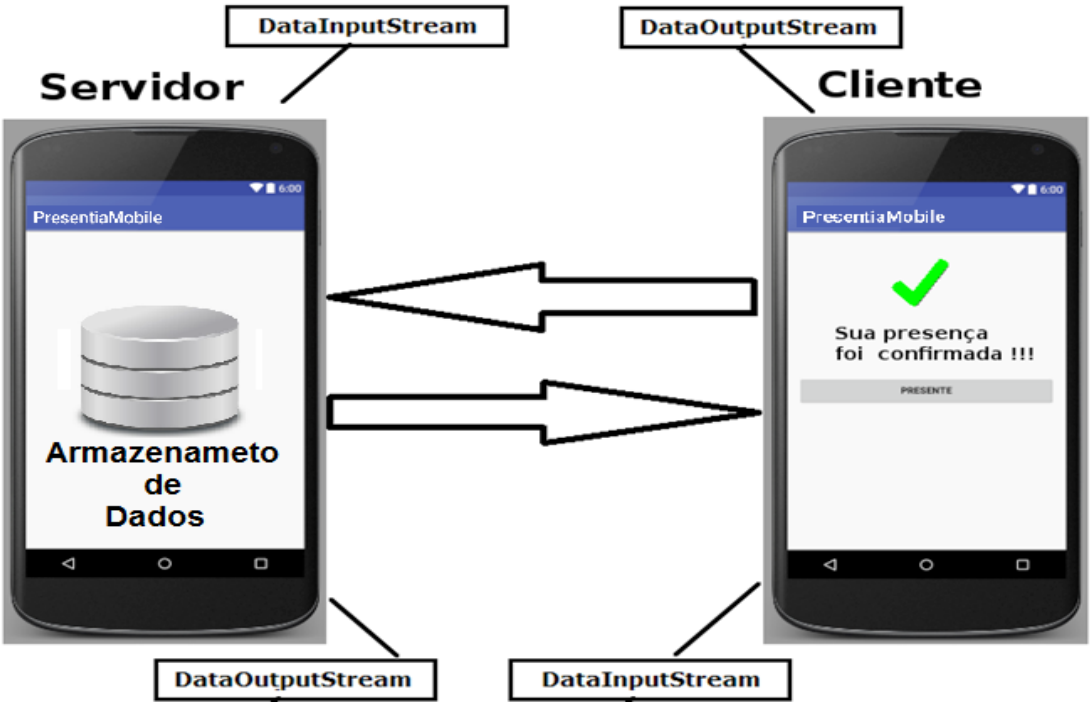


Figura 4.18 – Funcionamento do sistema

.

4.2.6 Considerações sobre a versão Android utilizada

O aplicativo foi implementado na versão 6.0 (API 23) do Android denominada Marshmallow, esta por sua vez apresenta funcionalidades diferentes se comparado as versões anteriores, no qual os pedidos de permissões eram todos aceitos durante a instalação, a partir desta versão em diante o usuário terá o livre arbítrio para aceitar ou negar as permissões que forem solicitadas em tempo de execução do aplicativo toda vez que um recurso for usado, como acessar a agenda, localização ou gravar arquivos no cartão de memória.

Essa iniciativa busca refinar mais ainda a segurança e privacidade no uso dos aplicativos, de modo que os desenvolvedores terão um pouco mais de complexidade na implementação dos aplicativos, pois precisam criar um método que verifique se a permissão de uso do recurso foi concedida ou não e mostrar o alerta ao usuário (LECHETA, 2016).

Para facilitar e alertar o usuário sobre o tipo de recurso que o aplicativo precisa utilizar, o Android classificou as permissões em dois grupos: normais e perigosas. O grupo das permissões normais acessa recursos que não traz riscos a privacidade do usuário e o sistema automaticamente concede a permissão ao aplicativo (Figura 4.19a). Já o grupo das permissões perigosas, abrange áreas que traz riscos a privacidade do

usuário, pois o aplicativo precisa de dados ou recursos que envolvem informações pessoais do usuário ou que podem afetar os dados armazenados no dispositivo móvel (Figura 4.19b).

PERMISSÕES NORMAIS
ACCESS_LOCATION_EXTRA_COMMANDS
ACCESS_NETWORK_STATE
ACCESS_NOTIFICATION_POLICY
ACCESS_WIFI_STATE
BLUETOOTH
BLUETOOTH_ADMIN
BROADCAST_STICKY
CHANGE_NETWORK_STATE
CHANGE_WIFI_MULTICAST_STATE
CHANGE_WIFI_STATE
DISABLE_KEYGUARD
EXPAND_STATUS_BAR
FLASHLIGHT
GET_PACKAGE_SIZE
INTERNET
KILL_BACKGROUND_PROCESSES
MODIFY_AUDIO_SETTINGS
NFC
READ_SYNC_SETTINGS
READ_SYNC_STATS
RECEIVE_BOOT_COMPLETED
REORDER_TASKS
REQUEST_INSTALL_PACKAGES

a)Grupo de Permissões Normais

PERMISSÕES PERIGOSAS	
GRUPO DE PERMISSÕES	PERMISSÕES
CALENDAR	READ_CALENDAR WRITE_CALENDAR
CAMERA	CAMERA
CONTACTS	READ_CONTACTS WRITE_CONTACTS GET_ACCOUNTS
LOCATION	ACCESS_FINE_LOCATION ACCESS_COARSE_LOCATION
MICROPHONE	RECORD_AUDIO
PHONE	READ_PHONE_STATE CALL_PHONE READ_CALL_LOG WRITE_CALL_LOG ADD_VOICEMAIL USE_SIP PROCESS_OUTGOING_CALLS
SENSORS	BODY_SENSORS
SMS	SEND_SMS RECEIVE_SMS READ_SMS RECEIVE_WAP_PUSH RECEIVE_MMS
STORAGE	READ_EXTERNAL_STORAGE WRITE_EXTERNAL_STORAGE

b) Grupo de Permissões Perigosas

Figura 4.19 – Tipos de Permissões do Android

5 Avaliação

Com o propósito de verificar o público atingido e o nível de aceitação do *software* proposto por este trabalho, foi solicitado aos professores da Universidade Estadual do Sudoeste da Bahia (UESB) que respondessem a um breve questionário durante o período de 17/02/2017 a 19/02/2018. O questionário tem como título "Pesquisa sobre o processo de execução da lista de presença em sala de aula", foi criado no Google Forms e contou com a participação de 30 professores do curso de Ciência da Computação. A seguir, os resultados serão discutidos.

Mais da metade dos professores que responderam ao questionário possuem dispositivos móveis com Android, ou seja, dos 30 professores que responderam 18 possuem um aparelho celular com sistema operacional Android e apenas 12 não possuem (Figura 5.1).

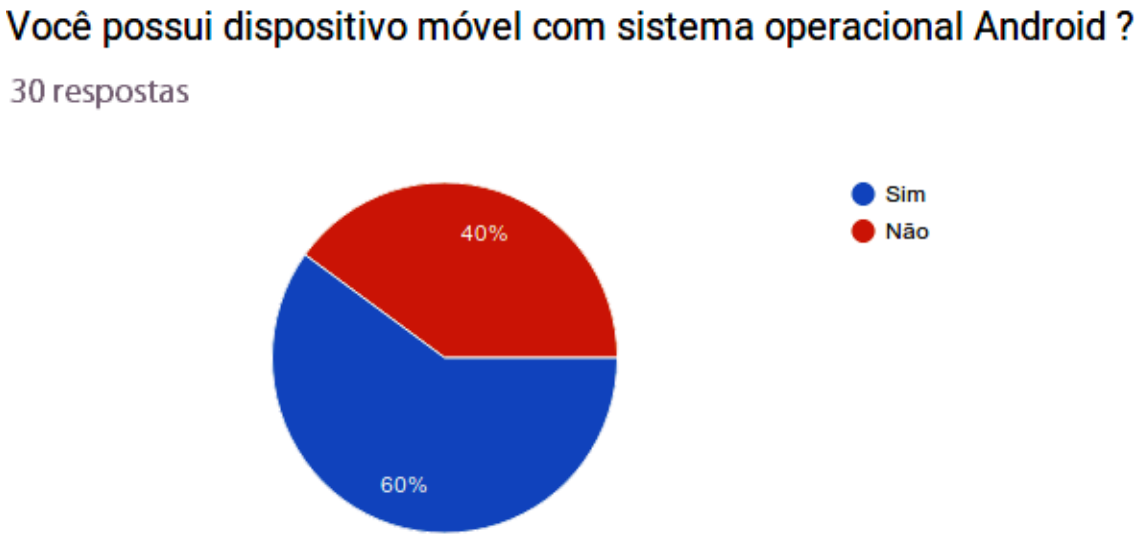


Figura 5.1 – Quantidade de dispositivos móveis com Android

Como pode ser observado a maioria dos professores da UESB realizam o processo de verificação da frequência dos alunos durante as aulas, ou seja, dos 30 professores que responderam ao questionário apenas 3 não realizam o obrigatório processo de verificação da frequência dos alunos durante as aulas (Figura 5.2).

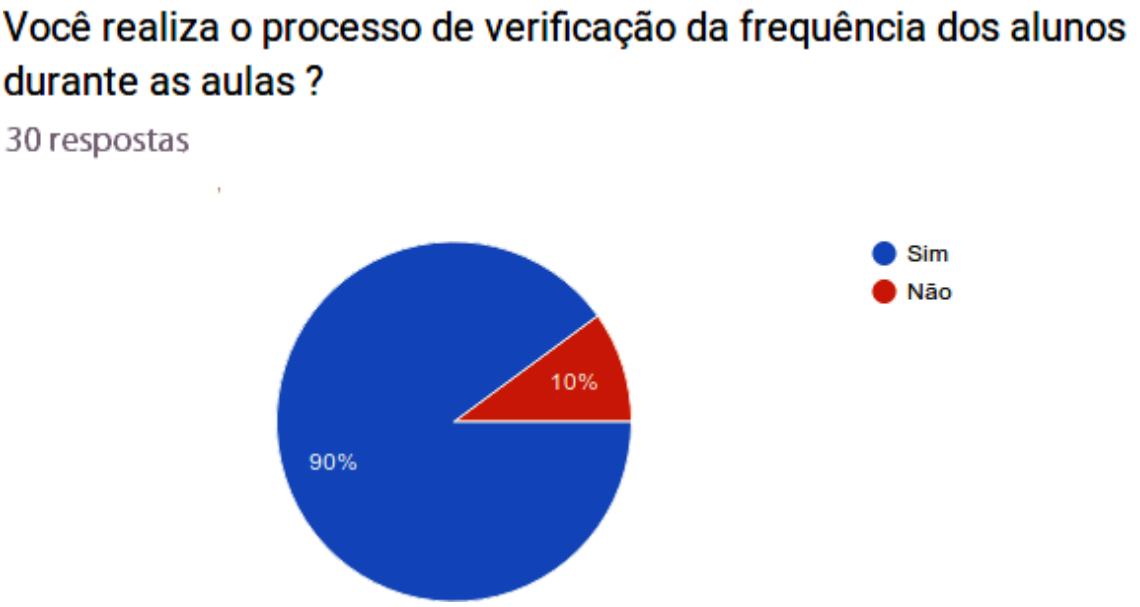


Figura 5.2 – Quantidade de professores que realizam a verificação da frequência dos alunos

Dos 30 professores que responderam ao questionário e expuseram sua opinião sobre o modelo tradicional de verificação da frequência dos alunos, possui opiniões diferentes, visto que 6 professores responderam que o processo é excelente; outros 6 que o processo é bom; e 6 que o processo é ruim. Estes somam 60% da pesquisa, restando

40%, equivalentes a um grupo de 12 professores que disseram que o modelo é regular; esta, portanto, foi a opção mais votada (Figura 5.3).

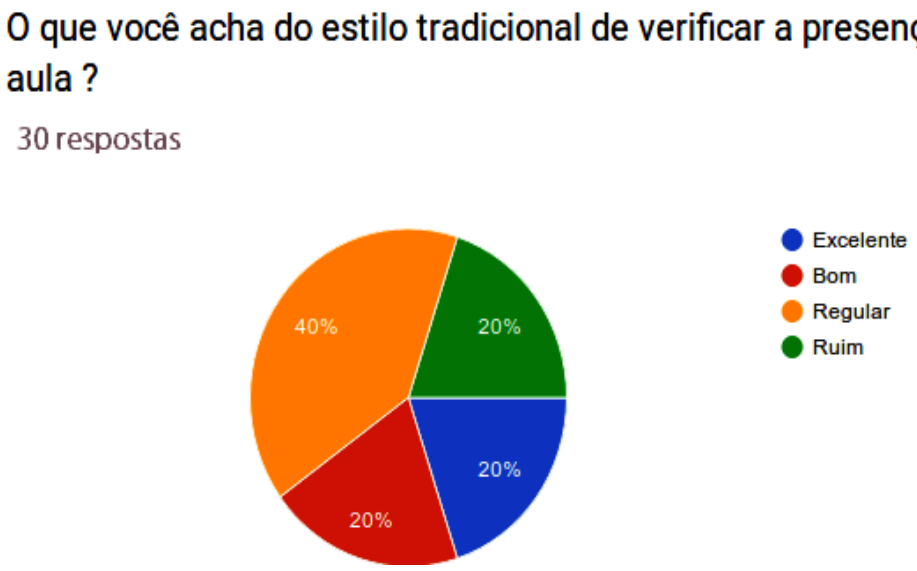


Figura 5.3 – Opinião dos professores sobre o modelo tradicional de verificação da frequência dos alunos

Por fim, todos os professores que responderam ao questionário acreditam que o desenvolvimento de um aplicativo móvel que venha a substituir o processo de aquisição e contabilização feito manualmente pelos docentes, certamente, seria bem vindo nas instituições de ensino, não só por causa dos possíveis benefícios, mas pelo fato de solucionar as deficiências do antigo processo que torna tudo muito dispendioso (Figura 5.4).

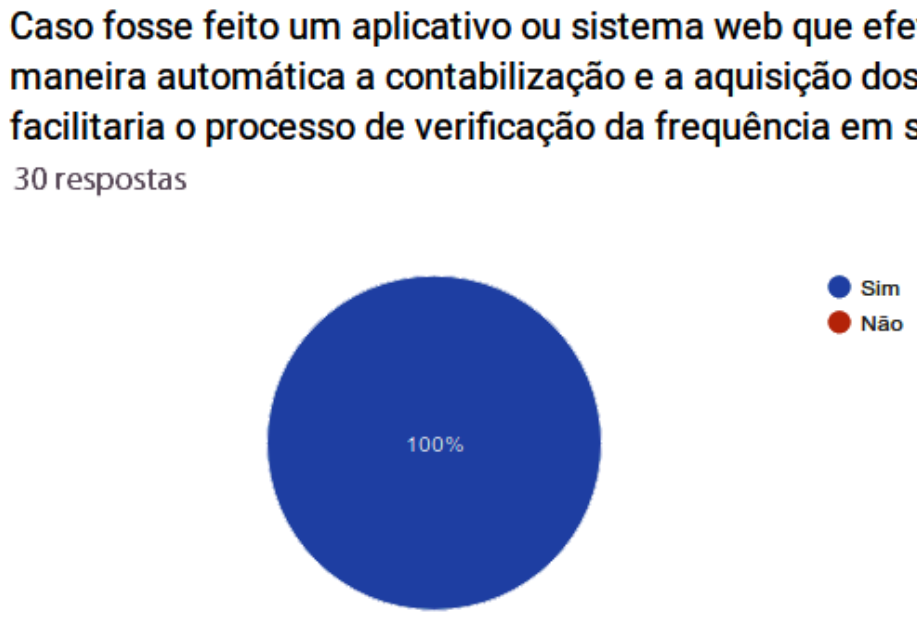


Figura 5.4 – Opinião dos professores sobre o desenvolvimento de um *software* que realize o processo de verificação da frequência dos alunos

6 Conclusões e Trabalhos Futuros

A popularização do uso de dispositivos móveis e, consequentemente, o desenvolvimento de aplicações móveis tem sido amplamente explorado, pois o uso desses aplicativos tem facilitado muito as atividades do dia a dia das pessoas. Em vista disso, fez se necessário à aplicação desta tecnologia nas habituais práticas educacionais realizadas pelos professores que na maioria das vezes, são efetivadas manualmente, o que gera um trabalho enorme, se tornando uma tarefa cansativa, além de gerar desperdício de papel.

Desta forma, o desenvolvimento deste trabalho teve como objetivo a criação de um aplicativo *mobile* que propõe otimizar o processo de efetivação do controle de frequência de modo diferente da maioria dos aplicativos que realizam este tipo de tarefa, pois ele realiza a aquisição automatizada de seus usuários de maneira satisfatória, como também a contabilização das presenças/faltas. As ferramentas disponíveis para desenvolvimento de aplicações móveis na plataforma Android são poderosas, mas extremamente complexas devido o seu amplo conjunto de funcionalidades e códigos que são específicos do Android, o que torna difícil a compreensão inicial, sendo necessário um certo tempo de estudo para conseguir compreende-los. Assim só foi possível desenvolver o aplicativo proposto em tempo hábil, mesmo partindo de um nível de conhecimento zero sobre a plataforma Android e suas ferramentas, graças à metodologia de miniprojetos.

Por fim, embora houvesse uma percepção da demanda por este tipo de *software*, o que motivou o projeto de pesquisa do orientador Prof. Marlos Marques, isso foi efetivamente constatado através do questionário realizado neste trabalho junto aos docentes do curso de Ciência da Computação da Universidade Estadual do Sudoeste da Bahia, na qual ficou comprovado que 100% dos professores são favoráveis ao desenvolvimento de um aplicativo que facilite o processo de verificação da frequência em sala de aula.

6.1 Trabalhos Futuros

Este projeto tem como futuras pretensões substituir o modo de armazenamento interno no aplicativo pelo uso do banco de dados SQLite (nativo da plataforma Android), de modo que possa melhorar e tornar mais seguro a performance do armazenamento dos dados gerados e obtidos pela aplicação. Realizar uma sequência de testes simulando uma situação real de dia letivo, em que todos os alunos estão presentes na classe, com o objetivo de explorar o limite do *software* através de realização de teste exaustivo, na qual todos discentes confirmam a frequência ao mesmo tempo e realizar também o teste de tempo de resposta, comparando o tempo entre a mensagem enviada pelo aluno e o recebimento desta pelo professor. Integrar o aplicativo a um sistema *online* (e em tempo real), que permita uma melhor comunicação e troca de conteúdo entre alunos e professores, a exemplo da plataforma SAV (Sala de Aula Virtual), que já foi citada neste trabalho e do SAGRES, portal acadêmico utilizado pela Universidade Estadual do Sudoeste da Bahia. E por fim desenvolver uma versão para o sistema operacional iOS, pois de acordo com o *ranking* de classificação dos sistemas operacionais móveis mais utilizados, ela aparece como a segunda plataforma mais utilizada pelos usuários de dispositivos móveis.

Referências Bibliográficas

ALBIERO, F. W. (2017) "BlueTApp – Um Aplicativo Móvel para Registro da Frequência Acadêmica através da Tecnologia Bluetooth", http://www.brie.org/pub/index.php/wcbie/article/view/7517, Setembro 2017.

BEZERRA, Eduardo. (2007). Princípios de Análise e Projeto de Sistemas com UML. 2 ed. Rio de Janeiro: Elsevier.

Brasil. (1996) Lei nº 9.394, de 20 de dezembro de 1996, http://www.planalto.gov.br/ccivil_03/leis/L9394.htm, Agosto de 2017.

BRUSTOLIN, V. (2010). Educação apresenta novo sistema para gerenciar escolas. Acesso em 2018, disponível em https://gov-rj.jusbrasil.com.br/noticias/606265/ educacao-apresenta-novo-sistema-para-gerenciar-escolas

GARTNER. *Gartner Says Worldwide Smartphone Sales Grew 9.7 Percent in Fourth Quarter of 2016.* Disponível em: <http://www.gartner.com/newsroom/id/3215217>. Acesso em: 27 de maio de 2016.

GHODEKAR, Vaishali; KUTE, Aboli; PATIL, Swati; THAKUR, Ritesh; *Automated Attendance system with RFID through smart card. International Journal of Engineering Research e Technology* – IJERT. Vol. 2. p. 2724-2728. 2013.

GLEISER, É. (2011). "Representação Social dos Docentes do Ensino Fundamental do Destrito Federal sobre o udo do *Laptop* nas Escolas", disponível em: http://bdm.unb.br/bitstream/10483/2303/1/2011_EderGleiserdaSilvaGondim.pdf. Acesso em 13 de Junho de 2017.

HECK, F. S. (2013) "Sistema Móvel de Controle de Presença", https://www.lume.ufrgs.br/bitstream/handle/10183/100288/000931702.pdf?sequence=1. Acesso em: 07 de Setembro de 2016.

IDC. *Smartphone OS Market Share, 2017 Q2.* 2017. Disponível em: <http://www.idc.com/prodserv/smartphone-os-market-share.jsp>. Acesso em: 27 de maio de 2017.

LECHETA, Ricardo R. (2016). *Google Android:* aprenda a criar aplicações para dispositivos moveis com o Android SDK. São Paulo: Novatec Ltda.

MORENA COSTA, M. (2011) "Frequência eletrônica chega ás escolas públicas". Disponível em: http://ultimosegundo.ig.com.br/educacao/escolas-do-espirito-santo-terao-chamada-registrada-em-tablets/n1238180205572.html. Acesso em 03 de Agosto de 2016

MORETTO, Clevis Eliano. (2004) "Automação do diário de classe aplicado no ambiente de aprendizagem *LEARNLOOP"*, http://campeche.inf.furb.br/tccs/2004-II/2004-2clevisemorettovf.pdf. Acessado em : 10 de Setembro de 2016.

QUEIRÓS, Ricardo. (2017). ANDROID: *Desenvolvimento de aplicações com Android Studio*, São Paulo: FCA – Editora de Informática, Ltda, v.1, Fevereiro, 2017. Disponível em: https://issuu.com/lidel/docs/android_834d0647016686/3?e=4804698/33719448.

RABELLO, R. R. (2009). "Android: um novo paradigma de desenvolvimento móvel", Disponível em: <http://www.cesar.org.br/site/files/file/WM18_Android.pdf>. Acesso em: 10 jul. 2017.

SOMMERVILLE, I. (2007). *Engenharia de Software.* São Paulo: Pearson.

VLIET, M. V. (2013) "Controle de Frequência para Android" http://repositorio.ufla.br/bitstream/1/5465/1/MONOGRAFIA_Controle_de_frequencia_para_android.pdf. Acesso em: 02 de Setembro 2016.

WALL, Dick. (2008) "*Introduction to Android Arquitecture*", disponível em: http://www.youtube.com/watch?v=D_pGLFyqPRo. Acesso em 16 de Março de 2018.

WEISER, M. (Setembro de 1991). *The Computador for the 21st Century*. Acesso em 16 de Março de 2018, disponível em https://www.lri.fr/~mbl/Stanford/CS477/papers/Weiser-SciAm.pdf.