CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA – CEFET/RJ

Aplicação M-learning em Android

Ruan Soares Minto
Vitor Fonseca Marques dos Santos

Prof. Orientador: Fábio Paschoal Júnior

Rio de Janeiro
Agosto de 2013

CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA
CELSO SUCKOW DA FONSECA – CEFET/RJ

Aplicação M-learning em Android

Ruan Soares Minto
Vitor Fonseca Marques dos Santos

Trabalho de Conclusão de Curso apresentado em cumprimento
às normas do Departamento de Ensino Superior
do CEFET/RJ, como parte dos requisitos para obtenção
do título de Graduação Tecnológica de Sistemas em Internet.

Prof. Orientador: Fábio Paschoal Júnior

Rio de Janeiro
Agosto de 2013

DEDICATÓRIA

À Deus,
aos amigos e professores da Coordenação de Informática do CEFET-RJ,
ao nosso orientador pela sua contribuição com o trabalho,
aos nossos familiares
e à Priscila, futura esposa do Vitor, pelo seu grande apoio nas horas mais difíceis.

AGRADECIMENTOS

Agradecemos a Deus, primeiramente, por proporcionar todo o necessário até o seguinte momento. Aos familiares por se transformarem num alicerce indestrutível nos momentos de necessidade. Ainda assim, os amigos somados por todos os lugares que passamos e contribuíram de algum modo. E todas as instituições puderam auxiliar no desenvolvimento do nosso dom.

RESUMO

Este trabalho é sobre um aplicativo que foi desenvolvido para o Android, que é um sistema operacional para dispositivos móveis. Este software, que foca no ambiente de sala de aula, tem como principal objetivo realizar a comunicação do professor com seus alunos, seja através de comunicados para todos os inscritos em uma determinada disciplina ou lançando notas individuais. Esta ferramenta é muito útil para derrubar as barreiras de tempo e de localidade, pois as informações poderão ser acessadas e recebidas a qualquer momento e sem precisar de um local específico para isto ocorrer.

Palavras-chave: Dispositivos móveis, Android, M-learning.

ABSTRACT

This work is about an application that was developed for Android, which is an operational system for mobile devices. This software, which focuses on the environment of the classroom, has as main objective to make the teacher's communication with their students, either through communications to all enrollees in a particular discipline or giving individual notes. This tool is very useful to break down the barriers of time and location, because the information can be accessed and received at any time and without a specific location for this to happen.

Keywords: Mobile devices, Android, M-learning.

SUMÁRIO

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

3G	Terceira geração de padrões e tecnologias de telefonia móvel
API	Application Programming Interface
AVA	Ambientes Virtuais de Aprendizagem
AVAM	Ambientes Virtuais de Aprendizagem Móvel
EAD	Educação à distância
GPRS	General Packet Radio Service
IDE	Integrated Development Environment
MLE	Mobile Learning Engine
RN	Regras Negócio
RNF	Requisitos Não Funcionais
RS	Requisitos de Sistema
RU	Requisitos de Usuário
SMS	Short Message Service
U-LEARNING	Ubiquitous Learning, aprendizagem onipresente
WAP	Wireless Application Protocol
WI-FI	Caracteriza rede local sem fios
XML	Extensible Markup Language

Capítulo 1

1. Introdução

Há 20 anos, seria difícil a idealização da tecnologia como requisito fundamental para avançar em todos os setores, assim como na criação de objetos de utilização em massa, desde carros inteligentes a celulares inteligentes. Nos dias atuais, o uso dessas tecnologias cria uma realidade que permite um mundo cada vez mais globalizado e conectado, onde as pessoas executam suas tarefas cotidianas com uma grande facilidade.

A evolução da Internet, sem dúvida, um dos segmentos que mais cresceu neste período, seguiu cada dia mais veloz e acessível a todas as classes sociais, incluindo as classes de baixa renda através de programas sociais e do barateamento dos serviços, foram fundamentais à massificação de utilização destas tecnologias.

Essa massificação foi criada como uma consequência do crescimento econômico e da inclusão social de consumidores, onde uma quantidade expressiva de celulares é encontrada nas lojas de telefonia e varejo de todo o país. Esses celulares são adquiridos com planos de dados e Internet 3G ou 4G. Muitas vezes, a flexibilidade de planos deixa o usuário com muitas opções e poucos se recusam a ingressar nesse “novo mundo”, mesmo sem conhecê-lo.

Por outro lado, a educação, um dos setores mais criticados no Brasil, também evolui de modo discreto e sofre influências deste avanço. Uma nova metodologia foi surgindo em diversas instituições de ensino e no mundo acadêmico: a educação à distância (EAD). Por meio desta, cursos e disciplinas são aplicados de maneira não presencial. Com isso, surgem ferramentas para auxiliar este tipo de ensino, onde o Moodle - portal onde professores interagem com os alunos através de *chats* (em português significa conversação ou bate-papo) e *fóruns*, disponibilizam materiais, informam datas de provas e trabalhos, esclarecem dúvidas, dentre outras funções - aparece como o mais utilizado. Ele foi criado para auxiliar na aprendizagem, somando um canal de comunicação que aproxima alunos e professores além da tradicional sala de aula.

Em Caldeira[1] é destacada a não necessidade de um lugar concreto para acontecer a realização do curso, mas afirma que para isso acontecer é preciso disciplina do aluno. Ainda

destaca a importância do texto na aprendizagem, com ele é possível a comunicação entre pessoas e troca de informações.

A necessidade real de pesquisa nasce da relação entre Internet, sistemas de auxílio à aprendizagem, educação à distância e desenvolvimento de softwares para dispositivos móveis. Da intersecção destes temas encontramos o m-learning (*mobile learning*) com foco principal em criar uma ferramenta em plataforma Android. Um aplicativo que realize função semelhante ao portal Moodle, porém, específica para usuários de dispositivos móveis.

Segundo Serrão et al. [2], o m-learning pode ser definido como "qualquer tipo de aprendizagem que ocorre quando o estudante não está em um local fixo, predeterminado, ou quando este tira proveito das oportunidades oferecidas por tecnologias móveis".

1.1 Justificativa

A exigência de aprendizagem e atualização nos dias atuais torna necessário o acesso ao conhecimento "Just in time", através de dados e informações, estando sempre atualizado sobre qualquer assunto a todo o momento.

Em Meirelles e Tarouco [3] é mostrada uma visão interessante sobre o m-learning. Eles dizem que no mundo tão globalizado de hoje e com a necessidade de cursos para capacitação dos profissionais, o ensino móvel é uma área que se mostra interessante pela disponibilidade de se ter o material em suas mãos a todo o momento.

Devido aos meios de comunicação da Internet, intranet e outras tecnologias como SMS, WAP, GPRS e Bluetooth torna-se importante a criação de servidores ou pontos responsáveis localmente pela centralização da informação garantindo a diversificação das metodologias que implementam uma rede EAD. Assim, podem-se obter essas informações com todos os tipos de dispositivos móveis como celulares, tablets e palmtops, dentre outros.

De acordo com Serrão et al. [2], a principal característica do m-learning é a mobilidade, as pessoas têm acesso ao conteúdo requerido em qualquer lugar e a qualquer momento.

Entre os principais sistemas de gerenciamento de dispositivos móveis do mercado encontramos o Android – sistema criado pela multinacional norte-americana Google Inc. – caracterizado por seu código aberto, teve grande aceitação dos fabricantes de dispositivos

móveis tornando-se a plataforma líder que gerencia a maior parte dos dispositivos vendidos na atualidade.

Diante da quantidade de venda gigantesca de dispositivos móveis, sejam eles celulares smartphones ou tablets, cresce expressivamente o mercado de desenvolvimento de softwares para estes fins. A cada dia, surgem milhares de novos apps – nome atribuído aos aplicativos desenvolvidos para plataforma móvel – alavancados por milhões de dispositivos vendidos ao ano e bilhões de investimento.

Todavia, a porcentagem de aplicativos voltados ao meio educacional acadêmico é pequena se comparado às demais categorias, comprometendo a expansão da EAD e consequentemente a aprendizagem. Por isto, faz se necessário a criação e desenvolvimento de apps que viabilizem a m-learning.

1.2 Objetivos

O objetivo principal deste trabalho foi produzir um software, no qual seja possível um professor criar disciplina, disponibilizar comunicados e avaliar o aluno. O aplicativo aceitará matrícula de alunos nas disciplinas e alertará enviando e-mails a cada atualização ou comunicado. Com isso, espera-se diminuir a dependência da localidade e temporariedade para aprender, ubiquidade.

Pretendeu-se oferecer por meio deste projeto, uma aplicação desenvolvida para plataforma Android que tornará a interação aluno-professor de fácil utilização e acesso. A junção entre tecnologia e educação oferece um grande avanço na utilização de dispositivos móveis, permitindo que os usuários acessem e recebam cotidianamente informações importantes quando matriculados em suas disciplinas específicas.

De acordo com Franciscato e Medina [4], os benefícios do mobile learning podem ser dados ao professor e aos alunos. Para os alunos, os benefícios serão que eles terão o material disponível a qualquer momento, podendo ser acessado por eles quando quiserem. Para o professor facilita a interação com os alunos e a maior facilidade para a publicação de materiais didáticos.

Com isso, almejou-se analisar e estudar de forma aprofundada o tema proposto junto à plataforma Android. Pretendeu-se desenvolver um software que inspire investimentos futuros

em favor do tema proposto: m-learning para dispositivos móveis.

1.2.1 Objetivos Específicos

As funções do sistema foram divididas dependendo do perfil do usuário, caso usuário seja professor são suas funções:

- Solicitar perfil de professor;
- Criação de matérias;
- Aprovação de inscrição dos alunos;
- Colocar comunicados, como data de provas e entregas dos trabalhos;
- Disponibilizar as notas dos alunos.

Caso usuário seja aluno, as funções são:

- Realizar cadastro;
- Solicitar inscrição na matéria;
- Visualizar suas respectivas notas;
- Visualizar comunicados dos professores.

1.3 Metodologia

O trabalho foi um software desenvolvido para atender as necessidades do usuário em meio acadêmico. Foram fornecidos avisos e informativos, disponibilizadas datas de provas e trabalho, mostrando informações sempre atualizadas em relação às responsabilidades acadêmicas.

Foi categorizado como um produto sem fim comercial e de baixo investimento inicial. Será disponibilizado gratuitamente a qualquer pessoa que se cadastre em uma conta (professor) e que poderá criar matérias para que os seus alunos possam se inscrever.

As etapas de desenvolvimento deste trabalho foram:

1. Definição dos requisitos;

2. Construção do Protótipo;
3. Modelagem;
4. Implementação;
5. Validação.

A arquitetura utilizada no desenvolvimento é composta por camadas (podem ser visualizadas na Figura 1) e são divididas em:

- Camada de apresentação: na qual a interação do usuário com a tela é realizada e onde os usuários fazem as requisições de dados.
- Camada lógica: é o servidor de aplicações e aqui é realizada a requisição a um banco de dados para retornar os pedidos do usuário.
- Camada de dados: na qual fica o banco de dados, onde as consultas são feitas e as informações são guardadas.

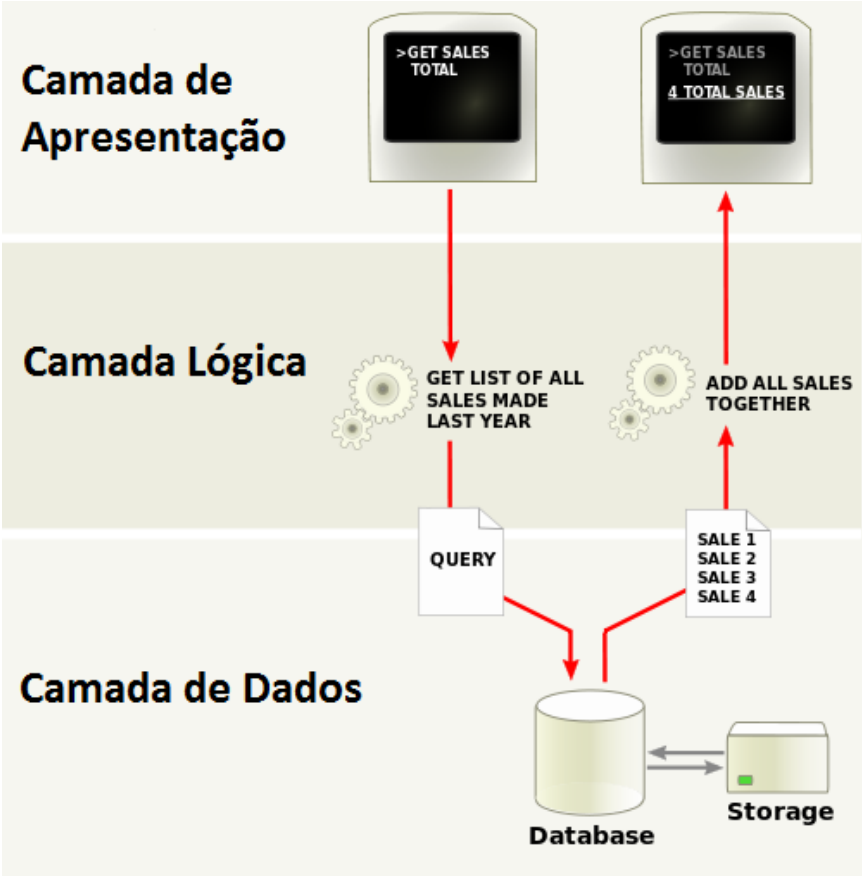

Figura 1: Camadas de arquitetura do ambiente de desenvolvimento [5].

Para desenvolver a aplicação, utilizou-se o ambiente de desenvolvimento Eclipse, uma IDE bem completa para este tipo de aplicação na linguagem Java, utilizando o plugin do Android para o Eclipse. Com relação ao banco de dados, ele foi criado através do SQLITE, que é um banco interno disponibilizado pelo Android.

1.4 Trabalhos Relacionados

Nesta seção, será feita uma análise sobre os trabalhos anteriores e eles estão colocados em ordem cronológica. Esta seção foi elaborada sempre se baseando em pesquisas realizadas na Internet ou em livros.

Em Meirelles e Tarouco [3] foi feita uma análise baseada nas tecnologias até então um pouco limitadas para a criação de um software. Nela, o autor realiza um estudo sobre um framework para aprendizagem com mobilidade. Em sua conclusão, ele diz que o m-learning tem um grande futuro e elogia o framework que fora analisado.

Em Oliveira e Medina [6] os autores realizam uma análise totalmente teórica sobre o tema proposto em nosso trabalho. Eles apresentam as limitações concretas para a criação de aplicativos de aprendizagem. Na realidade apresentada, havia o questionamento sobre as futuras condições para se produzir um software capaz, e hoje, sabemos que é bem possível.

Porém, Franciscato e Medina [4] apresentam os novos paradigmas para o m-learning, fazem uma análise da “nova” plataforma Android e da Flash Lite. Eles argumentam a falta de padronização das aplicações para ocorrer a troca de dados entre as mesmas. Os autores se mostram esperançosos com a plataforma Android no futuro.

Já em Tarouco et al. [7] foi feita uma análise da infraestrutura do ensino à distância através de e-Learning, onde foi destacada a importância da Internet como principal mecanismo para a evolução do m-learning, provendo ubiquidade no processo de aprendizagem e facilitando no desenvolvimento de estratégias para educação por meio de objetos de aprendizagem.

É destacado em Quinta e Lucena [8] o uso da tecnologia de dispositivos móveis e TV digital para suplemento das tecnologias de ensino existentes através de suporte a mídias e capacidade de processamento, padronização e construção do material instrucional. Este último é feito através de Odin, uma ferramenta de conversão automática de arquivos existentes em bases de dados em objetos de aprendizagem para diferentes dispositivos. O software oferece

acesso a arquivos personalizados de modo transparente tanto para o dispositivo de estudantes quanto para o dispositivo do docente.

No trabalho Quinta e Lucena [9] os autores enfatizam que, através da popularização de tecnologias móveis e a expansão da Internet, foi viablilizado o acesso ao conteúdo educacional de maneira ubíqua e a qualquer instante, desde que o conteúdo possa estar empregado em todos os dispositivos. Os autores também apresentam os problemas associados a este uso. O software Odin aparece como solução na adaptação automática de áudio, vídeo, imagens e texto para o conteúdo a diferentes dispositivos. O software permite adicionar extensão a servidores de aplicações para ensino á distância.

Foi destacado em Rodrigues [10] o aumento dos investimentos em tecnologia para a área da educação básica e da acessibilidade às tecnologias computacionais como dispositivos móveis, e, através do software LECA, buscou acompanhar as tendências de sistemas computacionais usados como instrumento auxiliar do professor na avaliação do processo de ensino e aprendizagem, de maneira qualitativa e quantitativa, mesmo sem uma conexão de dados com a Internet. A partir desta avaliação é possível analisar o aproveitamento da classe. O professor ainda poderá ter acesso às informações que podem auxiliar na tomada de decisões pedagógicas.

Foi realizada a análise e avaliação da aprendizagem EAD em Ribeiro et al. [11], através do AVAM MLE-Moodle utilizado no Curso de Capacitação Linguagem de Programação HTML da Universidade Federal de Santa Maria. A pesquisa mostra a satisfação dos alunos com o ambiente considerado incentivador e de grande utilidade. Nota-se maior abertura de atividades que possam ser realizadas devido a utilização dos dispositivos móveis, cumprindo assim o objetivo da usabilidade dentro do contexto do EAD. A partir disso, sente-se a necessidade de um recurso de armazenamento de vídeo que complemente atividades educacionais e auxilie professores e alunos disponibilizando o material, mantendo a característica principal da mobilidade de escolher o horário e local de estudo.

Já em Serrão et al. [2] os autores defendem a criação de um aplicativo para Android, que seja capaz de ser flexível a ponto de o usuário criar fóruns e comunidades para a troca de informações, de realizar a comunicação com Moodle e a troca de informações de alunos de diferentes universidades. A idéia proposta foi ótima e tem relação com o tema proposto para o nosso trabalho.

Em Silva et al. [12] os autores defendem a implementação de um ambiente de aprendizagem adequado às necessidades dos estudantes se preocupando sempre com a

diferença de conteúdo que é disponibilizado em salas de aula e no ambiente virtual, eles buscam soluções para resolver esse problema.

Em Caldeira [1] o autor realiza uma análise sobre os ambientes digitais de aprendizagem, fala um pouco sobre a história do ensino presencial e compara com o ensino à distância.

Foi realizada uma análise em Mühlbeier et al. [13] e observamos que os autores comentam sobre a evolução da tecnologia e, através da plataforma Android, analisam o aplicativo ToonDoo, que é utilizado para a criação de histórias em quadrinhos. Foram realizados testes com usuários e, após os testes, foram realizadas perguntas de satisfação para a melhoria do aplicativo.

Uma analogia entre EAD, Ambientes Virtuaisde Aprendizagem (AVA) e a evolução da indústria de dispositivos móveis foi feita em Fernandes [14], resultando em um m-learning que promove o ensino e a aprendizagem com mobilidade. Foi apresentada uma proposta de ampliação das estratégias de avaliação através do uso da aplicação, denominada Question Mobile, que tem o objetivo de permitir o uso de atividades avaliativas por meio de dispositivos móveis que sejam integrados aos AVA.

Em Orlandi e Isonati [15] é descrito o estudo e desenvolvimento de um sistema de autoria e distribuição de conteúdo educacional interativo para dispositivos móveis. A ferramenta permite que o aluno responda uma lista de exercícios de múltipla escolha e o sistema pode avaliar automaticamente as respostas gerando dados que podem ser usados para acompanhar o desenvolvimento dos usuários. O principal objetivo da ferramenta é permitir a verificação de dificuldades na aprendizagem de forma rápida e fácil com o auxílio do sistema computacional.

A maioria dos trabalhos descritos nessa seção são artigos que foram publicados e não possuem uma quantidade de informações muito grande. Alguns deles têm uma proposta semelhante à nossa, já outros trouxeram diversas dicas que ajudaram no desenvolvimento deste trabalho. A falta de trabalhos sobre a plataforma que escolhemos (Android) é grande devido à plataforma não ter muito tempo de vida. Porém, todos os trabalhos tiveram uma visão muito boa sobre o ensino à distância.

1.5 Organização do Trabalho

O capítulo 1 demonstrou uma breve análise introdutória ao texto que expõe os principais motivos que levaram a escolha deste tema, juntamente com as características do software desenvolvido e com os trabalhos relacionados.

No capítulo 2, são descritas as especificações do sistema, os diagramas, os modelos, a análise dos requisitos, o banco de dados e as telas do nosso aplicativo.

No capítulo 3, está a síntese das informações que levará a conclusão do texto, onde nota-se a opinião sobre a experiência e os benefícios gerados pelo software, nem sempre sendo totalmente positiva, todavia apresentando melhorias que tenham um grande potencial não somente em ambientes educacionais, mas também em ambiente empresarial e em ambiente doméstico.

Capítulo 2

2. Desenvolvimento

Nesta seção temos diversas seções e subseções, onde as seções principais são previamente apresentadas a seguir. Na fundamentação teórica, são apresentados os conceitos e teorias utilizadas no desenvolvimento do trabalho. Na especificação do sistema, os requisitos são definidos, juntamente com as regras de negócio. Após isso, são apresentados os modelos: casos de uso, classes e interações, incluindo os seus diagramas e as suas descrições. No projeto de banco de dados, são descritas as tabelas e seus respectivos atributos. Em projeto da interface gráfica é aonde é descrita a forma de interação do usuário com o sistema. E, por último, são apresentados os aspectos de implementação, descrevendo todos os passos que foram seguidos para o desenvolvimento do sistema.

2.1 Fundamentação Teórica

Dentre os conceitos de EAD e sua inter-relação com dispositivos móveis, torna estritamente necessária uma relação ao gigantesco fluxo de dispositivos como aparelhos celulares, smartphones, tablets, dentre outros, que são vendidos com o sistema Android, leva-se em consideração uma porcentagem relativamente alta de smartphones. Serão descritos a seguir alguns dos conceitos sobre EAD e de suas principais características, dos dispositivos móveis, de m-learning e da plataforma utilizada para desenvolvimento deste software.

2.1.1 Educação a Distância - EAD

A EAD, como forma de ensino que revoluciona o modo de interação entre professores e alunos, não está submetida ao conceito contemporâneo de tempo e espaço nas salas de aula. Esta surgiu da necessidade de preparo profissional e cultural de pessoas que não podem, por algum motivo, frequentar salas de aulas diariamente. Após evoluir, não somente através do surgimento de novas tecnologias, mas também o aprimoramento das tecnologias existentes tornou possível a realidade com inúmeros cursos disponibilizados através da Internet.

De acordo com Silva et al. [12], a educação à distância é "uma modalidade de ensino e aprendizagem que vem crescendo há alguns anos. De acordo com pesquisa realizada pela Associação Brasileira de Ensino à Distância (Abed) e pelo Ministério da Educação (MEC), a demanda em cursos de especialização à distância aumentou 60% de 2008 a 2010". E, pensando que em dois anos ocorreu o aumento de 60%, podemos imaginar o quanto mais cresceu de 2010 à 2013.

Ainda segundo Silva et al. [12], diz que nos dias de hoje podemos nos beneficiar com o uso do ensino à distância através dos dispositivos móveis, chamado de mobile learning ou m-learning, podendo deste modo, obter as informações necessárias a qualquer momento e em qualquer local.

Assim sendo, Rodrigues [10] diz que com o grande crescimento e disponibilidade de dispositivos móveis, aliada à utilização no ensino, originou o surgimento do m-learning, que será explicado na seção seguinte.

2.1.2 Dispositivos móveis e m-learning

Conforme dito anteriormente, devido ao gigantesco crescimento na utilização de dispositivos móveis e gadgets, juntamente com a vasta quantidade de cursos à distância que encontramos, originou-se a necessidade de obter informação em qualquer localidade ou momento, o que mais tarde fez surgir o m-learning.

Neste trabalho, foi escolhido o desenvolvimento de software para dispositivos móveis por causa da variedade de opções que temos para o mesmo, pois já dizia Fernandes et. Al [14]

que “O uso de dispositivos móveis amplia as possibilidades de ensino sem limites geográficos e temporais”.

Em Oliveira e Medina [6] é destacado o aumento significativo do uso de dispositivos móveis para várias finalidades. É destacado que os antigos dispositvos já eram fabricados com uma finalidade específica, restringindo bastante sua utilização (Ex: MP3s, primeiros celulares que só ligavam). Porém, com o grande desenvolvimento do setor computacional e com a facilidade de acesso à Internet, tornou-se possível o desenvolvimento de aplicações para dispositivos móveis.

De acordo com Mühlbeier et. Al [13], “O M-Learning é a fusão de diversas tecnologias de processamento e comunicação de dados que permite ao grupo de estudantes e aos professores uma maior interação”, e ainda definindo m-learning, Fernandes et. Al [14] diz que “O m-Learning passa a permitir que a informação e/ou o conhecimento saiam dos ambientes físicos das instituições e conquistem outros espaços, em diferentes tempos e momentos da vida (casa, trabalho, dentre outros)”.

2.1.3 Android - plataforma do passado, presente e futuro

A plataforma que mais cresce sem dúvida é a Android, que é um sistema operacional utilizado em celulares de diversas marcas nos dias de hoje, faz parte da padronização que já foi falada anteriormente, a qual facilitou o desenvolvimento de aplicações para dispositivos móveis.

Em Franciscato e Medina [4], ele analisa essa “nova” plataforma, porque no ano em que esse documento foi escrito (2008) não se tinha muitas informações sobre ela e nem muitos celulares disponíveis que rodavam Android. Ele diz que é muito promissora e revela que juntamente com o ensino à distância, pode-se obter uma mistura de sucesso, o que sabemos que ele estava certo.

Com isso, observamos que o Android ainda tem muito a evoluir, se há 5 anos atrás era promissor, podemos dizer que hoje é uma realidade, observando o fato de que a maioria dos smartphones disponíveis no mercado rodam esse sistema operacional.

2.2 Especificação do Sistema

Esta seção define toda a documentação do software produzido por este documento, a qual é dividida em subseções que estão logo a seguir.

2.2.1 Especificação dos Requisitos

Apresenta todas as solicitações e necessidades que levaram ao desenvolvimento do software. Os requisitos de usuários – solicitações feitas por usuários do sistema - e requisitos de sistemas – detalhamento das funções analisadas que serão exercidas diante das solicitações dos usuários - serão analisados como requisitos funcionais, enquanto os requisitos não funcionais – características e qualidades do software - e as regras de negócios serão apresentados a seguir.

2.2.1.1 Requisitos Funcionais

São especificações do sistema que relatam a estrutura adotada após interação com o cliente, como deverá se comportar e quais funções o software fornecerá ao usuário final. Serão especificados como requisitos de usuários e requisitos de sistema.

Os requisitos de usuário (RU) são:

RU1.	O sistema deve permitir o cadastro de alunos e professores.
RU2.	O sistema deve permitir ao professor cadastrar novas disciplinas.
RU3.	O professor deve pesquisar uma disciplina específica dentre todas as disciplinas criadas.
RU4.	O professor deve visualizar todas as disciplinas criadas.
RU5.	O professor deve editar todas as disciplinas criadas.

RU6.	O professor deve ser capaz de remover disciplinas.
RU7.	O professor poderá enviar comunicados.
RU8.	O professor poderá visualizar todos os alunos inscritos numa determinada disciplina.
RU9.	O professor poderá lançar nota para os alunos.
RU10.	O professor poderá confirmar ou cancelar a solicitação do aluno para sua disciplina.
RU11.	O professor poderá visualizar as informações de seus alunos.
RU12.	O aluno deve solicitar a inscrição na disciplina.
RU13.	O aluno deve pesquisar uma disciplina específica dentre todas as disciplinas.
RU14.	O aluno deve visualizar todas as disciplinas inscritas.
RU15.	O aluno deve ser capaz de cancelar a sua inscrição nas disciplinas.
RU16.	Os alunos poderão visualizar as informações das disciplinas.
RU17.	O sistema deve permitir que os alunos visualizem os comunicados lançados pelo professor.
RU18.	O sistema deve permitir que os alunos visualizem suas respectivas notas.
RU19.	O usuário receberá e-mails sobre atualização do aplicativo.

Tabela 1: Requisitos de Usuário.

Os requisitos de sistema (RS) são:

RS1.	O sistema deve ter dois perfis de usuário: aluno e professor.
RS2.	O sistema deve ser capaz de permitir o cadastro de alunos e professores com os seguintes atributos: nome, sobrenome, e-mail, login, senha e cpf e perfil, que pode ser ou aluno ou professor.
RS3.	O sistema deve ser capaz de permitir o professor cadastrar disciplinas com os seguintes atributos: instituicao e nome.
RS4.	O sistema deverá gerar a abreviação da disciplina, com uma combinação da instituição com o nome da disciplina.
RS5.	O sistema deverá enviar e-mails caso o aluno seja rejeitado ou aceito em alguma disciplina, mas o professor tem a opção de não enviar.
RS6.	O sistema deve permitir que os alunos realizem a solicitação de inscrição em

	disciplinas.
RS7.	Cada disciplina deve ter um único identificador.
RS8.	O cpf deverá ser o identificador do usuário.
RS9.	O sistema deve permitir a publicação de notas e comunicados pelo professor.
RS10.	O sistema deve permitir que os alunos visualizem suas notas e os comunicados feitos pelo professor.

Tabela 2: Requisitos de sistema.

2.2.1.2 Requisitos Não Funcionais

Os requisitos não funcionais (RNF) são:

RNF1.	O sistema apresentará fácil navegabilidade.
RNF2.	O sistema deverá consumir menos recurso quanto possível.
RNF3.	O tempo de resposta para consultas não deve ultrapassar 5 segundos.
RNF4.	A base de dados deve ser protegida para acesso apenas de usuários autorizados.
RNF5.	O tempo de desenvolvimento não deve ultrapassar 4 meses.
RNF6.	O usuário não necessita estar ambientado a mexer em dispositivos móveis para utilizar o software.
RNF7.	O software será desenvolvido apenas para o idioma Português-BR.
RNF8.	O acesso ao software será através de internet sem fio (Wi-fi), 3G ou 4G.

Tabela 3: Requisitos não funcionais.

2.2.1.3 Regras de Negócio

As regras de negócio (RN) são:

RN01.	Cadastro Disciplina	O campo instituição deverá ter no mínimo 4 caracteres e o campo de nome da disciplina 6 caracteres.
RN02.	Nome Disciplina	O nome da disciplina será uma combinação dos 4 primeiros caracteres da instituição com os 6 primeiros caracteres do nome da disciplina.
RN03.	Informações Disciplina	As informações mostradas da disciplina quando a mesma for selecionada serão: Quantidade de alunos, nome completo e instituição.
RN04.	Informações Aluno	As informações mostradas do aluno quando o mesmo for clicado serão: Nome, sobrenome, e-mail e login.
RN05.	Remover ou Editar Disciplina	Uma disciplina não poderá ser removida ou editada caso tenha aluno(s) inscrito(s) nela.
RN06.	Solicitação Disciplina	O aluno poderá fazer apenas uma solicitação por disciplina.
RN07.	Limite Disciplina	Cada disciplina será ministrada apenas por um professor.

Tabela 4: Regras de negócio.

2.2.2 Modelo de Casos de Uso

Nesta seção iremos representar as funcionalidades externas e como os elementos do nosso sistema vão se relacionar com essas funcionalidades. Primeiro, iremos apresentar o diagrama de casos de uso, que traz uma representação gráfica de UML (Unified Modeling Language). Em seguida teremos a descrição dos atores e dos agentes externos ao sistema e que interagem com ele, além da descrição dos casos de uso.

2.2.2.1 Diagrama de Casos de Uso

Na Figura 2, temos o diagrama de casos de uso da nossa aplicação Android:

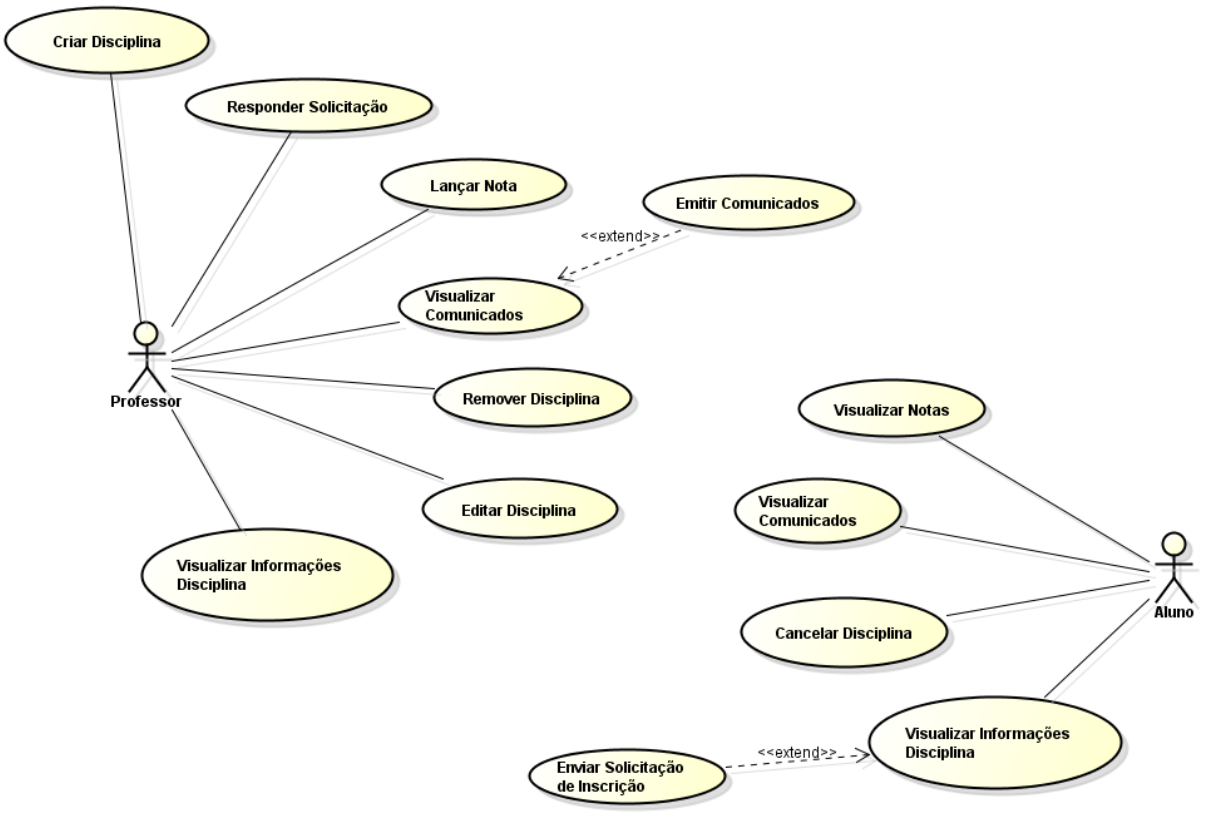

Figura 2: Diagrama de casos de uso.

2.2.2.2 Descrição dos Atores

Ator 1: Aluno

Descrição: Ator secundário no sistema mas não menos importante, sem ele o sistema não funciona perfeitamente.

Casos de uso: Visualizar notas, visualizar comunicados, cancelar disciplina, visualizar informações disciplina e enviar solicitação de inscrição.

Ator 2: Professor

Descrição: Principal ator do sistema, tem a maioria das funções e possibilidades, responsável pela comunicação com o aluno.

Casos de uso: visualizar informações disciplina, responder solicitação, lançar nota, criar disciplina, visualizar comunicados, emitir comunicados, remover disciplina, editar disciplina.

2.2.2.3 Descrição dos Casos de Uso

Nesta seção, cada caso de uso (CSU) será descrito em detalhes. Suas características são: nome do caso de uso, identificador, sumário, ator primário, ator secundário (opcional), pré-condições, fluxo principal, fluxo alternativo, fluxo de exceção, pós-condições, regras do negócio, autor e data.

Segue abaixo a descrição do caso de uso criar disciplina:

Nome do caso de uso	Criar disciplina
Identificador	CSU01
Sumário	Professor usa o sistema para criar disciplinas.
Ator primário	Professor
Pré-condições	O professor está logado no sistema.
Fluxo Principal	1. O professor, na sua tela inicial, clica em **Criar Disciplina**. 2. O sistema apresenta os campos que o professor deve preencher, os quais são o campo “instituição” e o campo “nome da disciplina”. 3. O professor preenche os campos de acordo com a RN01 e clica em “Salvar”. 4. O sistema gera o nome de abreviação da disciplina de acordo com a RN02 e apresenta uma mensagem dizendo “A disciplina foi criada com sucesso!”. 5. O professor clica em “OK” e o caso de uso se encerra.

Fluxo Alternativo	(2) Cancelamento de ação a. O professor clica no botão “Cancelar” e o caso de uso se encerra.
Fluxo de Exceção	(3) Não digitação dos campos a. O professor não preenche os campos e clica em “Salvar”. b. O sistema apresenta a mensagem “Todos os campos são obrigatórios!”. c. O professor clica em “OK” e volta para o passo 2. (3) Violação da RN01 a. O sistema apresenta a mensagem “O campo Instituição tem que ter 4 dígitos no mínimo e o nome da disciplina 6!”. c. O professor clica em “OK” e volta para o passo 2.
Pós-condições	O professor criou uma disciplina.
Regras do negócio	RN01 e RN02.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 5: CU01 – Criar disciplina.

Segue abaixo a descrição do caso de uso visualizar informações disciplina:

Nome do caso de uso	Visualizar informações disciplina
Identificador	CSU02
Sumário	O usuário utiliza o sistema para visualizar informações sobre uma disciplina desejada.
Ator primário	Professor e aluno.
Pré-condições	O usuário está logado no sistema.
Fluxo Principal	1. O usuário, na sua tela inicial clica em **Pesquisar**. 2. O sistema apresenta a tela de busca, listando todas as disciplinas já criadas. 3. O usuário digita o trecho de texto que pretende buscar e clica no botão “Pesquisar”. 4. O sistema apresenta o resultado da busca do usuário.

	5. O usuário seleciona a disciplina desejada clicando no nome dela. 6. O sistema apresenta as informações da disciplina de acordo com a RN03 e o caso de uso se encerra.
Pós-condições	O usuário visualizou as informações de uma disciplina desejada.
Regras de negócio	RN03.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 6: CU02 - Visualizar informações disciplina.

Segue abaixo a descrição do caso de uso responder solicitação:

Nome do caso de uso	Responder solicitação
Identificador	CSU03
Sumário	O professor utiliza o sistema para responder uma solicitação de inscrição de um aluno em sua disciplina.
Ator primário	Professor.
Pré-condições	O professor está logado no sistema.
Fluxo Principal	1. O professor, em sua tela inicial, clica em **Minhas Disciplinas.** 2. O sistema apresenta a lista de disciplinas que são lecionadas pelo professor. 3. O professor seleciona uma. 4. O sistema apresenta a tela de disciplina selecionada. 5. O professor clica em **Alunos.** 6. O sistema apresenta uma lista com os alunos da disciplina. 7. O professor clica em “Ver Solicitações”. 8. O sistema apresenta uma lista com as solicitações. 9. O professor clica em uma solicitação. 10. O sistema exibe a mensagem “Aluno deseja entrar em Disciplina. Entre com a justificativa abaixo:”, um campo não brigatório de justificativa e dois botões: Aceitar e

	recusar. 11. O professor clica em “Aceitar”. 12. O sistema apresenta uma janela para envio de e-mail ao aluno. 13. O professor clica em “Enviar” e o e-mail é enviado. 14. O sistema apresenta uma mensagem “A solicitação foi aceita com sucesso!”. 15. O professor clica em “Ok” e o caso de uso se encerra.
Fluxo Alternativo	(9) Clicando em “Aceitar Todas” a. O sistema apresenta uma janela para envio de e-mail aos aluno. b. O professor clica em “Enviar” e o e-mail é enviado. c. O sistema apresenta a mensagem “Todas as solicitações foram aceitas com sucesso!”. d. O professor clica em “Ok” e o caso de uso se encerra. (9) Clicando em “Recusar Todas” a. O sistema apresenta uma janela para envio de e-mail aos aluno. b. O professor clica em “Enviar” e o e-mail é enviado. c. O sistema apresenta a mensagem “Todas as solicitações foram recusadas com sucesso!”. d. O professor clica em “Ok” e o caso de uso se encerra. (11) Recusando uma a. O professor clica em “Recusar”. b. Continua a partir do 12. (13) Não enviando e-mail a. O professor clica em “Descartar” ou “Salvar como rascunho” e o e-mail não é enviado. b. Continua a partir do 14.
Fluxo de Exceção	(8) Nenhuma solicitação a. O sistema exibe a mensagem “Não existem solicitações para essa disciplina!”.

	b. O professor clica em “Ok” e o caso de uso se encerra.
Pós-condições	O professor respondeu as solicitações dos alunos.
Regras de negócio	Nenhuma.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 7: CU03 - Responder solicitação.

Segue abaixo a descrição do caso de uso lançar nota:

Nome do caso de uso	Lançar nota
Identificador	CSU04
Sumário	O professor utiliza o sistema para lançar a nota de um determinado aluno.
Ator primário	Professor.
Pré-condições	O professor está logado no sistema.
Fluxo Principal	1. O professor, em sua tela inicial, clica em **Minhas Disciplinas.** 2. O sistema apresenta a lista de disciplinas que são lecionadas pelo professor. 3. O professor seleciona uma. 4. O sistema apresenta a tela de disciplina selecionada. 5. O professor clica em **Alunos.** 6. O sistema apresenta uma lista com os alunos da disciplina. 7. O professor seleciona um aluno. 8. O sistema apresenta as informações do aluno selecionado de acordo com a RN04. 9. O professor clica em “Lançar Nota”. 10. O sistema apresenta os campos que deverão ser preenchidos pelo professor. 11. O professor preenche os campos e clica em “Salvar”. 12. O sistema apresenta uma janela para envio de e-mail ao aluno. 13. O professor clica em “Enviar” e o e-mail é enviado.

	14. O sistema apresenta a mensagem “A nota foi lançada com sucesso!”. 15. O professor clica em “Ok” e o caso de uso se encerra.
Fluxo Alternativo	(11) Campos não preenchidos a. O sistema apresenta a mensagem “Todos os campos são obrigatórios!”. b. O professor clica em “Ok” e o caso de uso se encerra. (13) E-mail não enviado a. O professor clica em “Descartar” ou “Salvar como rascunho”. b. Continua no 14. (11) Cancelar a. O professor clica em “Cancelar” e o caso de uso se encerra.
Fluxo de Exceção	(6) Nenhum aluno a. O sistema exibe a mensagem “Essa disciplina não tem alunos”. b. O professor clica em “Ok” e o caso de uso se encerra.
Pós-condições	O professor lançou uma nota apara um determinado aluno.
Regras de negócio	RN04.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 8: CU04 - Lançar nota.

Segue abaixo a descrição do caso de uso visualizar comunicados:

Nome do caso de uso	Visualizar comunicados
Identificador	CSU05
Sumário	O usuário deseja visualizar os comunicados de uma disciplina que leciona (professor) ou que está inscrito (aluno).
Ator primário	Professor e aluno.
Pré-condições	O usuário está logado no sistema.
Fluxo Principal	1. O usuário, em sua tela inicial, clica em **Minhas Disciplinas.**

	2. O sistema apresenta a lista de disciplinas que são lecionadas por ele (professor) ou que ele está inscrito (aluno). 3. O usuário seleciona uma disciplina. 4. O sistema apresenta a tela de disciplina selecionada. 5. O usuário clica em **Comunicados**. 6. O sistema apresenta uma lista de comunicados da disciplina e o caso de uso se encerra.
Fluxo de Exceção	(2) Nenhuma disciplina a. O sistema exibe a mensagem "Você não tem disciplinas!". b. O usuário clica em "Ok" e o caso de uso se encerra. (6) Nenhum comunicado (professor) a. O sistema exibe um item da lista dizendo "Nenhum comunicado!". (6) Nenhum comunicado (aluno) a. O sistema apresenta a mensagem "Não existem comunicados para esta disciplina!". b. O aluno clica em "Ok" e o caso de uso se encerra.
Pós-condições	O usuário visualizou os comunicados da disciplina.
Regras de negócio	Nenhuma.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 9: CU05 - Visualizar comunicados.

Segue abaixo a descrição do caso de uso emitir comunicados:

Nome do caso de uso	Emitir comunicados
Identificador	CSU06
Sumário	O professor deseja publicar um comunicado em uma disciplina.
Ator primário	Professor.
Pré-condições	O professor está logado no sistema.

Fluxo Principal	1. O professor, em sua tela inicial, clica em **Minhas Disciplinas.** 2. O sistema apresenta a lista de disciplinas que são lecionadas pelo professor. 3. O professor seleciona uma. 4. O sistema apresenta a tela de disciplina selecionada. 5. O professor clica em **Comunicados**. 6. O sistema apresenta uma lista de comunicados da disciplina. 7. O professor clica em "Fazer Comunicado". 8. O sistema apresenta os campos que deverão ser preenchidos. 9. O professor preenche devidamente os campos e clica em "Salvar". 10. O sistema apresenta uma janela para envio de e-mail aos alunos. 11. O professor clica em "Enviar" e o e-mail é enviado. 12. O sistema apresenta a mensagem "O comunicado foi enviado com sucesso!". 13. O professor clica em "Ok" e o caso de uso se encerra.
Fluxo Alternativo	(9) Cancelar a. O professor clica em "Cancelar" e o caso de uso se encerra.
Fluxo de Exceção	(2) Nenhuma disciplina a. O sistema exibe a mensagem "Você não tem disciplinas!". b. O usuário clica em "Ok" e o caso de uso se encerra. (6) Nenhum comunicado (professor) a. O sistema exibe um item da lista dizendo "Nenhum comunicado!". (9) Campos não preenchidos a. O professor esquece de preencher um campo e clica em "Salvar".

	b. O sistema apresenta a mensagem “Todos os campos são obrigatórios!”. c. O professor clica em “Ok” e volta para o passo 8.
Pós-condições	O professor emitiu um comunicado para todos os alunos da disciplina.
Regras de negócio	Nenhuma.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 10: CU06 - Emitir comunicados.

Segue abaixo a descrição do caso de uso remover disciplina:

Nome do caso de uso	Remover disciplina
Identificador	CSU7
Sumário	O professor deseja remover uma disciplina.
Ator primário	Professor.
Pré-condições	O professor está logado no sistema.
Fluxo Principal	1. O usuário, em sua tela inicial, clica em **Minhas Disciplinas.** 2. O sistema apresenta a lista de disciplinas que são lecionadas por ele (professor) ou que ele está inscrito (aluno). 3. O usuário seleciona uma. 4. O sistema apresenta a tela de disciplina selecionada. 5. O professor clica em **Remover Disciplina.** 6. Se estiver de acordo com a RN05, o sistema apresenta a mensagem de confirmação “Tem certeza que deseja cancelar essa disciplina?” 7. O professor clica em “Sim”. 8. O sistema apresenta a mensagem “Disciplina apagada com sucesso!”. 9. O professor clica em “Ok” e o caso de uso se encerra.
Fluxo Alternativo	(7) Não remove

	a. O professor clica em “Não” e o caso de uso se encerra.
Fluxo de Exceção	(6) Aluno na disciplina a. O sistema apresenta a mensagem “Essa disciplina não pode ser removida pois tem alunos inscritos nela!”. b. O professor clica em “Ok” e o caso de uso se encerra.
Pós-condições	O professor remove a disciplina desejada.
Regras de negócio	RN05.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 11: CU07 - Remover disciplina.

Segue abaixo a descrição do caso de uso editar disciplina:

Nome do caso de uso	Editar disciplina
Identificador	CSU12
Sumário	O professor deseja editar uma disciplina.
Ator primário	Professor.
Pré-condições	O professor está logado no sistema.
Fluxo Principal	1. O professor, em sua tela inicial, clica em **Minhas Disciplinas.** 2. O sistema apresenta a lista de disciplinas que são lecionadas por ele. 3. O professor seleciona uma. 4. O sistema apresenta a tela de disciplina selecionada. 5. O professor clica em **Editar Disciplina.** 6. Se estiver de acordo com a RN05, o sistema apresenta os campos já preenchidos com os dados da disciplina, instituição e nome da disciplina. 7. O professor altera aonde for preciso e clica em “Atualizar”. 8. O sistema exibe a mensagem “A alteração foi realizada com sucesso!”. 9. O professor clica em “Ok” e o caso de uso se encerra.
Fluxo Alternativo	(6) Cancelar

	a. O professor clica em “Cancelar” e o caso de uso se encerra. (7) Não muda os campos a. O professor clica em “Atualizar”. b. O sistema apresenta a mensagem “Já existe disciplina com esse nome criada. Tente colocar letras ou números no início para diferenciar.”. c. O professor clica em “Ok” e volta para o passo 6.
Fluxo de Exceção	(2) Nenhuma disciplina a. O sistema exibe a mensagem “Você não tem disciplinas!”. b. O usuário clica em “Ok” e o caso de uso se encerra. (6) Aluno na disciplina a. O sistema apresenta a mensagem “A alteração não pode ser realizada pois a disciplina já tem alunos inscritos!”. b. O professor clica em “Ok” e o caso de uso se encerra. (6) Campo não preenchido a. O professor apaga um dos campos e clica em “Atualizar”. b. O sistema apresenta a mensagem “Todos os campos são obrigatórios!”. c. O professor clica em “Ok” e volta para o passo 6.
Pós-condições	O professor editou as informações da disciplina desejada.
Regras de negócio	RN05.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 12: CU08 - Editar disciplina.

Segue abaixo a descrição do caso de uso visualizar notas:

Nome do caso de uso	Visualizar notas
Identificador	CSU09
Sumário	O aluno deseja visualizar suas notas de uma determinada

	disciplina.
Ator primário	Aluno.
Pré-condições	O aluno está logado no sistema.
Fluxo Principal	1. O aluno, em sua tela inicial, clica em **Minhas Disciplinas.** 2. O sistema apresenta a lista de disciplinas que ele está inscrito. 3. O aluno seleciona uma. 4. O sistema apresenta a tela de disciplina selecionada. 5. O aluno clica em **Notas**. 6. O sistema apresenta a lista de notas do aluno e o caso de uso se encerra.
Fluxo de Exceção	(2) Nenhuma disciplina a. O sistema exibe a mensagem "Você não tem disciplinas!". b. O usuário clica em "Ok" e o caso de uso se encerra. (6) Nenhuma nota a. O sistema exibe a mensagem "Nenhuma nota foi lançada!". b. O aluno clica em "Ok" e o caso de uso se encerra.
Pós-condições	O aluno visualizou suas notas de uma determinada disciplina.
Regras de negócio	Nenhuma.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 13: CU09 - Visualizar notas.

Segue abaixo a descrição do caso de uso cancelar disciplina:

Nome do caso de uso	Cancelar disciplina
Identificador	CSU10
Sumário	O aluno deseja cancelar a inscrição em uma disciplina.
Ator primário	Aluno.
Pré-condições	O aluno está logado no sistema.
Fluxo Principal	1. O aluno, em sua tela inicial, clica em **Minhas Disciplinas.** 2. O sistema apresenta a lista de disciplinas em que ele está

	inscrito. 3. O aluno seleciona uma. 4. O sistema apresenta a tela de disciplina selecionada. 5. O aluno clica em **Cancelar Disciplina.** 6. O sistema apresenta a mensagem de confirmação "Tem certeza que deseja cancelar essa disciplina?". 7. O aluno clica em "Sim". 8. O sistema apresenta a mensagem "Disciplina cancelada com sucesso!". 9. O aluno clica em "Ok" e o caso de uso se encerra.
Fluxo Alternativo	(7) Não remove a. O aluno clica em "Não" e o caso de uso se encerra.
Pós-condições	O aluno cancelou a disciplina desejada.
Regras de negócio	Nenhuma.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 14: CU10 - Cancelar disciplina.

Segue abaixo a descrição do caso de uso enviar solicitação de inscrição:

Nome do caso de uso	Enviar solicitação de inscrição
Identificador	CSU11
Sumário	O aluno utiliza o sistema para solicitar a inscrição em uma determinada disciplina.
Ator primário	Aluno.
Pré-condições	O aluno está logado no sistema.
Fluxo Principal	1. O aluno, na sua tela inicial clica em **Pesquisar**. 2. O sistema apresenta a tela de busca, listando todas as disciplinas já criadas. 3. O aluno digita o trecho de texto que pretende buscar e clica no botão "Pesquisar". 4. O sistema apresenta o resultado da busca do usuário. 5. O aluno seleciona a disciplina desejada clicando no nome

	dela. 6. O sistema apresenta as informações da disciplina de acordo com a RN03. 7. O aluno clica em “Solicitar Inscrição”. 8. O sistema apresenta a mensagem “Deseja solicitar inscrição em disciplina?”. 9. O aluno clica em “Sim” e o caso de uso se encerra.
Fluxo Alternativo	(9) Não solicita a. O aluno clica em “Não” e a solicitação não é feita. (7) Apenas uma solicitação por disciplina a. Não está de acordo com a RN06 e aí não poderá solicitar a inscrição. O caso de uso se encerra.
Pós-condições	O aluno solicitou a inscrição em uma determinada disciplina.
Regras de negócio	RN06.
Autor	Vitor Fonseca
Data	13/06/2013

Tabela 15: CU11 - Enviar solicitação de inscrição.

2.2.3 Modelo de Classes

O modelo de classes tem como sua principal característica representar as partes estruturais do sistema, sendo que cada classe representa uma entidade, que por sua vez, possuem seus respectivos atributos.

Logo a seguir, é apresentado graficamente o diagrama de classes. Depois, são apresentadas mais duas seções: o dicionário de classes e o VCP (Visões das classes participantes).

2.2.3.1 Diagrama de Classes

Na Figura 3, temos o diagrama de classes do nosso software:

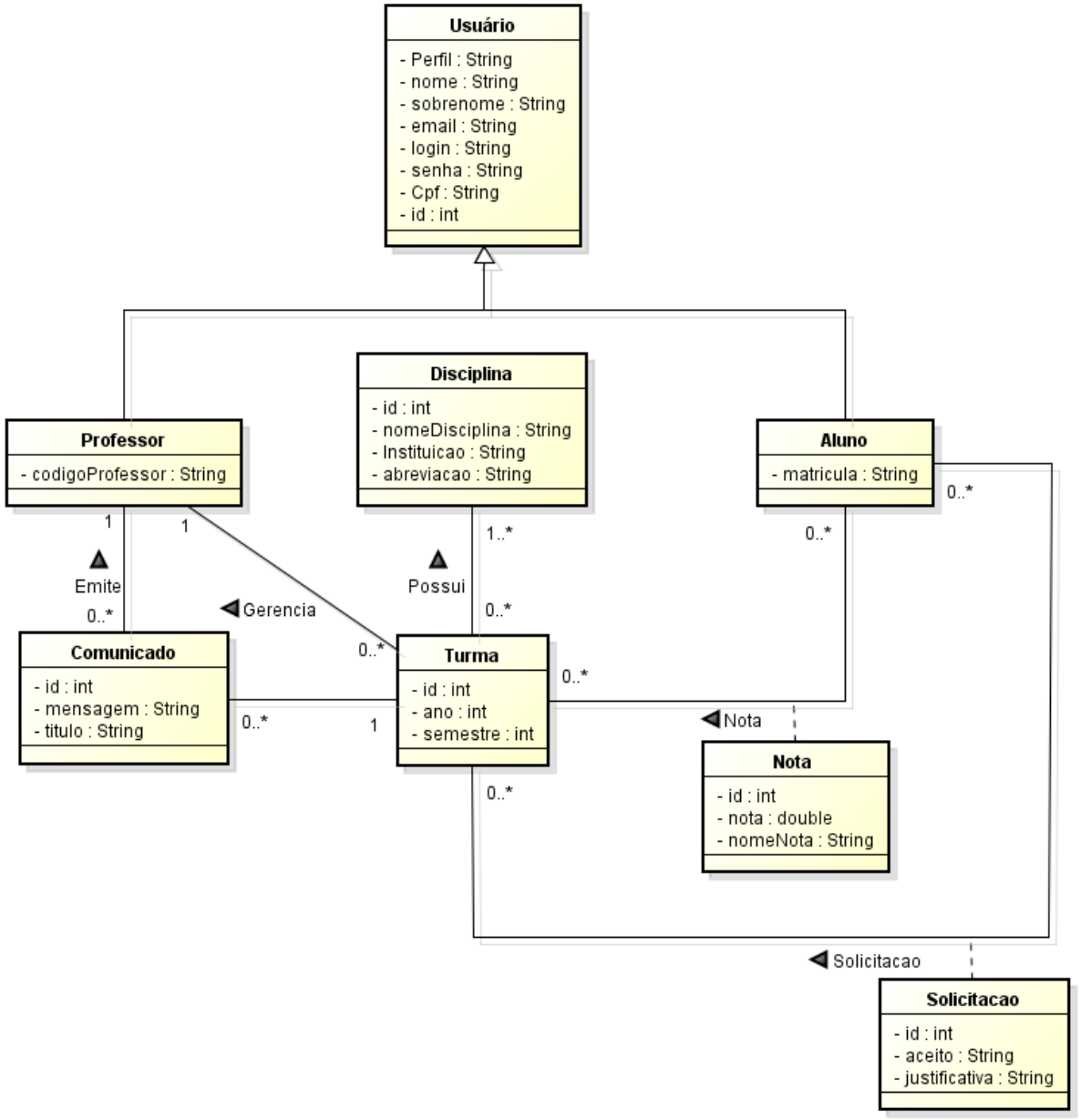

Figura 3: Diagrama de classes.

2.2.3.2 Dicionário de Classes

No dicionário de classes, são mostradas as informações importantes sobre cada classe do sistema. Cada entidade será dividida em cinco colunas: atributo, classe, domínio, tamanho e descrição. No atributo definimos o nome do campo. Na classe definimos se é determinante, caso de se ter um conteúdo único, ou simples. No domínio definimos o tipo do atributo, numérico, texto, data, etc. Os outros dois campos são de fácil entendimento.

Segue abaixo o dicionário da classe usuário:

Entidade: Usuário				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
_id	Determinante	Numérico		Código do usuário.
perfil	Simples	Texto		“Aluno” ou “Professor”.
nome	Simples	Texto	15	Nome do usuário.
sobrenome	Simples	Texto	15	Sobrenome do usuário.
email	Simples	Texto	40	E-mail do usuário.
login	Simples	Texto	15	Login do usuário.
senha	Simples	Texto	15	Senha do usuário.
cpf	Determinante	Texto	11	Cpf do usuário.

Tabela 16: Dicionário de classes da classe usuário.

Segue abaixo o dicionário da classe professor:

Entidade: Professor				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
codigoProfessor	Determinante	Texto	6	Código único do professor.

Tabela 17: Dicionário de classes da classe professor.

Segue abaixo o dicionário da classe aluno:

Entidade: Aluno

Atributo	Classe	Domínio	Tamanho	Descrição
matricula	Determinante	Texto	8	Matrícula do aluno.

Tabela 18: Dicionário de classes da classe aluno.

Segue abaixo o dicionário da classe disciplina:

Entidade: Disciplina				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
_id	Determinante	Numérico		Código da disciplina.
nomeDisciplina	Simples	Texto	30	Nome da disciplina.
instituicao	Simples	Texto	30	Instituição da disciplina.
abreviacao	Determinante	Texto	11	Abreviação da disciplina.

Tabela 19: Dicionário de classes da classe disciplina.

Segue abaixo o dicionário da classe comunicado:

Entidade: Comunicado				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
_id	Determinante	Numérico		Código do comunicado.
titulo	Simples	Texto	30	Título do comunicado.
comunicado	Simples	Texto	100	Mensagem do comunicado.

Tabela 20: Dicionário de classes da classe comunicado.

Segue abaixo o dicionário da classe turma:

Entidade: Turma				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
_id	Determinante	Numérico		Código da turma.
ano	Simples	Numérico	4	Ano que a turma foi criada.
semestre	Simples	Numérico	1	Semestre que a turma foi criada.

Tabela 21: Dicionário de classes da classe turma.

Segue abaixo o dicionário da classe nota:

Entidade: Nota				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
_id	Determinante	Numérico		Código da nota.
nota	Simples	Numérico (double)	5	Nota do aluno.
nomeNota	Simples	Texto	15	Nome da nota.

Tabela 22: Dicionário de classes da classe nota.

Segue abaixo o dicionário da classe solicitações:

Entidade: Solicitação				
Atributo	**Classe**	**Domínio**	**Tamanho**	**Descrição**
_id	Determinante	Numérico		Código da solicitação.
aceito	Simples	Texto	1	“S” = Sim. “N” = não. “J” = justificativa entregue.
justificativa	Simples	Texto		Justificativa de resposta do professor.

Tabela 23: Dicionário de classes da classe solicitações.

2.2.3.3 Visões das Classes Participantes

Uma visão das classes participantes (VCP) é feita por cada caso de uso, retratando graficamente todos os objetos que participam dele.

A Figura 4 representa a VCP do caso de uso criar disciplina:

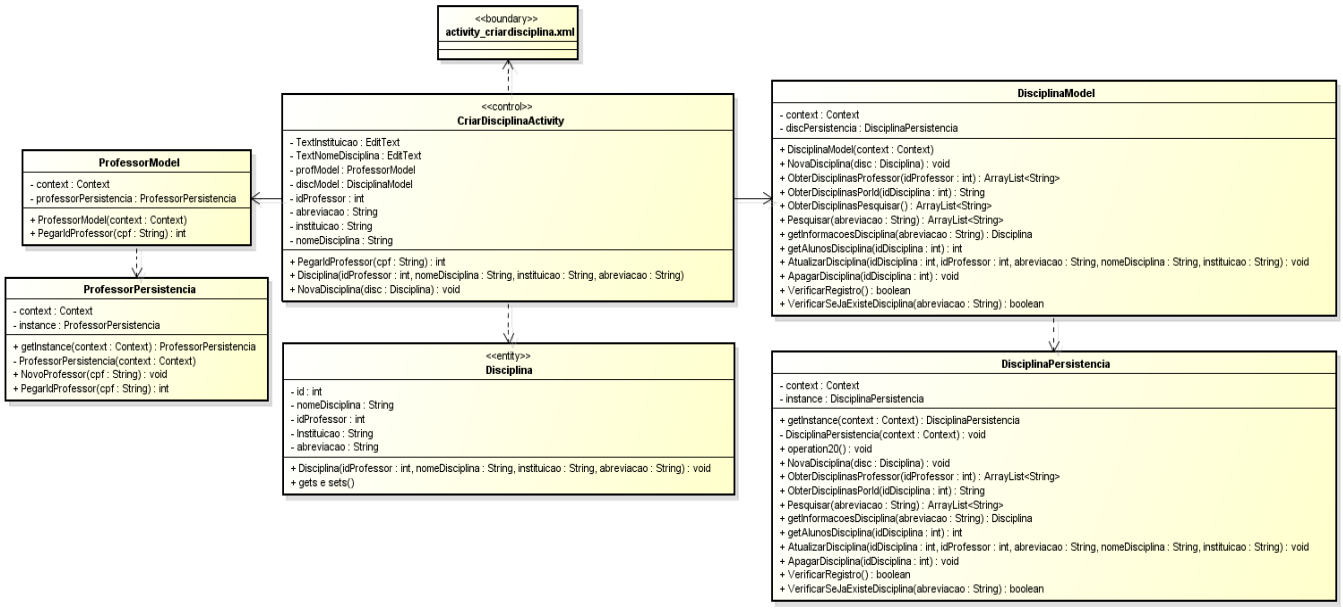

Figura 4: VCP – Criar disciplina.

A Figura 5 representa a VCP do caso de uso emitir comunicados:

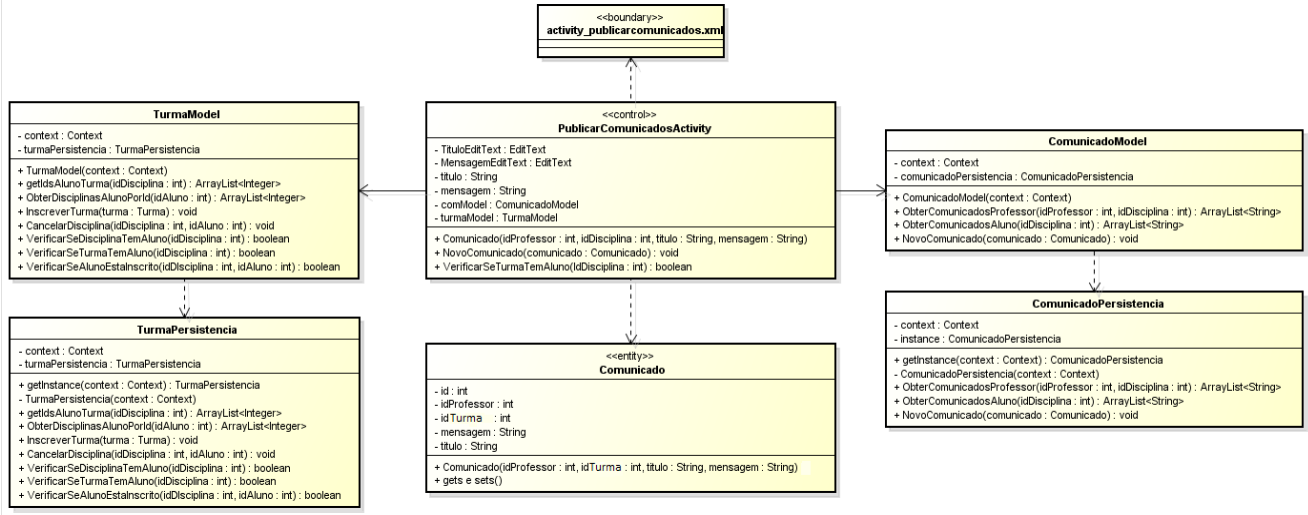

Figura 5: VCP – Emitir comunicados.

A Figura 6 representa a VCP do caso de uso lançar nota:

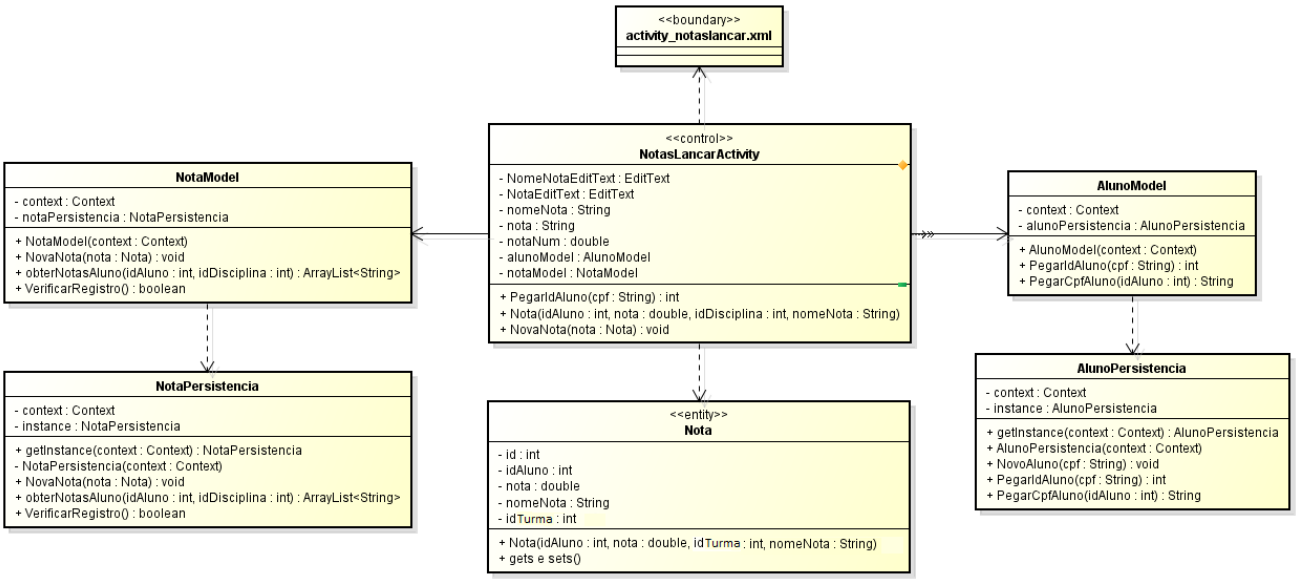

Figura 6: VCP – Lançar nota.

A Figura 7 representa a VCP do caso de uso remover disciplina:

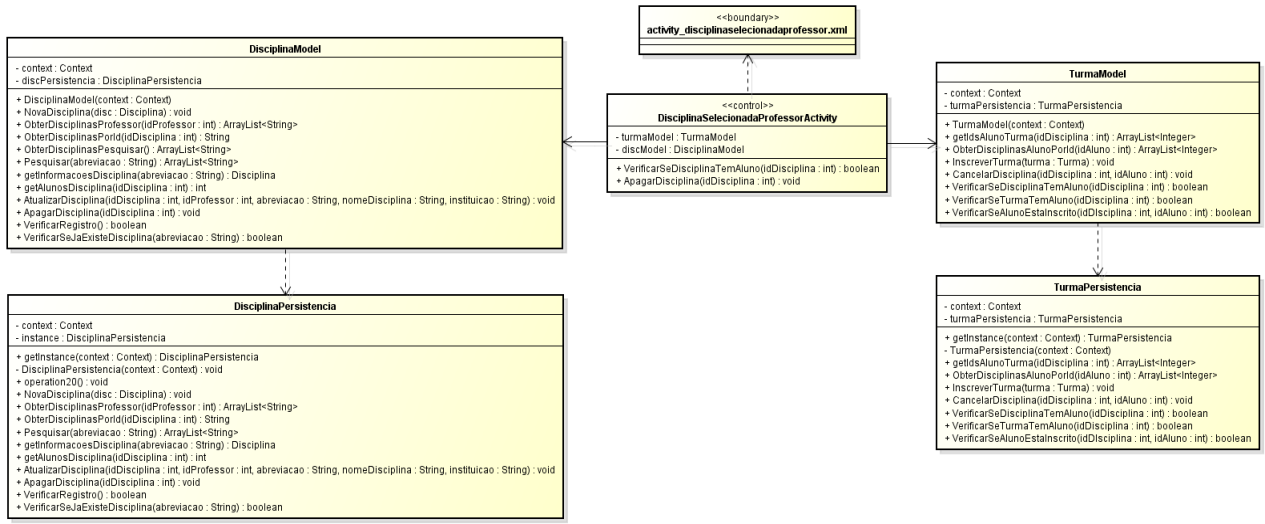

Figura 7: VCP – Remover disciplina.

A Figura 8 representa a VCP do caso de uso visualizar notas:

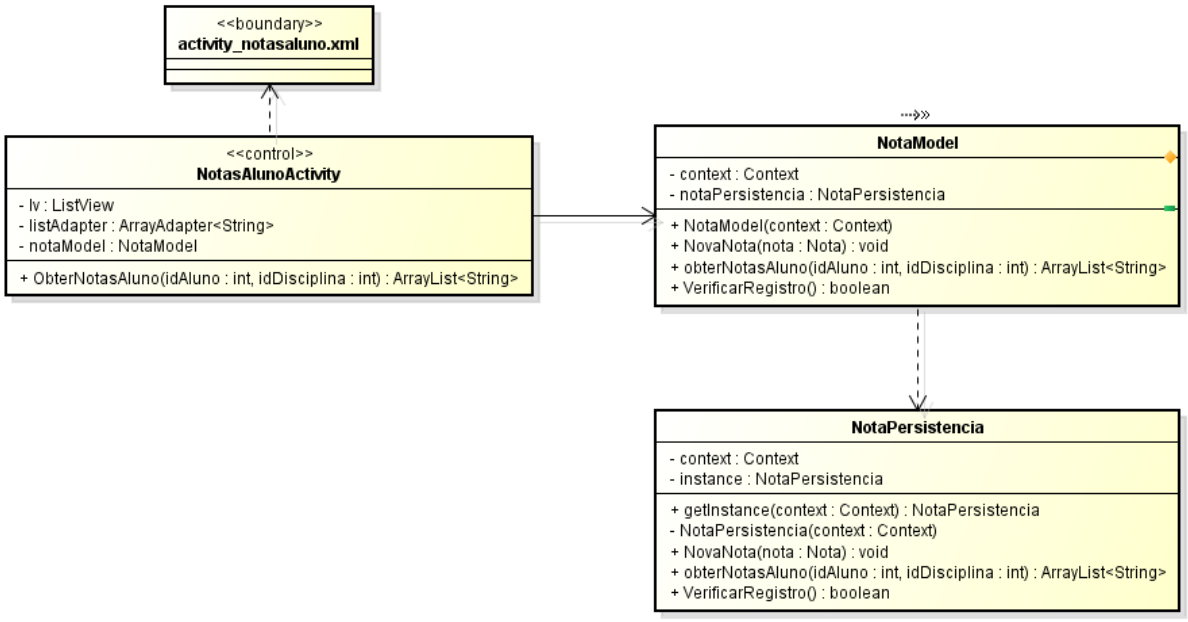

Figura 8: VCP – Visualizar notas.

2.2.4 Modelo de Interações

Para completar a apresentação de modelos do sistema, apresentamos o modelo de interações, o qual representa as diversas mensagens que são trocadas entre os vários objetos para a execução de métodos de uma respectiva tela do sistema. Devido ao grande número de métodos que o nosso software tem e para não ficar muito longa esta parte, apresentaremos o diagrama de sequência dos métodos mais importantes apenas. Inicialmente, iremos descrever as operações do sistema (métodos) textualmente, e, logo em seguida, apresentar os diagramas de sequência.

2.2.4.1 Descrição Textual das Operações do Sistema

Apresentamos nesta seção, os métodos das classes de persistência do projeto. Esses métodos são chamados por sua classe correspondente de modelo (se for AlunoPersistencia, seu modelo é a classe AlunoModel), que por sua vez são chamados pela sua Activity

correspondente. Nas classes de persistência são realizadas as operações com o banco de dados (select, update, create, insert, delete) e as activities são as telas do sistema.

AlunoPersistencia:

Operação:	+ NovoAluno (in matricula: String)
Responsabilidade:	Salvar um novo aluno, passando como parâmetro a matrícula do aluno.

Operação:	+ PegarIdAluno (in cpf: String): int
Responsabilidade:	Retornar o id do aluno, passando como parâmetro o cpf do mesmo.

Operação:	+ PegarCpfAluno (in idAluno: int): String
Responsabilidade:	Retornar o cpf do aluno, passando como parâmetro o id do mesmo.

ComunicadoPersistencia:

Operação:	+ ObterComunicadosProfessor (in idProfessor: int, in idTurma int): ArrayList<String>
Responsabilidade:	Retornar uma lista de comunicados do professor, passando como parâmetro o identificador do professor e o identificador da turma.

Operação:	+ ObterComunicadosAluno (in idTurma: int): ArrayList<String>
Responsabilidade:	Retornar uma lista de comunicados do aluno, passando como parâmetro o identificador da turma.

Operação:	+ NovoComunicado (in com: Comunicado)
Responsabilidade:	Salvar um novo comunicado no banco de dados.

DisciplinaPersistencia:

Operação:	+ NovaDisciplina (in disc: Disciplina)
Responsabilidade:	Salvar uma nova disciplina, passando como parâmetro o objeto disciplina.

Operação:	+ ObterDisciplinasProfessor (in idProfessor: int): ArrayList<String>
Responsabilidade:	Retornar uma lista de nomes de disciplinas de um professor, passando como parâmetro o identificador do professor.

Operação:	+ ObterDisciplinasPorId (in idDisciplina: int): String
Responsabilidade:	Retornar um nome de disciplina, passando como parâmetro o identificador da disciplina.

Operação:	+ ObterDisciplinasPesquisar () : ArrayList<String>
Responsabilidade:	Retornar a lista de disciplinas que já foram criadas no banco de dados.

Operação:	+ Pesquisar (in abreviacao: String): ArrayList<String>
Responsabilidade:	Retornar uma lista de disciplinas, passando como parâmetro uma string.

Operação:	+ getInformacoesDisciplina (in abreviacao: String): Disciplina
Responsabilidade:	Retornar um objeto disciplina, passando como parâmetro uma string.

Operação:	+ getAlunosDisciplina (in idDisciplina: int): int
Responsabilidade:	Retornar a quantidade de alunos de uma disciplina, passando como parâmetro o identificador da disciplina.

Operação:	+ AtualizarDisciplina (in idDisciplina: int, in idProfessor: int, in abreviacao: String, in nomeDisciplina: String, in instituicao: String)
Responsabilidade:	Atualizar as informações de uma disciplina, passando como parâmetro todos os atributos da disciplina.

Operação:	+ ApagarDisciplina (in idDisciplina: int)
Responsabilidade:	Apagar uma disciplina, passando como parâmetro o identificador da disciplina.

Operação:	+ VerificarRegistro(): boolean
Responsabilidade:	Verificar se tem alguma disciplina criada no banco de dados.

Operação:	+ VerificarSeJaExisteDisciplina (in abreviacao: String): boolean
Responsabilidade:	Verificar se tem uma disciplina criada no banco de dados, passando como parâmetro uma string.

NotaPersistencia:

Operação:	+ NovaNota(in nota: Nota)
Responsabilidade:	Salvar uma nota no banco de dados, passando o objeto nota como parâmetro.

Operação:	+ ObterNotasAluno (in idAluno: int, in idTurma: int): ArrayList<String>
Responsabilidade:	Retornar uma lista com notas, passando como parâmetro o identificador do aluno e o identificador da turma.

Operação:	+ VerificarRegistro (): Boolean
Responsabilidade:	Verificar se tem alguma nota lançada no banco de dados.

ProfessorPersistencia:

Operação:	+ NovoProfessor (in codigoProfessor: String)
Responsabilidade:	Salvar um novo professor, passando como parâmetro o código do professor.

Operação:	+ PegarIdProfessor (in cpf: String): int
Responsabilidade:	Retornar o identificador do professor, passando como parâmetro o cpf do mesmo.

SolicitacaoPersistencia:

Operação:	+ CriarSolicitacao (in solic: Solicitacoes)
Responsabilidade:	Salvar uma nova solicitação no banco de dados, passando o objeto solicitações como parâmetro.

Operação:	+ getSolicitacoes (in idTurma: int): ArrayList<Integer>
Responsabilidade:	Retornar uma lista de identificadores das solicitações de uma disciplina, passando como parâmetro o identificador da turma.

Operação:	+ RetornarJustificativas (in idAluno: int): ArrayList<Solicitacoes>
Responsabilidade:	Retornar uma lista de objetos de solicitações, passando como parâmetro o identificador do aluno.

Operação:	+ AtualizarSolic (in solic: Solicitacoes)
Responsabilidade:	Atualizar uma solicitação, passando como parâmetro o objeto solicitações.

Operação:	+ AtualizarSolicitacaoJ (in solic: Solicitacoes, in aceito: String)
Responsabilidade:	Atualizar uma solicitação, passando como parâmetro o objeto solicitações e uma string.

Operação:	+ VerificarRegistroSolic (in idAluno: int, in idTurma: int): Boolean
Responsabilidade:	Verificar se tem um registro no banco de dados, passando como parâmetro o identificador do aluno e o identificador da turma.

Operação:	+ getQtdSolicitacoes (in idTurma: int): int
Responsabilidade:	Retornar a quantidade de solicitações que uma disciplina tem, passando como parâmetro o identificador da turma.

TurmaPersistencia:

Operação:	+ InscreverTurma (in turma: Turma)
Responsabilidade:	Salvar um novo registro na tabela turma, passando como parâmetro o objeto turma.

Operação:	+ CancelarDisciplinaAluno (in idDisciplina: int, in idAluno: int)
Responsabilidade:	Apagar um registro na tabela turma, passando como parâmetro o identificador da disciplina e o identificador do aluno.

Operação:	+ VerificarSeDisciplinaTemAluno (in idDisciplina: int): boolean
Responsabilidade:	Verificar se uma disciplina tem alunos inscritos, passando como parâmetro o identificador da disciplina.

Operação:	+ getIdsAlunosTurma (in idDisciplina: int): ArrayList<Integer>
Responsabilidade:	Retornar uma lista de identificadores dos alunos de uma turma, passando como parâmetro o identificador da disciplina.

Operação:	+ VerificarSeTurmaTemAluno (in idDisciplina: int): Boolean
Responsabilidade:	Verificar se uma turma tem alunos nela, passando como parâmentro o identificador da disciplina.

Operação:	+ VerificarSeAlunoEstaInscrito (in idDisciplina: int, in idAluno: int): boolean
Responsabilidade:	Verificar se um aluno está inscrito em uma disciplina, passando como parâmetro o identificador da disciplina e o identificador do aluno.

UsuarioPersistencia:

Operação:	+ NovoUsuario (in nome: String, in sobrenome: String, in email: String, in login: String, in senha: String, in cpf: String, in perfil: String)
Responsabilidade:	Salvar um novo usuário, passando como parâmetro todos os atributos da tabela usuário.

Operação:	+ ObterEmailUsuario (in cpf: String): String
Responsabilidade:	Retornar o e-mail do usuário, passando como parâmetro o cpf do mesmo.

Operação:	+ PegarInformacoesAluno (in cpf: String): Usuario
Responsabilidade:	Retorna um objeto de usuário, passando como parâmentro o cpf do mesmo.

Operação:	+ VerificarSeLoginExiste (in login: String): boolean
Responsabilidade:	Verificar se o login passado como parâmetro existe no banco de dados.

Operação:	+ VerificarSeCpfExiste (in cpf: String): boolean
Responsabilidade:	Verifica se o cpf passado como parâmetro existe no banco de dados.

Operação:	+ ObterNomeUsuario (in cpf: String): String
Responsabilidade:	Retornar o nome do usuário, passando como parâmetro o cpf do mesmo.

Operação:	+ VerificarRegistro(): boolean
Responsabilidade:	Verificar se tem usuário criado no sistema.

CriarTabelasPersistencia:

Operação:	+ criarUsuario()
Responsabilidade:	Criar se não existir a tabela usuário no banco de dados.

Operação:	+ criarProfessor()
Responsabilidade:	Criar se não existir a tabela professor no banco de dados.

Operação:	+ criarAluno()
Responsabilidade:	Criar se não existir a tabela aluno no banco de dados.

Operação:	+ criarDisciplina()
Responsabilidade:	Criar se não existir a tabela disciplina no banco de dados.

Operação:	+ criarNota()
Responsabilidade:	Criar se não existir a tabela nota no banco de dados.

Operação:	+ criarComunicado()
Responsabilidade:	Criar se não existir a tabela comunicado no banco de dados.

Operação:	+ criarTurma()
Responsabilidade:	Criar se não existir a tabela turma no banco de dados.

Operação:	+ DeletarTudoTurma()
Responsabilidade:	Apagar todo o conteúdo da tabela turma.

Operação:	+ criarSolicitacao()
Responsabilidade:	Criar se não existir a tabela solicitacao no banco de dados.

Operação:	+ VerificarLogin (in login: String, in senha: String): boolean
Responsabilidade:	Verificar se o login e senha inseridos pelo usuário, estão corretos.

2.2.4.2 Diagrama de Sequência

O diagrama de sequência (DS) representa as ações que o sistema deve realizar para um determinado caso de uso. Na parte superior estão as classes, telas e atores. Na parte inferior é retratado o relacionamento entre esses objetos. Apresentamos abaixo cinco diagramas de sequência do nosso software, que são eles: criar disciplina, visualizar notas, lançar nota, emitir comunicados e remover disciplina.

A Figura 9 representa o diagrama de sequência do caso de uso criar disciplina:

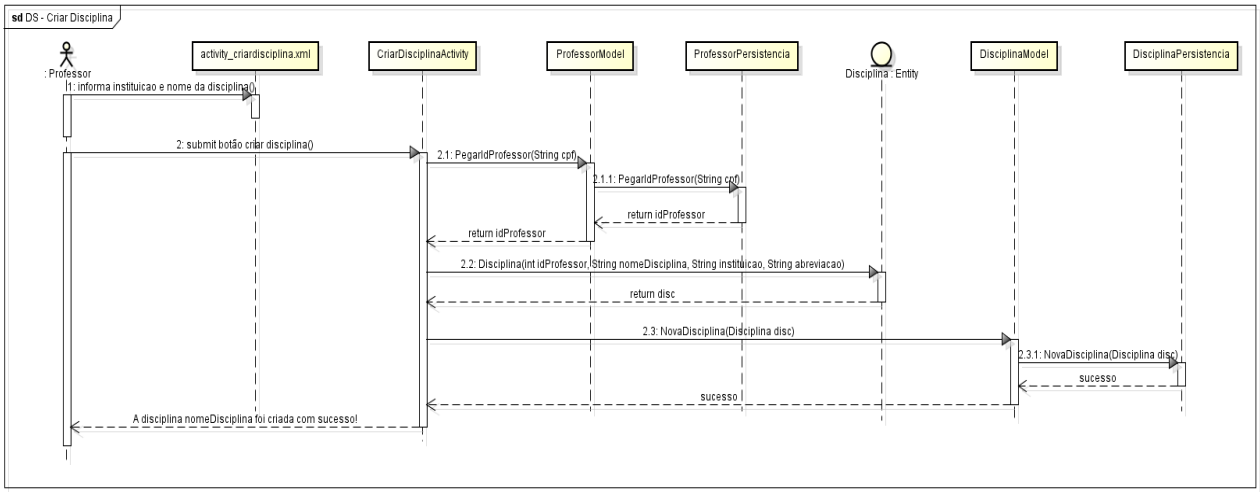

Figura 9: DS – Criar disciplina.

A Figura 10 representa o diagrama de sequência do caso de uso visualizar notas:

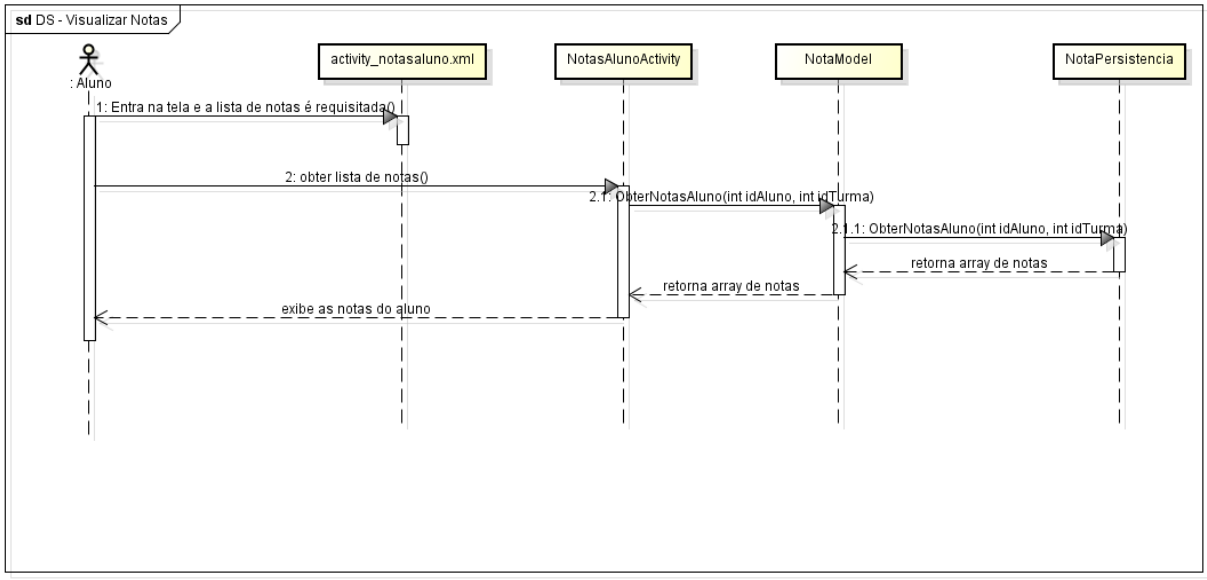

Figura 10: DS – Visualizar notas.

A Figura 11 representa o diagrama de sequência do caso de uso lançar nota:

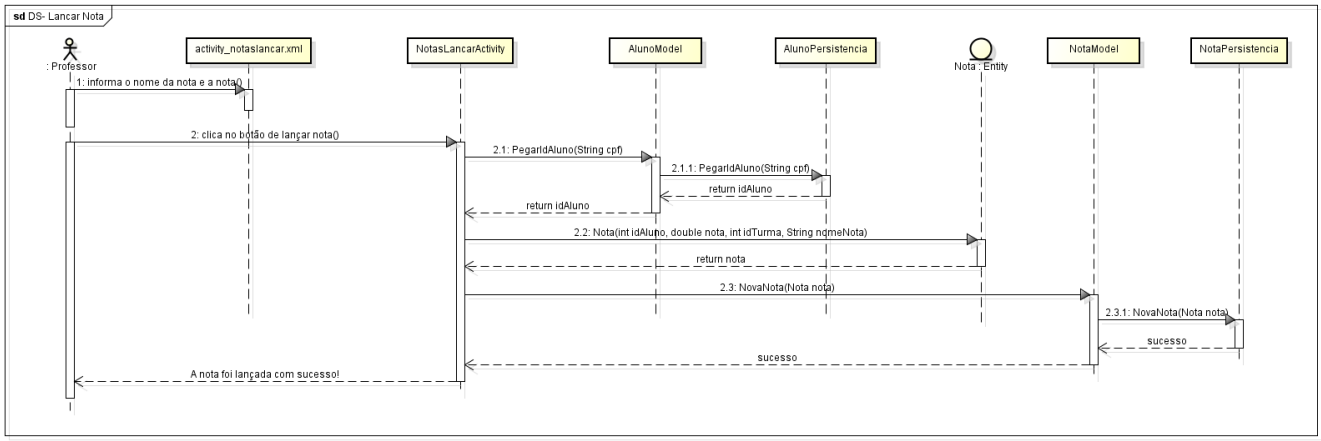

Figura 11: DS – Lançar nota.

A Figura 12 representa o diagrama de sequência do caso de uso emitir comunicados:

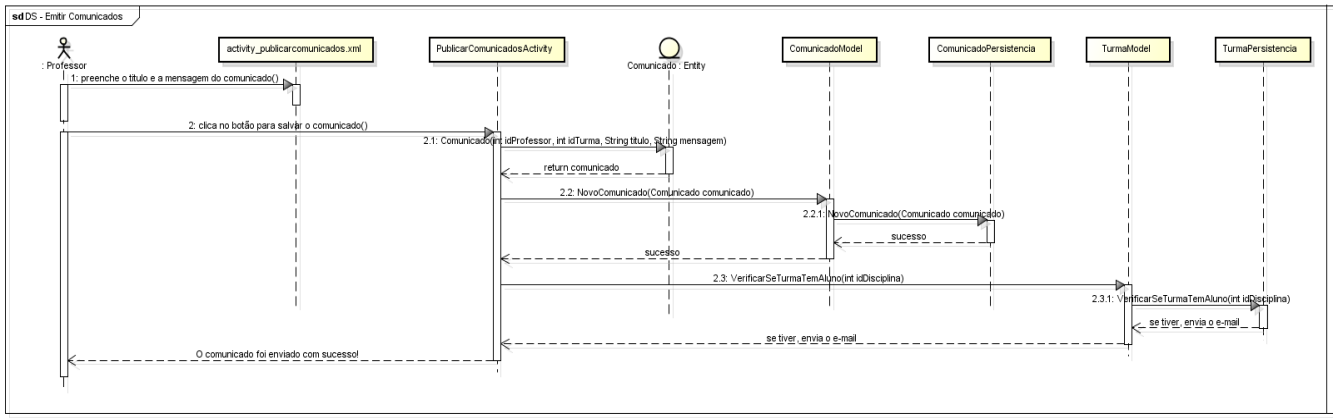

Figura 12: DS – Emitir comunicados.

A Figura 13 representa o diagrama de sequência do caso de uso remover disciplina:

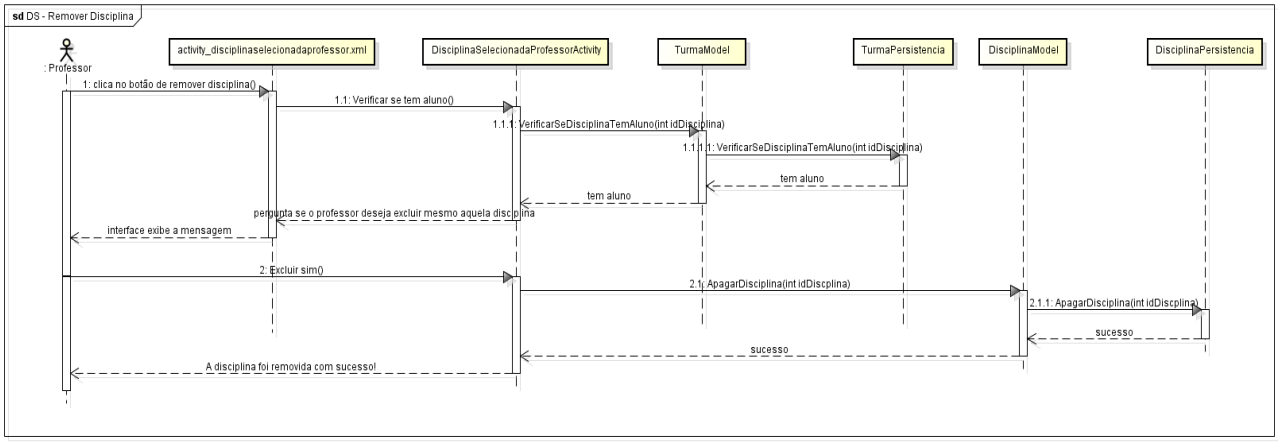

Figura 13: DS – Remover disciplina.

2.2.5 Projeto de Banco de Dados

Nesta seção do trabaho sobre o projeto de banco de dados é onde todas as tabelas do sistema são descritas. Foram criadas subseções, cada uma representando seus níveis de abstrações: projeto lógico de banco de dados e projeto físico de banco de dados.

2.2.5.1 Projeto Lógico de Banco de Dados

O projeto lógico de banco de dados tem como objetivo avaliar o esquema conceitual do uso de banco de dados, visando o desenvolvimento de seu esquema lógico. O modelo utilizado foi o relacional, que retrata da melhor maneira possível o que o usuário precisa.

Colocaremos o nome de cada tabela, com os respectivos atributos dentro dos parênteses. A chave primária está em negrito, as chaves estrangeiras estão em itálico e as restrições de integridade (RI) estão abaixo das respectivas tabelas.

O projeto lógico é o seguinte:

usuario(**id**, perfil, nome, sobrenome, email, login, senha, cpf)

RI01: perfil deve ser diferente de nulo.

RI02: nome deve ser diferente de nulo.

RI03: sobrenome deve ser diferente de nulo.

RI04: email deve ser diferente de nulo.

RI05: email deve ser único.

RI06: login deve ser diferente de nulo.

RI07: login deve ser único.

RI08: senha deve ser diferente de nulo.

RI09: cpf deve ser diferente de nulo.

RI10: cpf deve ser único.

Professor(codigoProfessor)

RI01: codigoProfessor deve ser diferente de nulo.

RI02: codigoProfessor deve ser único.

Aluno(matricula)

RI01: matricula deve ser diferente de nulo.

RI02: matricula deve ser único.

Disciplina(**id**, nomeDisciplina, idProfessor, instituicao, abreviacao)

RI01: nomeDisciplina deve ser diferente de nulo.

RI02: idProfessor deve ser diferente de nulo.

RI03: instituicao deve ser diferente de nulo.

RI04: abreviacao deve ser diferente de nulo.

RI05: abreviacao deve ser único.

Nota(**id**, *idAluno*, *idTurma*, nota, nomeNota)

RI01: idAluno deve ser diferente de nulo.

RI02: idTurma deve ser diferente de nulo.

RI03: nota deve ser diferente de nulo.

RI04: nomeNota deve ser diferente de nulo.

Turma(**id**, *idAluno*, *idDisciplina*)

RI01: idAluno deve ser diferente de nulo.

RI02: idDisciplina deve ser diferente de nulo.

Solicitacao(**id**, *idAluno*, *idTurma*, aceito, justificativa)

RI01: idAluno deve ser diferente de nulo.

RI02: idTurma deve ser diferente de nulo.

Comunicado(**id**, *idProfessor*, *idTurma*, mensagem, titulo)

RI01: idProfessor deve ser diferente de nulo.

RI02: idTurma deve ser diferente de nulo.

RI03: mensagem deve ser diferente de nulo.

RI04: titulo deve ser diferente de nulo.

2.2.5 Projeto Físico de Banco de Dados

O projeto físico de banco de dados se baseia no esquema lógico para desenvolver o esquema físico. Tem como objetivo otimizar o desempenho das operações de consulta do sistema, manipulando os dados da melhor maneira possível. Após completar esta etapa, o banco de dados já pode ser criado e populado. A Figura 14 representa o projeto físico de banco de dados do nosso sistema.

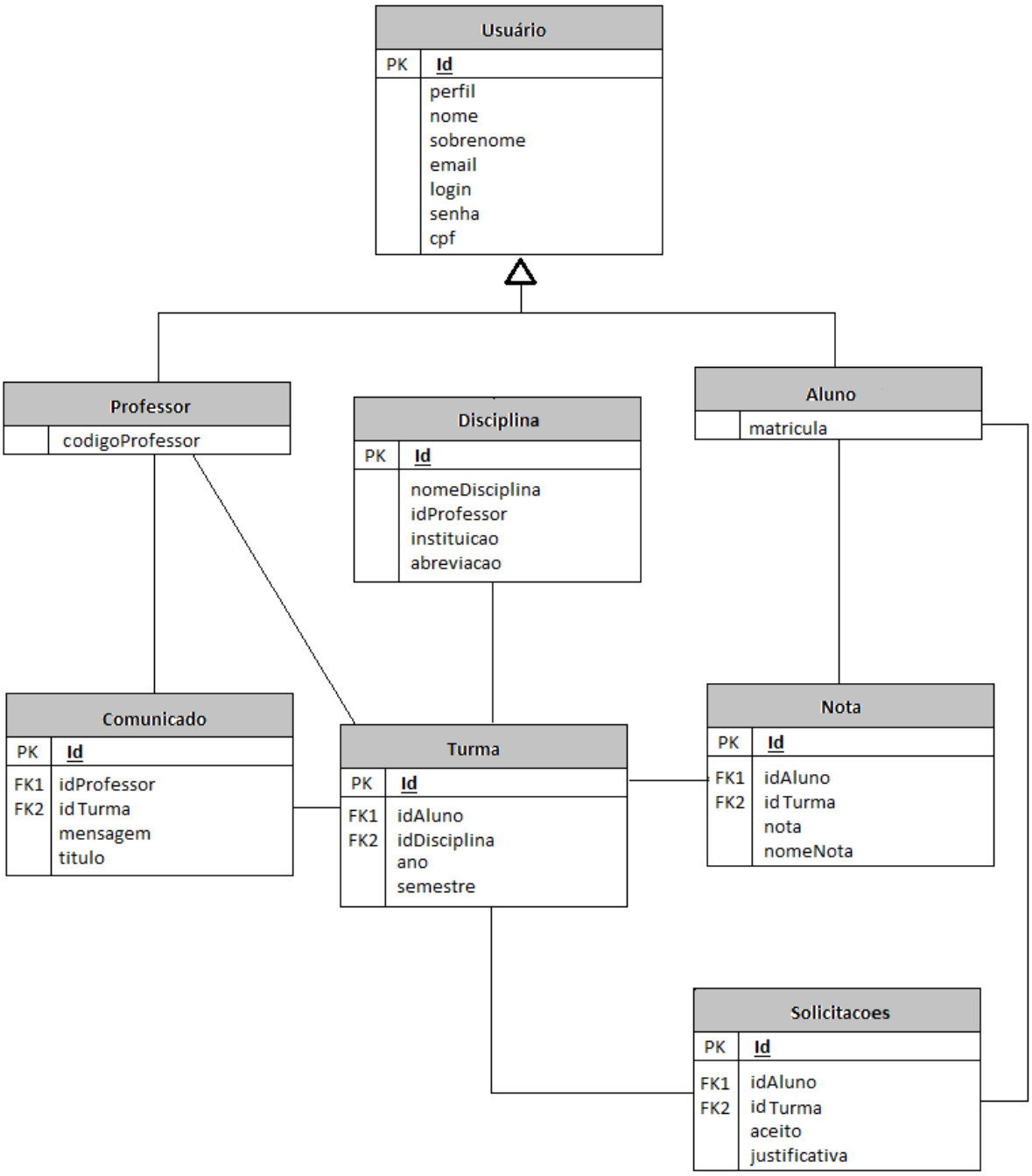

Figura 14: Projeto físico do banco de dados.

2.2.6 Projeto de Interface Gráfica

O projeto de interface gráfica descreve toda a parte prática do desenvolvimento do sofware proposto, para escrever esta seção, todas as etapas de implementação do software têm

que estar completas. As telas do sistema estão na seção de apêndices e, na sequência, será mostrada a hierarquia das telas do sistema.

2.2.6.1 Hierarquia das telas

No Android, cada Activity representa uma tela do sistema. Abaixo, mostraremos a hierarquia de telas separadas por perfil (professor e aluno) utilizando o nome de cada Activity.

A Figura 15 representa o caminho entre as telas disponíveis do sistema para quem tiver o perfil de professor. Após ser feito o login, ele tem para navegação no software o seguinte caminho:

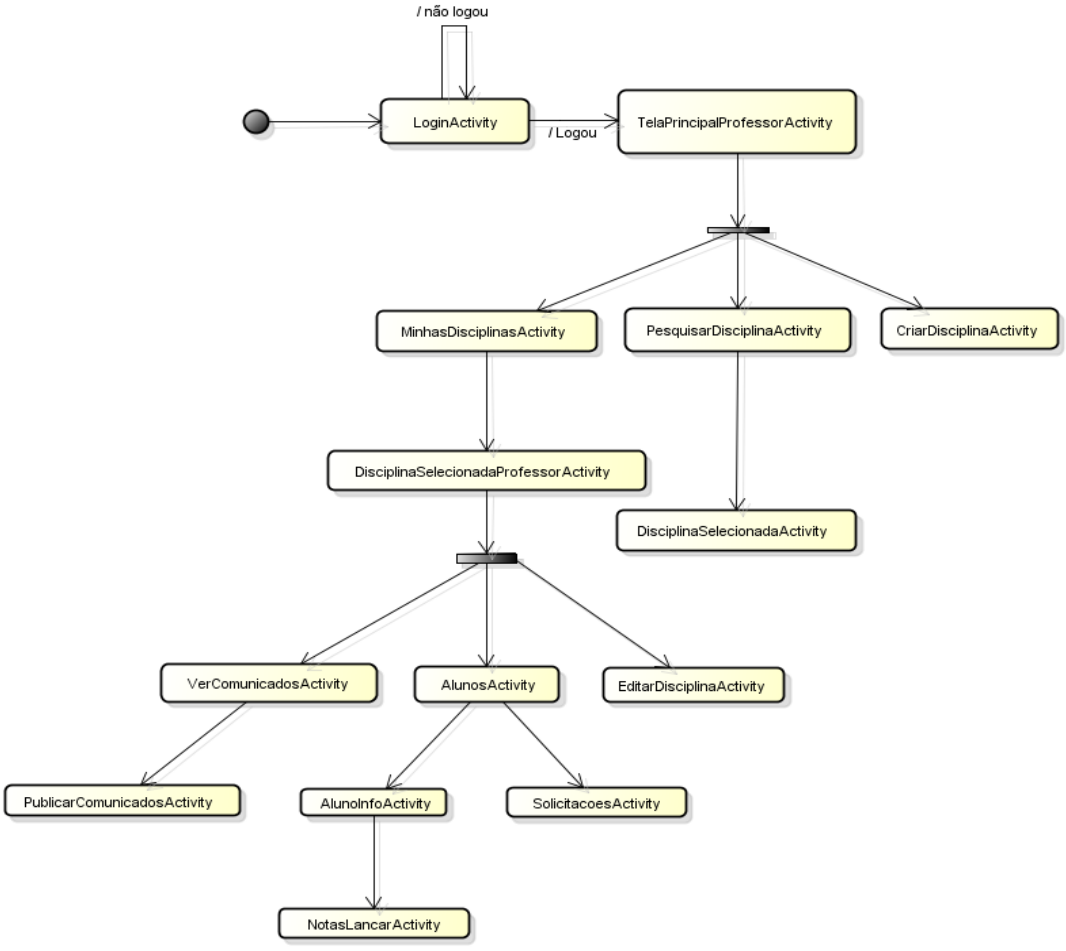

Figura 15: Hierarquia das telas do perfil de professor.

A Figura 16 representa o caminho entre as telas disponíveis do sistema para quem tiver o perfil de aluno. Podemos notar que o número de telas disponibilizadas para o aluno é bem menor do que para o professor, pois o mesmo tem mais tarefas para realizar no sistema. Após ser feito o login, ele tem para navegação no software o seguinte caminho:

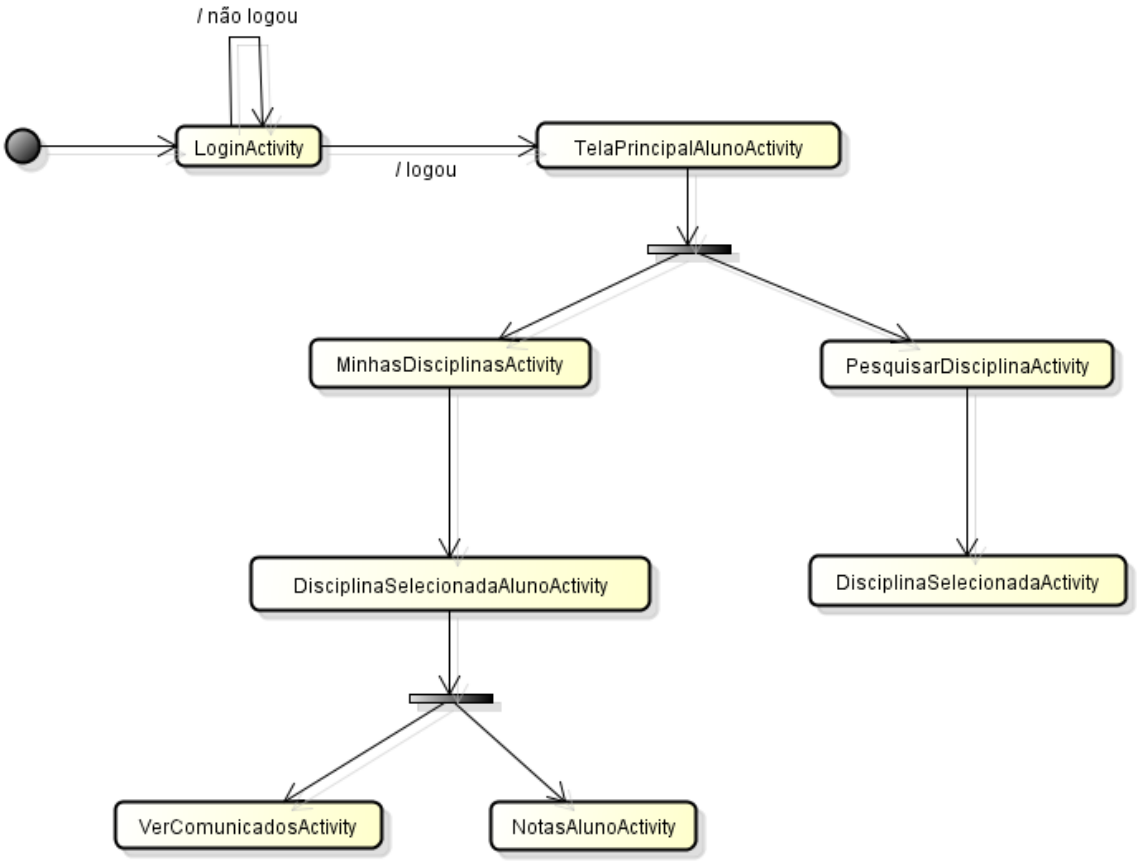

Figura 16: Hierarquia das telas do perfil de aluno.

2.3 Aspectos de Implementação

Todo software possui etapas de desenvolvimento e para o melhor entendimento da elaboração deste aplicativo, vamos mostrar primeiro a arquitetura de uma aplicação Android, descrevendo todas as suas camadas logo em seguida. Após isso, será realizada uma análise geral, com foco na implementação do software, aonde as etapas que foram utilizadas para desenvolver este aplicativo serão mostradas e descritas.

2.3.1 Arquitetura da Aplicação

Devido a grande evolução do mercado de desenvolvimento de aplicações para dispositivos móveis e tablets optamos por aprender algo novo e que seria útil para melhorar ainda mais o currículo. A plataforma Android foi a escolhida porque abrange uma maior variedade de dispositivos que rodam este sistema operacional. A sua escolha também foi relacionada com a linguagem que a mesma suporta, a linguagem Java, que já é de amplo domínio, pois foi a linguagem mais utlizada durante todo o curso.

Para iniciarmos o desenvolvimento foi instalado o JDK (Java Development Kit), que é um kit de desenvolvimento Java, que possui as ferramentas necessárias para desenvolver nesta linguagem. Após isso, foi instalado o Android SDK (Software Development Kit) , que é um kit de desenvolvimento para aplicações que irão rodar na plataforma Android. O Eclipse, que é uma IDE (Integrated Development Environment) desenvolvida em Java, é o próximo a ser instalado, sendo escolhido pela sua interface simples e funcional, uma outra opção era o NetBeans. Por último, deverá ser instalado no Eclipse, um plugin para o Android, chamado de ADT (Android Development Tools).

O banco de dados que foi escolhido foi o SQLite, que é um banco nativo da plataforma, ele é uma pequena biblioteca desenvolvida na linguagem C e que tem acesso a operações SQL. Foi uma opção gratuita e altamente funcional, novamente foi priorizada a experiência com um novo banco para a escolha.

Para o software ser testado, o Eclipse tem a opção de criar máquinas virtuais para realizar esse teste. Nossa aplicação foi testada nas máquinas virtuais contendo as versões 2.2 e 4.2 do Android.

2.3.2 Camadas de Aplicação

A arquitetura de aplicações Android, que é dividida em camadas, é mostrada na Figura 17:

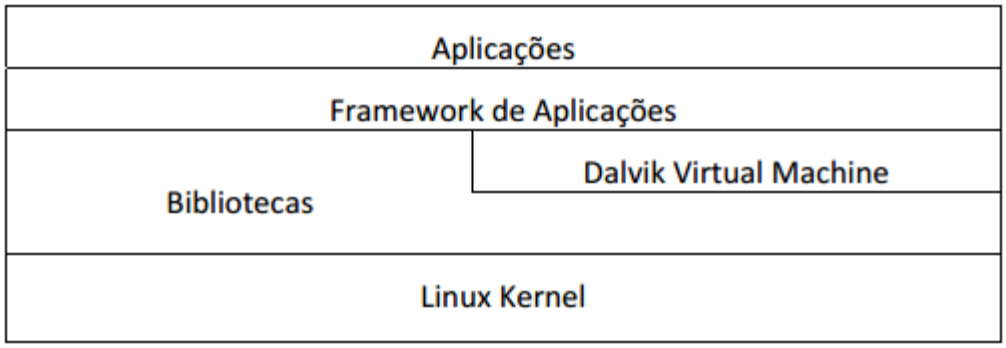

Figura 17: Arquitetura de aplicações Android [16].

O Android é executado sobre um Kernel Linux, que foi desenvolvido utilizando o sistema operacional Linux e é nele que encontramos os programas de gerenciamento de memória, configurações de segurança os muitos drivers de hardware. A máquina virtual é a Dalvik Virtual Machine, que é utilizada pelo Android para rodar cada aplicação com o seu próprio processo e é uma tecnologia de software livre. Um ponto positivo de ter processos independentes é que nenhuma aplicação é dependente da outra, uma parando não afetará as outras que estiverem rodando. Nas bibliotecas temos vários comandos que dizem ao dispositivo como interagir com os diferentes tipos de dados que ele lida. No framework de aplicações temos os programas que gerenciam as funções básicas do telefone. É no nível das aplicações que se encontram as funções básicas que são gerenciadas pelo framework da aplicação, é aonde a iteração do usuário com o dispositivo móvel é realizada através dos aplicativos instalados.

Após ter uma visão muito teórica da arquitetura do Android, será apresentada a estrutura de uma aplicação, com as partes mais importantes comentadas. A Figura 18 representa essa estrutura:

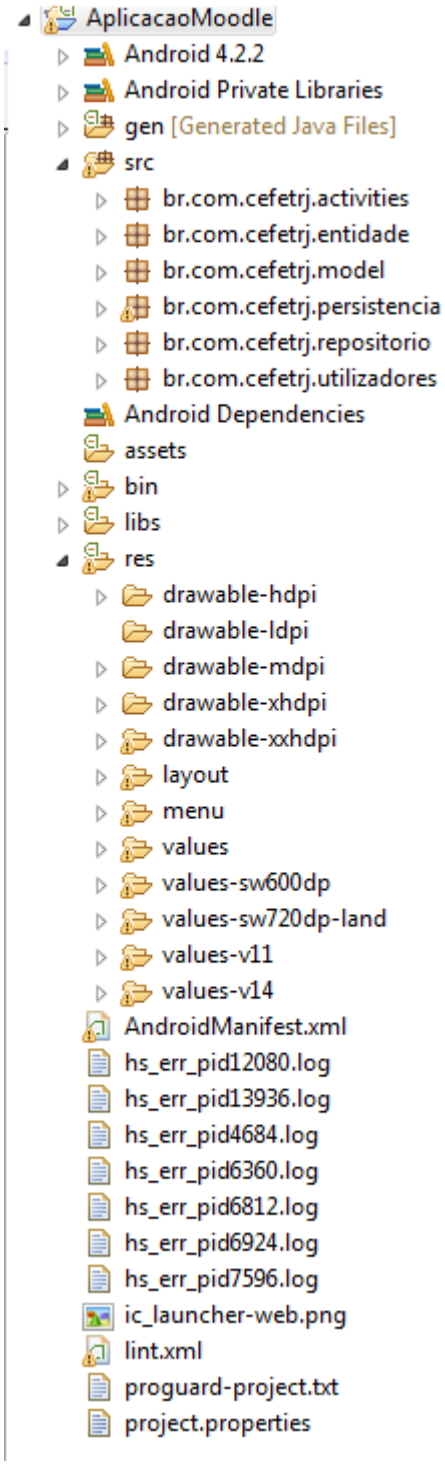

Figura 18: Estrutura de pastas de uma aplicação Android.

Os pontos de destaque são:

- src : aonde ficam os pacotes contendo as classes do sistema.
- res: aonde fica toda a parte gráfica de uma aplicação android (imagens, telas, string, etc).

- layout: pasta aonde ficam os arquivos xml, que representam as telas do sistema.
- AndroidManifest.xml: contém as informações de configuração necessárias para instalar a aplicação corretamente em um dispositivo. Nele são declaradas as classes, os eventos, permissões e etc.

A Figura 19 traz o código do arquivo AndroidManifest.xml:

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="br.com.cefetrj.activities"
    android:versionCode="1"
    android:versionName="1.0" >

    <uses-sdk
        android:minSdkVersion="8"
        android:targetSdkVersion="17" />

    <uses-permission
        android:name="android.permission.INTERNET" />

    <application
        android:allowBackup="true"
        android:icon="@drawable/ic_launcher"
        android:label="@string/moodle"
        android:theme="@style/AppTheme" >
        <activity
            android:name="br.com.cefetrj.activities.MainActivity"
            android:label="@string/moodle" >
        </activity>
         <activity
            android:name="br.com.cefetrj.activities.LoginActivity"
            android:label="@string/moodle" >
         </activity>
         <activity
            android:name="br.com.cefetrj.activities.ApresentacaoActivity"
            android:label="@string/moodle" >
             <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
         </activity>
         <activity
            android:name="br.com.cefetrj.activities.CadastroActivity"
            android:label="@string/moodle" >
         </activity>
         <activity
            android:name="br.com.cefetrj.activities.TelaPrincipalAlunoActivity"
            android:label="@string/moodle" >
```

Figura 19: Estrutura do arquivo AndroidManifest.xml.

Cada tag tem a sua explicação e a sua função, portanto, cada uma será devidamente analisada abaixo:

- uses-sdk: Indica qual a versão do SDK que está sendo utilizada, a versão mínima (minSdkVersion) e a versão utilizada na aplicação (targetSdkVersion).

- uses-permission: Dá permissão para a aplicação acessar a Internet.
- application: Aqui é a declaração da aplicação, declarando cada elemento.
- activity: Declara uma activty, que implementa parte da interface do usuário. Todas as activities devem ser declaradas, pois senão não serão visíveis e ocorrerá erro no sistema.
- intent-filter: Declara a activity que será a primeira a ser apresentada no sistema, é a sua activity inicial.

O AndroidManifest é o arquivo mais importante do sistema, ele é a base de uma aplicação android. Como já foi dito, se as informações não forem declaradas ou forem declaradas incorretamente nele, ocorrerá erro ao tentar rodar a aplicação. Antes de começar a documentar os arquivos xml que representam as telas do software, vamos falar um pouco sobre o padrão que escolhemos para desenvolver, que foi o MVC (Model, View e Controller).

O padrão escolhido para estruturar o nosso projeto foi o MVC, que foi escolhido por já termos familiaridade com ele e ser de fácil implementação. Sua principal característica é permitir dividir as funcionalidades do software em camadas, que são apresentadas logo a seguir:

- Model: usado para administrar as informações de uma forma mais detalhada, sendo requisitado, sempre que possível, para relizar consultas, cálculos e as regras de negócio. Tem acesso as informações que vem do banco de dados.
- View: Realiza a comunicação com o usuário, é responsável por tudo que o usuário final visualiza. Não deve ter lógica de negócio, o que fica para a camada de modelo.
- Controller: Responsável por controlar todo o fluxo de informação do sistema, pega a informação da camada de modelo e passa para a camada de visão, para ser visualizada pelo usuário.

A Figura 20 representa a arquitetura MVC para Android:

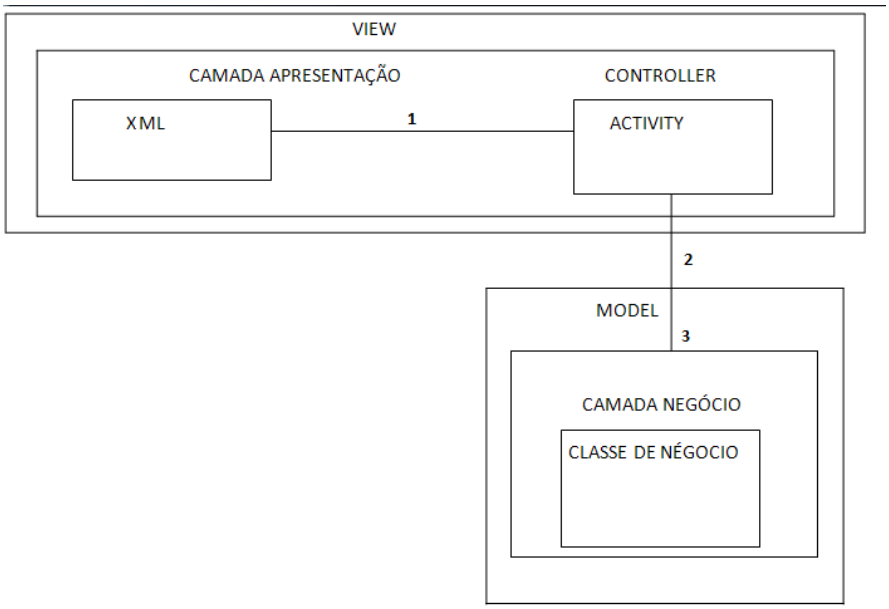

Figura 20: Representação da arquitetura MVC no Android [17].

Descrevemos o fluxo a seguir:

1) O usuário, interagindo com o XML, faz a requisição de um dado na aplicação. A activity, que faz referência a esse XML, é chamada e a interação com o controller é realizada.
2) O controller realiza a comunicação com o model para fazer a consulta que foi pedida.
3) O model recebe os dados, faz a validação, e, se estiver tudo certo, retorna o que foi pedido pelo usuário.

2.3.3 Procedimentos de Implementação

Após serem apresentadas a arquitetura e as camadas do Android e de uma aplicação Android, apresentamos os passos que foram seguidos para a implementação de uma tela do sistema, e que foram repetidos para as demais. Instalamos todos os programas que devem ser instalados e criamos um projeto de uma aplicação android. As classes professor, aluno, comunicado, disciplina, nota, solicitações, turma e usuário foram também criados no pacote "br.com.cefetrj.entidade". Na Figura 21, será apresentado o código de um documento XML

que representa uma tela do sistema, a tela escolhida foi a de alunos (mostra uma lista de alunos de uma determinada disciplina):

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="match_parent"
    android:layout_height="match_parent"
    android:orientation="vertical" >

    <ListView android:layout_width="fill_parent"
      android:layout_height="300dp"
      android:id="@+activity_alunos/listaDeAlunos">
    </ListView>

    <RelativeLayout
    android:layout_width="match_parent"
    android:layout_height="match_parent" >

        <Button
            android:id="@+activity_alunos/btnVerSolicitacoes"
            android:layout_width="150dp"
            android:layout_height="wrap_content"
            android:layout_marginTop="50dp"
            android:layout_centerHorizontal="true"
            android:text="@string/btnVerSolicitacoes"
            style="@style/letrasBotoes" />

    </RelativeLayout>

</LinearLayout>
```

Figura 21: Estrutura de uma tela Android xml.

O que não pode faltar em um arquivo XML que representa uma tela da aplicação Android é o LinearLayout, que representa uma forma linear de distribuir os elementos (botões, texto, etc.) dentro de uma tela, ou o RelativeLayout, aonde os elementos podem ser colocados em qualquer parte da tela sem uma distribuição linear, ou ambos. A ListView representa uma lista de alunos que vai ser preenchida se a disciplina tiver alunos. O botão é colocado dentro do RelativeLayout pois o seu alinhamento é colocado ao centro, o que pode também ser feito pelo LinearLayout, mas no RelativeLayout é mais fácil.

O próximo passo é criar uma classe no pacote “br.com.cefetrj.activities” para referenciar a tela que fora analisada. A Figura 22 representa essa classe:

```
package br.com.cefetrj.activities;

import java.util.ArrayList;

public class AlunosActivity extends Activity {

    Button btnVerSolicitacoes;
    private ListView lv;
    private ArrayAdapter<String> listAdapter ;
    ArrayList<String> listaAlunos = new ArrayList<String>();
    private Intent intent;
    DadosLogado perfilDados = new DadosLogado();
    private AlunoModel alunoModel;
    private TurmaModel turmaModel;
    private UsuarioModel usuModel;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_alunos);
```

Figura 22: Estrutura de uma Activity.

Devemos declarar essa activity no arquivo AndroidManifest.xml e logo depois colocar o “extends Activity” para essa classe herdar da classe “android.app.Activity”, a tornando uma atividade no Android. Após estender a classe, é obrigatório escrever o método “onCreate”, aonde será selecionado o arquivo XML que essa classe faz referência. Em “setContentView(R.layout.activity_alunos);” é feita essa associação, aonde activity_alunos é o nome do XML.

A Figura 23 mostra como um model é chamado a partir de uma activity, ainda usaremos a AlunosActivity para a demonstração:

```
public void PreencherListaAlunos() throws Exception{

    lv = (ListView) findViewById(R.activity_alunos.listaDeAlunos);

    listAdapter = new ArrayAdapter<String>(this, android.R.layout.simple_list_item_1, listaAlunos);
    turmaModel = new TurmaModel(getApplicationContext());
    alunoModel = new AlunoModel(getApplicationContext());
    usuModel = new UsuarioModel(getApplicationContext());

    ArrayList<Integer> listaIdsAluno = turmaModel.getIdsAlunosTurma(perfilDados.getIdDisciplinaSelecionada());
    final ArrayList<String> ListaCpfAlunos = new ArrayList<String>();
```

Figura 23: Estrutura de um método que chama uma classe modelo.

Na primeira linha do método, referenciamos a “listaDeAlunos” que é uma ListView do documento XML a ListView que foi declara nessa classe, a “lv”. Depois, criamos um ArrayAdapter, para atribuirmos um ArrayList de String ao “listAdapter”. Essas exlicações não tem muito a ver com a chamada do Model, e sim com o preenchimento da lista de alunos.

No “turmaModel = new(getApplicationContext());”, atribuímos ao Model, todos as configurações e recursos desta activity, para serem usados posteriormente. No trecho “turmaModel.getIdsAlunoTurma (perfilDados.getIdDisciplinaSelecionada());” é chamado um método da classe TurmaModel para obter os identificadores únicos dos alunos de uma determinada disciplina, passando como parâmetro, o Identificador da disciplina.
Na Figura 24, é mostrado esse método na classe TurmaModel:

```
package br.com.cefetrj.model;

import java.util.ArrayList;

public class TurmaModel {

    private Context context;
    private TurmaPersistencia turmaPersistencia;

    public TurmaModel(Context context) {

        this.context = context;
    }

    public ArrayList<Integer> getIdsAlunosTurma(int idDisciplina)
            throws Exception{

        turmaPersistencia = TurmaPersistencia.getInstance(context);
        return turmaPersistencia.getIdsAlunosTurma(idDisciplina);
    }
```

Figura 24: Estrutura de uma classe modelo.

Como o Identificador da disciplina vai estar sempre preenchido, pois para estar nessa tela o usuário tem que selecionar uma disciplina anteriormente, não tem a validação dentro do método. Em “turmaPersistencia = TurmaPersistencia.getInstance(context);” as configurações e recursos da Activity que chamou este método serão passadas para a classe TurmaPersistencia, que é responsável por fazer a comunicação com o banco de dados. No “return”, após for realizada a consulta no banco, o método retorna um array de inteiros para a Activity que chamou esse método. Obs: A classe TurmaModel foi criada no pacote ”br.com.cefetrj.model” e a classe TurmaPersistencia foi criada em “br.com.cefetrj.persistencia”.

Na Figura 25, é mostrada a classe TurmaPersistencia, que faz a comunicação com o banco de dados:

```
package br.com.cefetrj.persistencia;

import java.util.ArrayList;

public class TurmaPersistencia {

    private static SQLiteDatabase BancoDados;
    Cursor cursor;

    private Context context;
    private static TurmaPersistencia instance;

    public static TurmaPersistencia getInstance(Context context) {
        if (instance == null) {
            synchronized (TurmaPersistencia.class) {

                if (instance == null) {

                    instance = new TurmaPersistencia(context);

                    BancoDados = context.openOrCreateDatabase(
                            DatabaseCreate.TUTORIAL_DB, Context.MODE_PRIVATE,
                            null);
                }
            }
        }
        return instance;
    }

    private TurmaPersistencia(Context context) {

        this.context = context;

    }

    public ArrayList<Integer> getIdsAlunosTurma(int idDisciplina){
        ArrayList<Integer> listaAlunos = new ArrayList<Integer>();
        int id = 0;
        try {
            cursor = BancoDados.rawQuery("SELECT t.idAluno FROM turma t where t.idDisciplina = "+ idDisciplina +"", null);

             if  (cursor.moveToFirst()) {
                 do {
                     id = cursor.getInt(cursor.getColumnIndex("idAluno"));
                     listaAlunos.add(id);
                 }while (cursor.moveToNext());
             }

        } catch (Exception erro) {

        }
        return listaAlunos;
    }
```

Figura 25: Estrutura de uma classe de persistência.

No método getInstance, é verificado se a instância enviada do Model para essa classe está nula, se não estiver, uma conexão com o banco de dados é aberta, se já existir, ou criada, se não existir. No “getIdsAlunosTurma”, um array de inteiros é declarado para ser preenchido e ser retornado para a classe Model. Um inteiro chamado de “id” é declarado logo após o ArrayList. A consulta é realizada chamando a função “rawQuery”, que pertence a biblioteca denominada “SQLiteDataBase”, e passando a consulta SQL dentro desse método. O resultado dessa consulta é atribuído a um cursor, que é posteriormente utilizado para recuperar o valor de uma coluna. Dentro do while, os identificadores que forem sendo encontrados vão sendo adicionados a lista de inteiros, e, após preencher tudo, a lista é retornada para o Model, que retorna a lista para a Activity, continuando o código para preencher a lista de alunos.

Um método que foi muito utilizado no decorrer do desenvolvimento desta aplicação Android foi o de envio de e-mail nativo do Android, ele chama o aplicativo de e-mail do android, já preenchedo todos os campos devidamente, deixando para o usuário apenas a opção de enviar ou não o e-mail. A Figura 26 mostra esse método:

```
public void EnviarEmail(Nota nota) throws Exception{
    String email = null, body;

    usuModel = new UsuarioModel(getApplicationContext());
    Usuario usu = usuModel.PegarInformacoesAluno(perfilDados.getCpfAlunoSelecionado());
    email = usu.getEmail();
    body = "Uma nota foi lançada na disciplina " + perfilDados.getDisciplinaSelecionada() + ": \r\n \r\n  Nome nota: " +
            nota.getNomeNota() + " \r\n Nota: " + nota.getNota() + ". \r\n \r\n \r\n \r\n Mensagem enviada do aplicativo 'Moodle para Android'.";


    Intent i = new Intent(Intent.ACTION_SEND);
    i.setType("message/rfc822");
        i.putExtra(Intent.EXTRA_EMAIL  , new String[]{email});
    i.putExtra(Intent.EXTRA_SUBJECT, "Moodle para Android - Nota Lançada");
    i.putExtra(Intent.EXTRA_TEXT   , body);
    try {
        startActivity(Intent.createChooser(i, "Send mail..."));
    } catch (android.content.ActivityNotFoundException ex) {
        Toast.makeText(NotasLancarActivity.this, "Por favor, entre com sua conta de e-mail para poder enviar e-mail.", Toast.LENGTH_SHORT).show();
    }
}
```

Figura 26: Estrutura do método de enviar e-mail.

Esse método é chamado após o professor lançar uma nota para um determinado aluno. As informações do aluno são gravadas na variável “usu”, o e-mail do aluno é preenchido, o body que é o corpo do e-mail também. O assunto é preendhido em “Intent.EXTRA_SUBJECT”. Para finalizar, se o usuário não estiver logado com sua conta no aplicativo de e-mail do Android, ele não envia o e-mail e informa isso ao usuário. Achamos melhor ter essa opção de enviar ou não enviar o e-mail, pois o professor às vezes quer que o aluno demonstre interesse e entre no aplicativo para se informar melhor.

Para terminar a explicação de implementação do software, iremos mostrar algumas operações SQL de diferentes tipos que são usadas na aplicação. A Figura 27 representa o CREATE TABLE:

```
public void criarProfessor(){
    try {
        String SQL = "CREATE TABLE IF NOT EXISTS professor (_id integer PRIMARY KEY AUTOINCREMENT, cpf TEXT)";
        BancoDados.execSQL(SQL);
    } catch (Exception erro) {

    }
}
```

Figura 27: Estrutura de uma operação CREATE.

A Figura 28 representa o INSERT:

```
public void NovoAluno(String cpf){
    try {
        String SQL = "INSERT INTO aluno (cpf) VALUES ('"+ cpf +"')";
        BancoDados.execSQL(SQL);

    } catch (Exception erro) {

    }
}
```

Figura 28: Estrutura de uma operação INSERT.

A Figura 29 representa o UPDATE:

```
public void AtualizarDisciplina(int idDisciplina, int idProfessor, String abreviacao, String nomeDisciplina, String instituicao){
    try {

        ContentValues args = new ContentValues();
        args.put("abreviacao", abreviacao);
        args.put("nomeDisciplina", nomeDisciplina);
        args.put("instituicao", instituicao);

        BancoDados.update("disciplina", args, "_id = " + idDisciplina + " and idProfessor = " + idProfessor + "", null);
    } catch (Exception erro) {

    }
}
```

Figura 29: Estrutura de uma operação UPDATE.

No ContentValues, passamos os campos que queremos atualizar e dentro da função “update”, passamos os campos que são utilizados no WHERE, no caso é o idDisciplina e o idProfessor. A Figura 30 representa o DELETE:

```
public void ApagarDisciplina(int idDisciplina){
    try {
        BancoDados.delete("disciplina", "_id = " + idDisciplina + "", null);
    } catch (Exception erro) {

    }
}
```

Figura 30: Estrutura de uma operação DELETE.

Dentro da função “delete”, passamos os campos que são utilizados no WHERE para excluir uma disciplina.

Todos os passos para a implementação de um aplicativo Android foram devidamente explicados nessa seção e, na próxima, apresentaremos a conclusão do documento.

Capítulo 3

3. Conclusão

Conclui-se que a aplicação, mesmo com menor grau de complexidade, poderá ter uma boa contribuição principalmente para área educacional atuando na síntese e absorção do conhecimento necessário, ainda como importante mecanismo da EAD, podendo se tranformar também em uma ferramenta complementar do ensino direto nas salas de aula.

O software deve contribuir na interação aluno-professor de todo o modo, seja presencial, não presencial, síncrono ou assíncrono. Sua utilização em dispositivos móveis, fornece ferramentas para aumentar de maneira significativa a aprendizagem nas disciplinas solicitadas. Além disso, professores podem desfrutar de maior eficiência na comunicação decorrente deste canal adicional com seus alunos, desta forma, a aplicação poderá agregar aos usuários a facilidade de se manter conectado à sala de aula virtual sem restrições de horário e local – característica do conceito de ubiquidade potencializado principalmente pela evolução dos aparelhos e tecnologias de acesso a Internet.

Todo software tem suas falhas e qualidades, e, neste parágrafo, falaremos um pouco sobre o que falta para este aplicativo tornar-se melhor. O software peca um pouco na usabilidade, na falta de alguns atalhos adicionais e a criação de um menu iria favorecer muito a utilização do mesmo por seus usuários. Na seção de trabalhos futuros (3.2), serão apresentadas algumas melhorias para as futuras versões deste aplicativo.

Durante todo o processo de desenvolvimento foram realizados pesquisas e estudos visando aprofundar o desenvolvimento de aplicações em ambiente Android. A linguagem Java, dominada pelos desenvolvedores deste software, nunca havia sido usada desta maneira na elaboração de um aplicativo Android .

Sem dúvida, as escolhas feitas antes do desenvolvimento deste software obtiveram sucesso e serão analisadas na próxima seção.

3.1 Considerações Finais

Dentre os objetivos que foram estabelecidos no ínicio do desenvolvimento deste software, destaca-se a navegabilidade do software com consequente facilidade na utilização do software. A utilidade da aplicação superou expectativas demonstrando alto potencial de aceitação e interesse sobre o tema apresentado diante de professores, alunos e instituições educacionais.

Ainda assim, como outro item do objetivo inicial que obteve sucesso, houve contribuição relevante ao acúmulo de informações no desenvolvimento de softwares através da linguagem de programação Java, em ambiente mobile, ou seja, em ambiente para dispositivos móveis, principalmente smartphones, utilizando como ferramenta para o desenvolvimento o Eclipse. Levando em consideração a plataforma para a qual foi desenvolvida a aplicação, destaca-se outra contribuição relevante no desenvolvimento de aplicativos para ambiente Android que, mesmo dominando o respectivo segmento de mercado, necessita de aplicativos educacionais com foco em aprendizagem em ambiente virtual.

3.2 Trabalhos Futuros

A evolução do software torna-se tão importante quanto o seu desenvolvimento, portanto, algumas melhorias são necessárias para o aprimoramento desta aplicação. Ainda que haja possibilidade no surgimento de softwares complementares, primeiramente, almeja-se o desenvolvimento de novas versões ou simplesmente atualizações cotidianas. Aqui, serão apresentadas algumas dessas melhorias.

Além da tradução do software em novos idiomas existe a necessidade de fórum para discursão de idéias e de dúvidas entre professores e alunos. Ainda assim, um chat online para interação síncrona pode ser desenvolvido para auxiliar e facilitar o entendimento ou tratamento de casos mais específicos por disciplina. Pode ser melhorado ainda na otimização e na implantação de alertas através de notificações, e-mail e pop-ups com uma maior usabilidade do software e com uma maior interação com a disciplina. Deste modo, é

extremamente importante direcionarmos os recursos disponíveis para o aprimoramento da aplicação gerada que demonstra alto potencial em assunto pouco explorado.

Outras melhorias seriam poder realizar o lançamento de notas para todos os alunos inscritos em uma disciplina em apenas uma tela do sistema e a implantação do sistema de notificações, avisando em tempo real sempre que alguma nota for lançada ou comunicado for emitido em uma disciplina. Colocar um calendário em que ficassem marcadas as datas de prova e permitir aos usuários anexar e visualizar arquivos também seriam algumas melhorias.

Referências

[1] A. Caldeira, "**Avaliação da aprendizagem em meios digitais: novos contextos**", Avaliação da aprendizagem em meios digitais: novos contextos, p. 12, abr. 2012.Disponível em:<https://www2.ufmg.br/ead/content/download/9706/70559/file/CALDEIRA.pdf>. Acesso em: 15dez. 2012.

[2] T. Serrão, L. Braz, S. Pinto, e G. Clunie, "**Construção Automática de Redes Sociais Online no Ambiente Moodle**", UNISINOS - Universidade do Vale do Rio dos Sinos, Rio Grande do Sul, Brasil, 2011.Disponível em: <http://br-ie.org/pub/index.php/sbie/article/view/1647/1412>. Acesso em: 15dez. 2012.

[3] L. Meirelles e L. Tarouco, "**Framework para Aprendizagem com Mobilidade**", UFRGS - Universidade Federal do Rio Grande do Sul, Rio Grande do Sul, Brasil, 2005. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/446/432>. Acesso em: 27dez. 2012.

[4] F. Franciscato e R. Medina, "**M-Learning e Android: um novo paradigma?**", UFSM - Universidade Federal de Santa Maria, Rio Grande do Sul, Brasil, 2008. Disponível em: <http://seer.ufrgs.br/renote/article/view/14671/8580>. Acesso em: 27dez. 2012.

[5] Figura 1 Disponível em: <http://en.wikipedia.org/wiki/File:Overview_of_a_three-tier_application_vectorVersion.svg > Acesso em agosto de 2013.

[6] L. Oliveira e R. Medina, "**Desenvolvimento de objetos de aprendizagem para dispositivos móveis: uma nova abordagem que contribui para a educação.**", UFSM - Universidade Federal de Santa Maria, Rio Grande do Sul, Brasil. Disponível em: <http://lumenagencia.com.br/dcr/arquivos/desenvolvimento_de_objetos_de_aprendizagem_para_dispositivos_m%F3veis.pdf>. Acesso em: 21 dez. 2012.

[7] L. Tarouco, M.-C. Fabre, M. Konrath, e A. Grando, "**Objetos de Aprendizagem para M-Learning**", Sucesu. Disponível em: <http://objectosaprendizagem.no.sapo.pt/pdf/objetosdeaprendizagem_sucesu.pdf>. Acesso em: 5 jan. 2013.

[8] M. Quinta e F. Lucena, "**Odin - Viabilizando e-learning em múltiplos dispositivos**", Universidade Federal de Goiás, Goiânia, GO, 2007. Disponível em: <http://www.br-ie.org/WIE2010/pdf/sp01_07.pdf>. Acesso em: 5 jan. 2013.

[9] M. Quinta e F. Lucena, "**Problemas e soluções em u-learning e a adaptacão de conteúdo de objetos de aprendizagem para diferentes dispositivos**", Universidade Federal de Goiás, Goiânia, GO, 2007. Disponível em: <http://br-ie.org/pub/index.php/sbie/article/view/1501/1266>. Acesso em: 10 jan. 2013.

[10] J. Rodrigues, "**Uso de m-learning no Ensino Superior**", Dissertação, Universidade de Aveiro, Aveiro, Portugal, 2007. Disponível em: <http://ria.ua.pt/bitstream/10773/1533/1/2008001205.pdf>. Acesso em: 12 jan. 2013.

[11] P. Ribeiro, F. Franciscato, P. Mozzaquatro, e R. Medina, "**Validação de um Ambiente de Aprendizagem Móvel em Curso a Distância**", UFSM - Universidade Federal de Santa Maria, Rio Grande do Sul, Brasil. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/1153/1056>. Acesso em: 12 jan. 2013.

[12] L. Silva, F. Neto, e L. Júnior, "**MobiLE: Um ambiente Multiagente de Aprendizagem Móvel para Apoiar a Recomendação Sensível ao Contexto de Objetos de Aprendizagem**", Universidade Federal Rural do Semi-Árido, Mossoró, Rio Grande do Norte, 2011. Disponível em: <http://ceie-sbc.educacao.ws/pub/index.php/sbie/article/view/1593/1358>. Acesso em: 3jan. 2013.

[13] A. Mühlbeier, P. Mozzaquatro, R. Medina, R. Moreira, e R. Antoniazzi, "**MOBILE HQ: O USO DE SOFTWARES EDUCATIVOS NA MODALIDADE M-LEARNING**", UFSM - Universidade Federal de Santa Maria, Rio Grande do Sul, Brasil, 2012. Disponível em: <http://www.br-ie.org/pub/index.php/sbie/article/view/1742/1503>. Acesso em: 3jan. 2013.

[14] K. Fernandes, G. Trindade, B. Souza, A. Gomes, e M. Lucena, "**Question Mobile: Ampliando Estratégias de Avaliação da Aprendizagem por Meio de Dispositivos Móveis**", UFRN - Universidade Federal do Rio Grande do Norte, Natal, RN - Brasil, 2012. Disponível em: <http://ceie-sbc.tempsite.ws/pub/index.php/wcbie/article/view/1940/1700>. Acesso em: 10jan. 2013.

[15] B. Orlandi e S. Isonati, "**Uma Ferramenta para Distribuição de Conteúdo Educacional Interativo em Dispositivos Móveis**", USP - Universidade de São Paulo, São Paulo - Brasil, 2012. Disponível em: <http://br-ie.org/pub/index.php/sbie/article/view/1792/1553>. Acesso em: 10jan. 2013.

[16] Figura 37 Traduzida de: <http://www.redrails.com.br/2011/04/18/google-android-como-um-ambiente-de-desenvolvimento-de-aplicacoes-para-sistemas-de-decodificacao-de-dtv-tv-digital/> Acesso em julho de 2013.

[17] Figura 40 Disponível em: <http://mobilidade.fm/tag/mvc-para-android/> Acesso em julho de 2013.

Apêndice I: Telas do sistema

A Figura 31 representa a tela de cadastro:

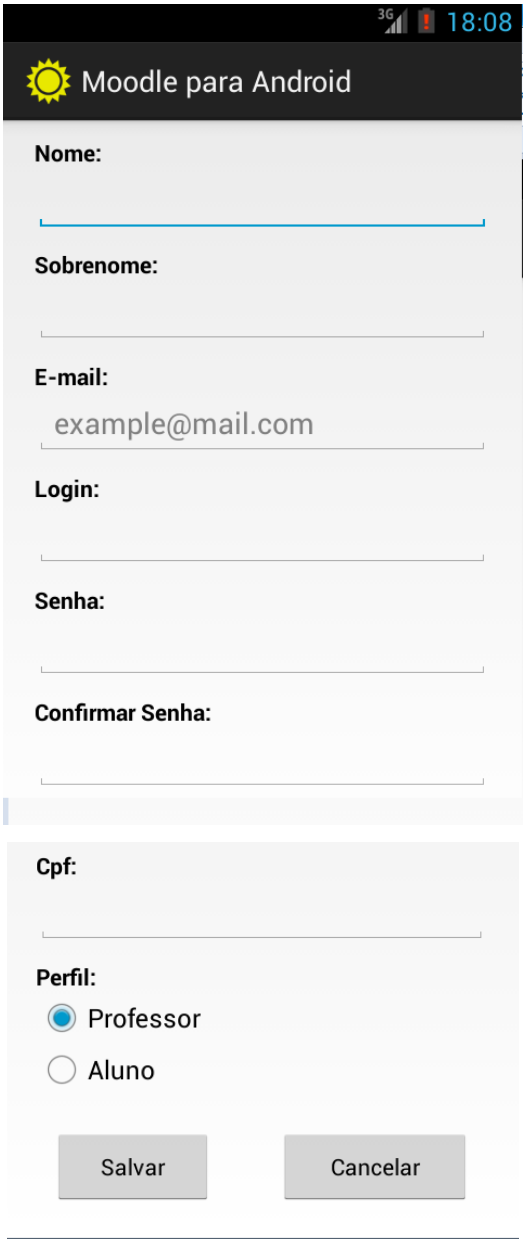

Figura 31: Tela de cadastro.

Permite que uma pessoa que queira acessar o sistema crie o seu cadastro para posteriormente realizar o seu login.

A Figura 32 representa a tela de login:

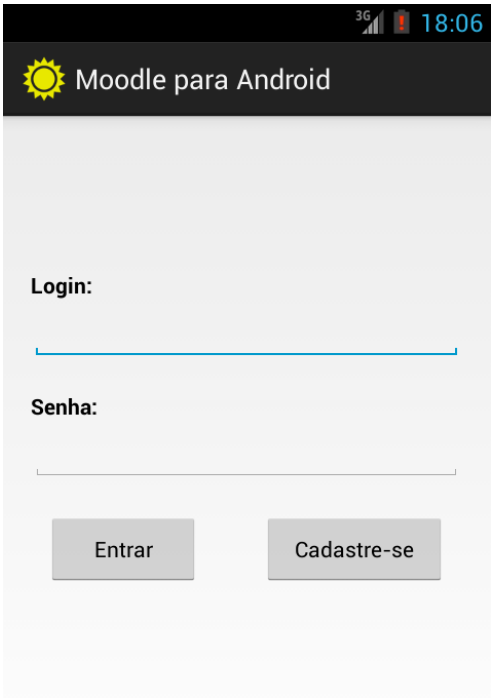

Figura 32: Tela de login.

Tela inicial do sistema que permite ao usuário realizar login se o mesmo já estiver cadastro no sistema.

A Figura 33 representa a tela inicial do aluno:

Figura 33: Tela inicial do aluno.

Após realizar o login, se o perfil do usuário for de aluno, esta tela será apresentada a ele. Nela, ele poderá clicar em “Minhas Disciplinas” ou “Pesquisar”.

A Figura 34 representa a tela inicial do professor:

Figura 34: Tela inicial do professor.

Após realizar o login, se o perfil do usuário for de professor, esta tela será apresentada à ele. Nela, ele poderá clicar em “Minhas Disciplinas”, “Pesquisar” ou “Criar Disciplina”.

A Figura 35 representa a tela de minhas disciplinas:

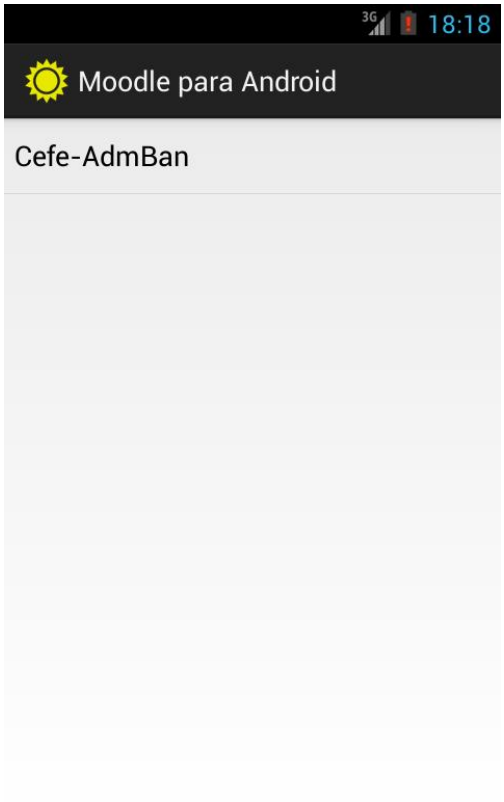

Figura 35: Tela de minhas disciplinas.

Tela que apresenta as disciplinas cursadas pelo aluno ou as disciplinas lecionadas pelo professor.

A Figura 36 representa a tela de disciplina selecionada do aluno:

Figura 36: Tela de disciplina selecionada do aluno.

Após selecionar a disciplina, o sistema conduz o aluno para esta tela, na qual ele poderá visualizar os comunicados desta disciplina, ver suas notas e cancelar sua inscrição.

A Figura 37 representa a tela de disciplina selecionada do professor:

Figura 37: Tela de disciplina selecionada do professor.

Após selecionar a disciplina, o sistema conduz o professor para esta tela, na qual ele poderá realizar algumas funções.

A Figura 38 representa a tela de comunicados do aluno:

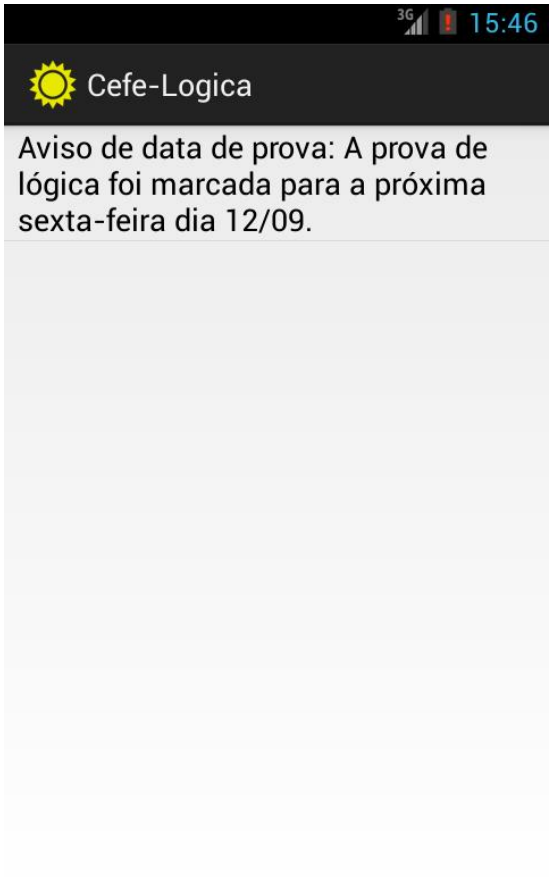

Figura 38: Tela de comunicados do aluno.

Após clicar em comunicados, o sistema apresentará esta tela para o aluno com uma lista dos comunicados da disciplina.

A Figura 39 representa a tela de comunicados do professor:

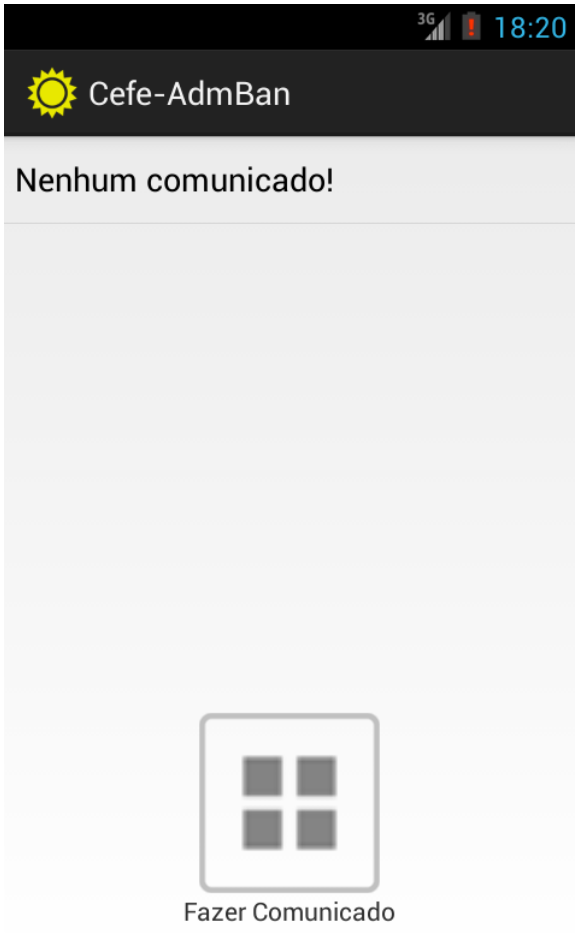

Figura 39: Tela de comunicados do professor.

Após clicar em comunicados, o sistema apresentará esta tela para o professor, uma lista com os comunicados da disciplina e um botão.

A Figura 40 representa a tela de fazer comunicado:

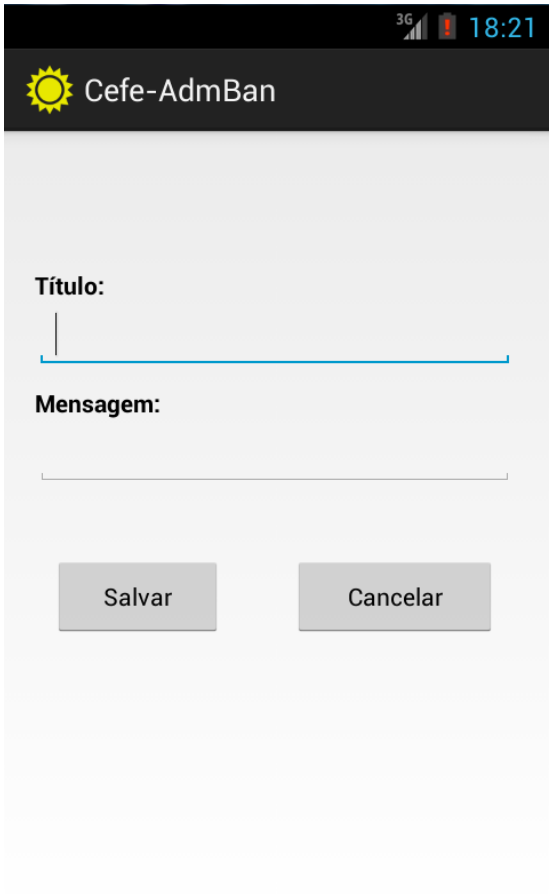

Figura 40: Tela de fazer comunicado.

Após clicar em “Fazer Comunicado”, o sistema conduz o professor para esta tela de envio de comunicados aos alunos desta disciplina.

A Figura 41 representa a tela de notas do aluno:

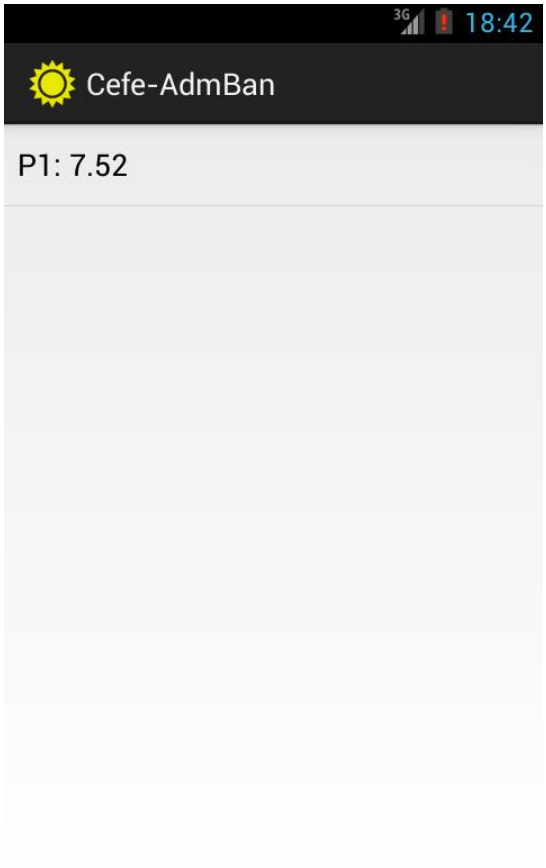

Figura 41: Tela de notas do aluno.

Após clicar em notas, o sistema apresentará esta tela para o aluno com uma lista das notas da disciplina.

A Figura 42 representa a tela de criar disciplina:

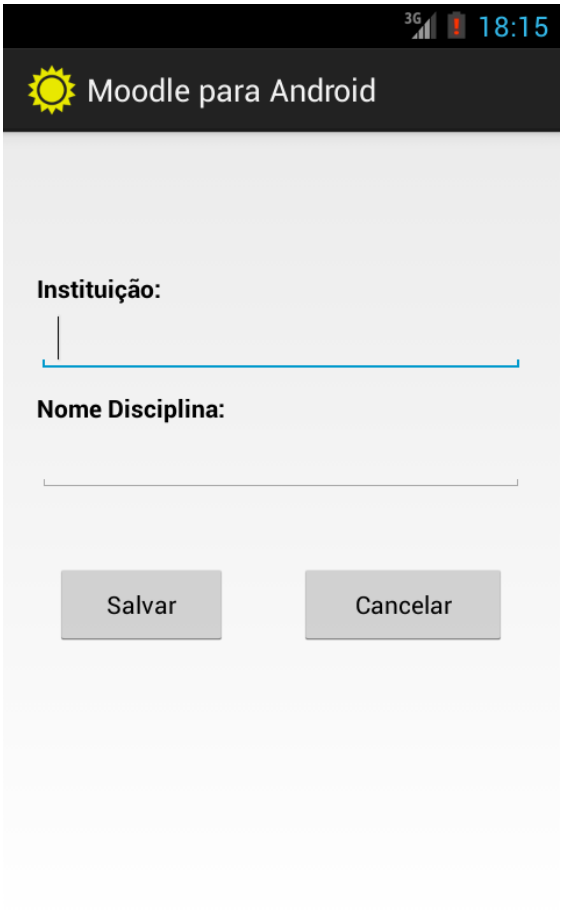

Figura 42: Tela de criar disciplina.

Após clicar em “Criar Disciplina”, o sistema conduz o professor para esta tela.

A Figura 43 representa a tela de pesquisar disciplina:

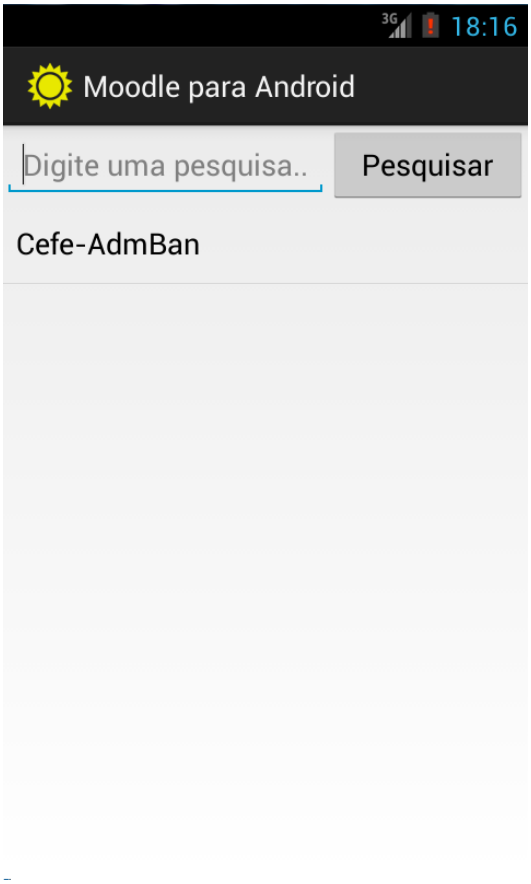

Figura 43: Tela de pesquisar disciplina.

Após o professor ou o aluno clicarem em “Pesquisar Disciplina”, o sistema os conduz para esta tela, aonde podem ser realizadas buscas com o objetivo de encontrar uma disciplina específica.

A Figura 44 representa a tela de visualizar informações da disciplina vista pelo aluno:

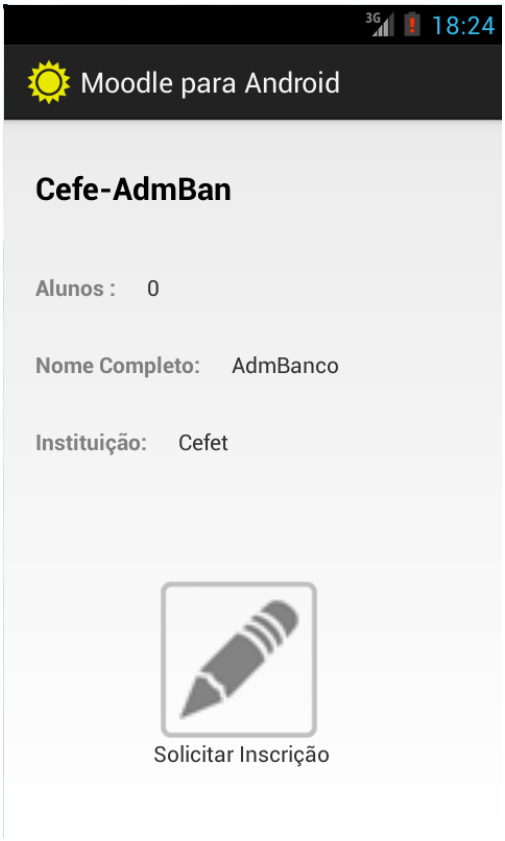

Figura 44: Tela de visualizar informações da disciplina vista pelo aluno.

Após buscar uma disciplina e selecionar ela, o sistema conduz o aluno para esta tela. Algumas informações importantes sobre a disciplina são mostradas.

A Figura 45 representa a tela de visualizar informações da disciplina vista pelo professor:

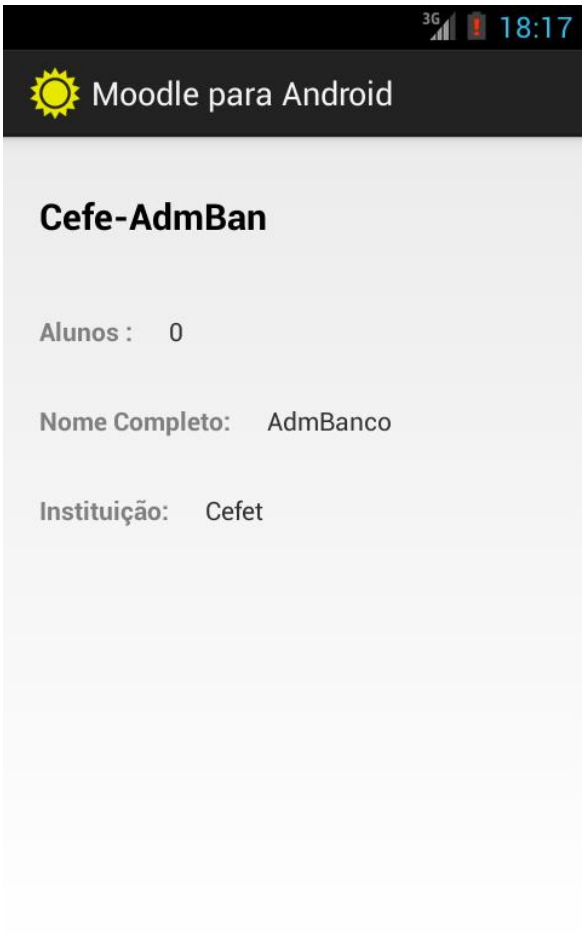

Figura 45: Tela de visualizar informações da disciplina vista pelo professor.

Após buscar uma disciplina e selecionar ela, o sistema conduz o professor para esta tela. Algumas informações importantes sobre a disciplina são mostradas.

A Figura 46 representa a tela de editar disciplina:

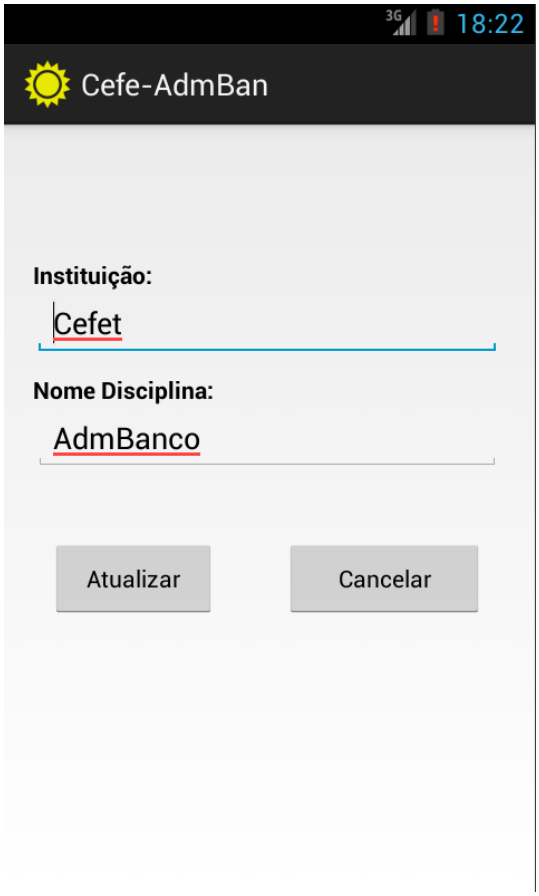

Figura 46: Tela de editar disciplina.

Após clicar em “Editar Disciplina”, o sistema conduz o professor para esta tela.

A Figura 47 representa a tela de alunos de uma disciplina:

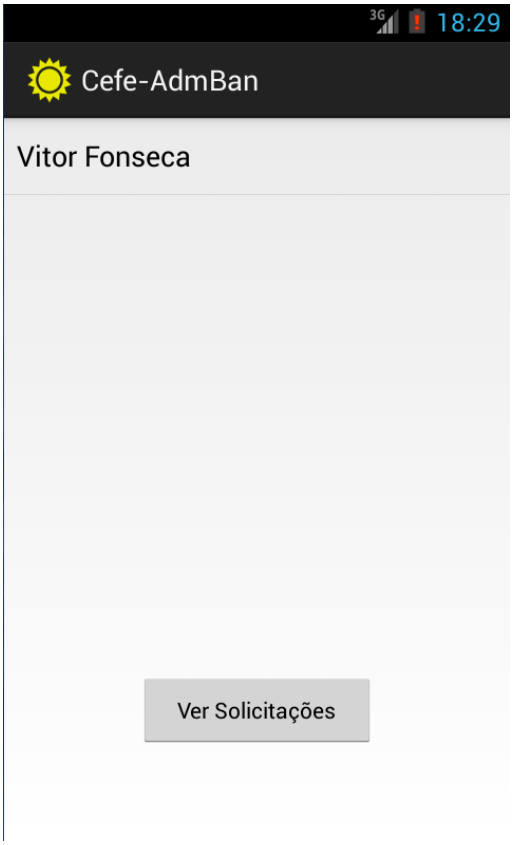

Figura 47: Tela de alunos de uma disciplina.

Quando o professor clica no botão “Alunos” na tela de disciplina selecionada, o sistema conduz ele para esta tela.

A Figura 48 representa a tela de informações de um aluno:

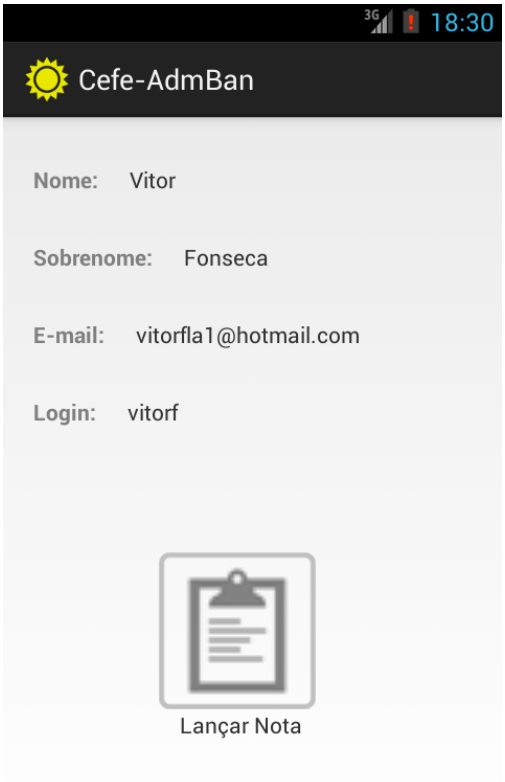

Figura 48: Tela de informações de um aluno.

Após clicar no nome do aluno na tela de alunos, o sistema apresenta esta tela ao professor. Ele poderá ver informações de cadastro do aluno como nome, sobrenome, e-mail e login.

A Figura 49 representa a tela de lançar nota:

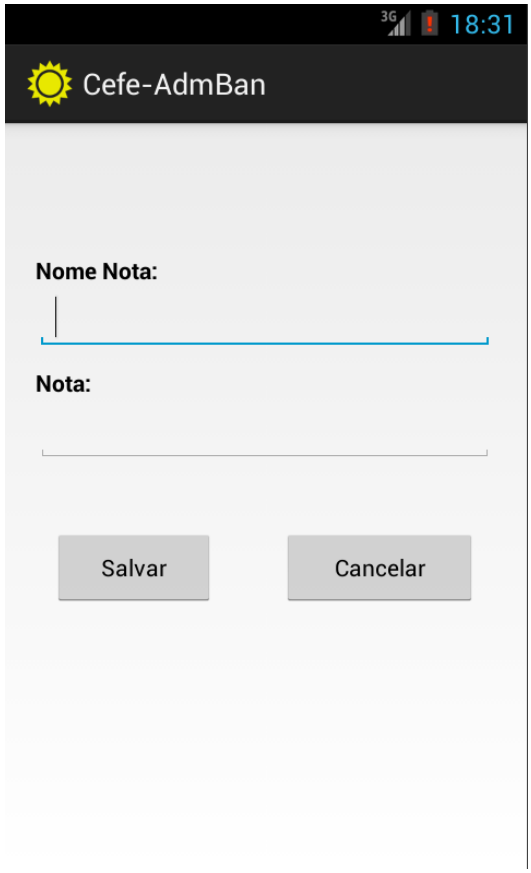

Figura 49: Tela de lançar nota.

Após clicar em “Lançar Nota”, o professor é apresentado à esta tela.

A Figura 50 representa a tela de visualização de solicitações de uma disciplina:

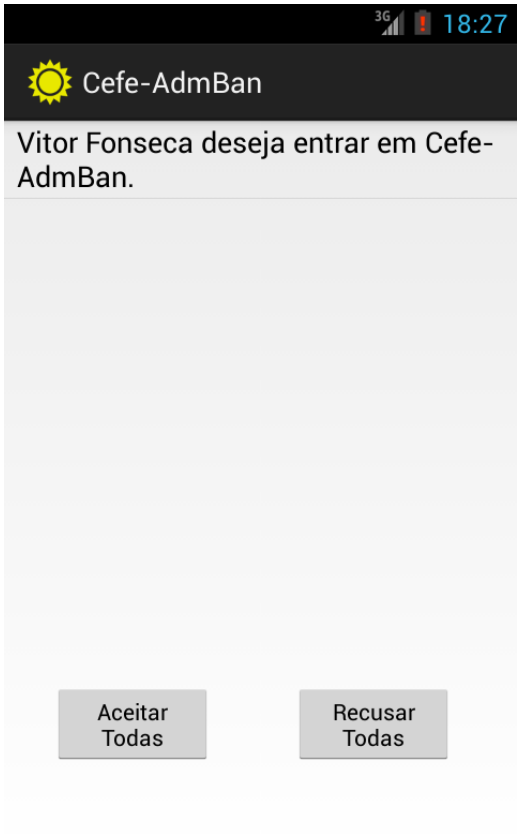

Figura 50: Tela de visualização de solicitações de uma disciplina.

Após clicar no botão “Ver Solicitações” na tela de Alunos, o sistema conduz o professor para esta tela. Nela, poderão ser aceitos ou recusados os pedidos de inscrição em uma determinada disciplina pelo professor.