UNIVERSIDADE ESTADUAL DO NORTE DO PARANÁ

***CAMPUS* LUIZ MENEGHEL**

LUCAS HENRIQUE FERNANDES

DESENVOLVIMENTO SOBRE A PLATAFORMA ANDROID DE UMA APLICAÇÃO PARA GESTÃO FINANCEIRA DE REPÚBLICAS DE ESTUDANTES

Bandeirantes

2014

LUCAS HENRIQUE FERNANDES

DESENVOLVIMENTO SOBRE A PLATAFORMA ANDROID DE UMA APLICAÇÃO PARA GESTÃO FINANCEIRA DE REPÚBLICAS DE ESTUDANTES

Monografia apresentada à Universidade Estadual do Norte do Paraná – *campus* Luiz Meneghel – como requisito parcial para aprovação no Curso de Sistemas de Informação.

Orientador: Prof. Rodrigo Tomaz Pagno

Bandeirantes

2014

LUCAS HENRIQUE FERNANDES

DESENVOLVIMENTO SOBRE A PLATAFORMA ANDROID DE UMA APLICAÇÃO PARA GESTÃO FINANCEIRA DE REPÚBLICAS DE ESTUDANTES

Monografia apresentada à Universidade Estadual do Norte do Paraná – *campus* Luiz Meneghel – como requisito parcial para aprovação no Curso de Sistemas de Informação.

COMISSÃO EXAMINADORA

_____
Prof. Me. Rodrigo Tomaz Pagno
UENP – *Campus* Luiz Meneghel

_____
Prof. Me. André Luis Andrade Menolli
UENP – *Campus* Luiz Meneghel

_____
Prof. Me. Glauco Carlos Silva
UENP – *Campus* Luiz Meneghel

Bandeirantes, 13 de Junho de 2014

AGRADECIMENTOS

Agradeço a todos que contribuíram para a conclusão deste trabalho, de forma direta ou indireta. Agradecimentos especiais para o professor Rodrigo Tomaz Pagno, pela orientação e grande participação no desenvolvimento deste trabalho, e aos membros da banca de correção, professor Glauco Carlos Silva e professor André Andrade Menolli, pela ajuda em diversos pontos e compreensão em diversas fases do trabalho. Agradecimentos também ao professor Estevan Braz Brandt Costa, pela ajuda de diversas maneiras. Outro agradecimento em especial para meus familiares e minha companheira Rafaela Rossatto, por toda força e apoio para comigo durante todas as fases deste trabalho, obrigado.

RESUMO

A formação em um curso do ensino superior se tornou essencial para a maioria dos jovens atualmente, isso ocasiona a migração de muitas pessoas para centros universitários buscando uma universidade de renome com o curso desejado, que possa não se encontrar em sua região. A aglomeração de estudantes nos centros universitários facilita a criação das repúblicas de estudantes, moradias onde convivem principalmente estudantes em busca da diminuição dos custos de vida, no entanto, muitos destes jovens não têm experiência na administração de uma moradia, portanto, possuem dificuldades nessa nova fase de sua vida. A diferença de rotinas entre os moradores da casa pode gerar um desencontro, que juntamente com outros fatores dificultam a comunicação das dívidas de sua residência. Visando resolver estes problemas, é proposto neste trabalho à criação de um aplicativo que auxilie os moradores de repúblicas em sua administração, o aplicativo tem o propósito de armazenar e facilitar a comunicação das informações entre os moradores, e calcular as despesas automaticamente. Assim, deixando de modo mais claro e de fácil acesso para todos integrantes a situação financeira de sua moradia. O aplicativo foi desenvolvido na plataforma Android, visando os dispositivos móveis, pretendendo deixar de forma mais dinâmica o uso da aplicação. O Android possui código aberto e um vasto número de ferramentas no auxílio do desenvolvimento como bibliotecas, *frameworks* de aplicação, entre outras. Para o desenvolvimento foi feito o levantamento dos requisitos perante a análise do ambiente de uma república estudantil, utilizando o método de etnografia seguido de entrevista e questionário aplicado aos futuros usuários. Foi desenvolvido uma base de dados seguindo o modelo de entidade relacionamento para um melhor tratamento das informações inseridas no software, seguido de sua implementação e teste de uso utilizando fatores contidos no método de mcCall de qualidade de software.

Palavras-chave: Andorid; Programação; Ferramentas de Administração; Repúblicas de Estudantes.

ABSTRACT

Training on a course of higher education has become essential for most young people today , it causes the migration of many people to universities seeking a reputed university or your course , you cannot find in your region . The clustering of students in universities facilitates the creation of the Republics of students , mainly houses where students live together in search of lower costs of living , however , many of these young people have no experience in managing a housing therefore have difficulties in this new phase of his life , as the difference of routines before the residents of the house where you can generate a mismatch , hindering communication debts from his residence . Aiming to solve these problems , is proposed in this paper to create an application that helps residents of republics in its administration, the application will aim to store and facilitate the communication of information among residents , and calculate the cost automatically. Thus leaving more clearly and easily accessible to all members of the financial situation of your house . The application was developed on the Android platform , targeting mobile devices , intending to leave more dynamically use the application . The Android open source and has a vast number of tools to aid the development as libraries , application frameworks , among others . For the development of the survey requirements before the analysis of a student Republic setting was done using the method of ethnography followed by interview and questionnaire applied to future users . A database on the model of entity relationships for better processing of information entered into the software , followed by implementation and use testing using factors contained in the method of McCall software quality was developed .

Keywords: Andorid; Programming; Administration Tools; Students Republics.

TABELA DE SIGLAS

ADT	Android Development Tool
API	Application Programming Interface
AVD	Android Virtual Device
CVS	Competitor Version System
IDE	Integrated Development Enviroment
JDK	Java Development Kit
JVM	Java Virtual Machine
DVM	Dalvik Virtual Machine
OHA	*Open Handset Alliance*
OMA	*Open Mobile Alliance*
SDK	*Software Development Kit*
SGBD	Sistema Gerenciador de Banco de Dados
SGL	Scalated Graphical Library
SQL	Structured Querry Language
SMS	Short Message Service

LISTA DE FIGURAS

SUMÁRIO

1 INTRODUÇÃO

A utilização de residências como repúblicas estudantis existe há muito tempo. Segundo Fernandes *et al.* (2003), as acomodações surgiram em 1088 com a criação da Universidade de Bolonha, na Itália. Estas acomodações são ainda mais populares atualmente. No Brasil o surgimento das acomodações não é preciso e há controvérsia de alguns autores, como será citada no decorrer do trabalho.

O fator primordial para este trabalho é a administrações destes recintos estudantis. Segundo Barre (1963), a administração deriva-se do estudo do comportamento humano referente a seu ambiente externo, visando suprir seus desejos e suas necessidades levando em consideração seus meios para isso. Noutra visão, Malanos (1967), diz que a economia esta relacionada com a produção e distribuição dos bens escassos para a satisfação das necessidades. Tendo essa duas visões conclui-se que uma boa administração de modo geral está em um limiar entre seus desejos, necessidades e meio externo ao indivíduo. Uma república estudantil, segundo Sardi (2000), tem um ambiente considerado de exageros, como será mostrado no decorrer do trabalho. Assim constata-se que o ambiente que cerca os estudantes pode não ser o mais propício para se obter uma boa administração de suas moradias.

Neste contexto, este trabalho propõe a criação de um software para facilitar a administração de repúblicas estudantis. Para melhor mobilidade do usuário o aplicativo foi desenvolvido para Android, uma plataforma para dispositivos móveis. O Android contem código aberto, seu sistema operacional é baseado em Linux, utiliza a linguagem JAVA oferece uma plataforma que auxilia o desenvolvimento com uma vasta gama de ferramentas de suporte, além de bibliotecas e *frameworks*. (LECHETA, 2009).

1.1 Objetivos

Neste tópico serão apresentados os objetivos do trabalho, separados em objetivo geral e objetivos específicos.

1.1.1 Objetivo Geral

O trabalho tem como objetivo geral o desenvolvimento de um software sobre a plataforma Android, para auxiliar os moradores de repúblicas de estudantes a administrarem financeiramente suas residências.

Assim a ferramenta deve executar as funções de armazenamento de informações, cálculo da divisão das dívidas, e principalmente facilitar a comunicação entre os usuários, não necessitando estarem no mesmo local para troca de informações.

1.1.2 Objetivos específicos

Os objetivos específicos se dividem entre a parte de pesquisa e a de desenvolvimento:

- Pesquisar e analisar ferramentas correlatas que auxiliem na administração;
- Levantar os requisitos necessários para o desenvolvimento do software;
- Desenvolver um módulo para a comunicação da ferramenta por meio de troca de emails;
- Implementar um módulo para os cálculos financeiros;
- Criar um módulo de cadastro para a casa e os integrantes (usuários);
- Desenvolver um banco capaz de armazenar as informações necessárias;
- Validar os requisitos.

1.2 Justificativa

Devida a crescente procura pelos jovens de concluir um curso superior e a falta de universidades em todas as cidades e regiões, a criação de repúblicas aumenta inevitavelmente. Assim, a elaboração de uma ferramenta para o auxílio da administração das moradias é de grande ajuda para que, não haja perca de informações, dificuldade de comunicação e nem desigualdade na divisão das dívidas.

Desta maneira, a administração passa a ser realizado por todos integrante da república, compartilhando um gerenciamento mais honesto e atualizado, além de não haver a necessidade dos membros estarem próximos para a troca de informações. A

tecnologia móvel destacou diferentes formas de se trocar informação entre softwares, como troca por SMS (Short Message Service), por email, entre outras. Este trabalho selecionou a troca por email para sua metodologia de comunicação, para melhor estudar um destes diferentes métodos de troca de informação, também é destaque a não necessidade do usa de um servidor.

1.3 Método

Este trabalho visa à análise do desenvolvimento para a plataforma Android através da realização de uma pesquisa classificada como exploratória (LAKATOS, 2010), bem como do tipo bibliográfica devido à necessidade de um levantamento bibliográfico para o entendimento do assunto e do ambiente em que esta ferramenta foi desenvolvida, o trabalho contém natureza aplicada devido à utilização de questionário e entrevista para o levantamento de requisitos (VILAÇA, 2010).

A aplicação foi desenvolvida na linguagem Java sobre a plataforma Android, seguindo um cronograma, facilitando o controle das tarefas, as quais foram separadas referentes aos módulos que foram desenvolvidos.

Os módulos de comunicação, banco de dados e de cálculos tiveram maior prioridade, por sustentarem o objetivo geral da ferramenta. Referente à parte de elaboração da interface e o módulo de cadastramento tem por sua vez dar suporte as principais funções do software.

Devido a essa diferenciação de prioridades, o desenvolvimento iniciou-se pelos módulos com maior importância, visando à rápida elaboração de uma versão teste, para a execução da validação de uso. A ferramenta foi desenvolvida utilizando a plataforma de desenvolvimento Eclipse versão 4.2.1, também se utilizou o *plug-in* e a SDK do Android que disponibiliza as ferramentas necessárias para o desenvolvimento e compilação da aplicação.

1.4 Organização do Trabalho

Este trabalho é organizado visando à melhor estrutura para o fácil entendimento do leitor.

Na seção 2 são apresentadas referências teóricas em relação as repúblicas de estudantes e a plataforma Android, também abrangendo a ferramenta Eclipse que é a principal ferramenta para desenvolvimento na plataforma. Em seguida o tópico 2.1 refere-se ao surgimento das repúblicas no Brasil e a origem do termo. Logo após no tópico 2.2 é apresentada a bibliografia de como surgiu a plataforma de desenvolvimento Android. No tópico 2.2.1 é explicado a SDK (Software Development Kit do Android), necessária para o desenvolvimento para a plataforma. O tópico 2.3 apresenta a arquitetura do Android, mostrando através de uma imagem suas respectivas partes de funcionamento. O próximo tópico 2.3.1 apresenta o *Kernel* do Android, que utiliza da base Linux. Em seguida o próximo tópico 2.3.2 descreve a funcionalidade das bibliotecas da arquitetura e apresenta alguma delas. Logo após na seção 2.3.2.1 é referenciada a biblioteca SQLite, que tem a função te controlar o banco de dados nativo do Android. Em seguida na seção 2.3.3 é apresentada a parte de compilação *runtime* da arquitetura, e o método Dalvik. No tópico 2.3.4 é explicado o *framework* de aplicação e suas funcionalidades.No tópico 2.4 é apresentado o activity. Após, no tópico 2..4.1 é apresentada uma estrutura importante do activity, o intent. No tópico 2.5 é apresentada a ferramenta de desenvolvimento Eclipse. No tópico 3 inicia-se o desenvolvimento do trabalho. A seção 3.1 é referente a análise da ferramentas correlatas. No tópico 3.2 inicia-se o desenvolvimento do protótipo. No tópico 4 é apresenta a validação de uso do protótipo. Enfim no tópico 5 são dadas as considerações finais do trabalho.

2 FUNDAMENTAÇÃO TEÓRICA

Nesta seção serão apresentados os conceitos sobre o surgimento das repúblicas de estudantes e como são atualmente, visando identificar um perfil para o auxílio na elaboração da ferramenta. Também será explorado o embasamento teórico sobre desenvolvimento para a plataforma Android necessário para a criação da mesma.

2.1 Repúblicas de Estudantes

O termo “república” tem sua origem do latim res publica, que significa “coisa pública”, dando-se a entender de um bem público, de algo coletivo. Denominando as moradias estudantis há diversas suposições considerando que os moradores normalmente são desconhecidos com educação e culturas diferentes (SARDI, 2000, e REPOLÊS, 2007).

No Brasil há certa dificuldade da afirmação de quando surgiram as repúblicas de estudantes, levando em consideração que não havia registros da locação de imóveis por eles, que ficavam breves períodos de tempo nas mesmas.

Segundo Sardi (2000), somente na data de 1897 se encontra registros oficiais de repúblicas estudantis em Minas Gerais, com a transferência de sua capital para Belo Horizonte. Já segundo Naspolini (2008), surgiram diversas moradias estudantis em 1827 devido à criação da faculdade de direito de São Paulo.

Uma república de estudantes pode ser considerada um lugar de extrapolação do prazer (SARDI, 2000). Devido a grande liberdade que os moradores dispõem e também ao ambiente universitário repleto de festas, liberdade sexual e uso de drogas. Isto ocorre principalmente nos dias atuais onde estes prazeres parecem ser mais atraentes.

Um fator pouco citado e muito importante neste assunto é como os moradores administram a moradia, considerando que a diminuição e divisão igualitária das dividas seja o principal objetivo de uma republica de estudantes. Também deve-se considerar que muitos dos estudantes que ingressam em repúblicas não têm experiência no controle e pagamento de dívidas de uma moradia. Assim uma má administração das

despesas pode ser um fator extremamente prejudicial tanto para a situação financeira do estudante quanto para o convívio entre os moradores.

2.2 Android

O Android foi inicialmente desenvolvido no Vale do Silício por uma *startup* chamada Android inc., que foi adquirida pela Google no ano de 2005. Após o amadurecimento do projeto foi levado ao público no ano de 2007. O lançamento da primeira plataforma open source para dispositivos móveis fez tanto sucesso que gerou a criação de um grupo, juntando as maiores empresas no mercado de telefonia como: Sansung, Sony Ericsson, LG, Motorola e outras. Este grupo se denomina Open Handset Alliance (OHA) (LECHETA, 2009).

A plataforma Android é baseada no sistema operacional Linux, geralmente usada em celulares Smartphones e Tablets, disponibiliza muitas ferramentas para o desenvolvimento de aplicativos, também contêm variada gama de bibliotecas e interfaces gráficas (LECHETA, 2009).

Segundo mostra estudo realizado pelo instituto Nielsen, foi constatado que dispositivos de plataforma Android estão sendo preferidos pelos consumidores a o ios no mercado americano, os dispositivos baseados em Android ultrapassam com 31% da preferência os baseados no iOS com 30%. Dado que o investimento no desenvolvimento para a plataforma Android pode ser muito promissor.

2.2.1 SDK Android

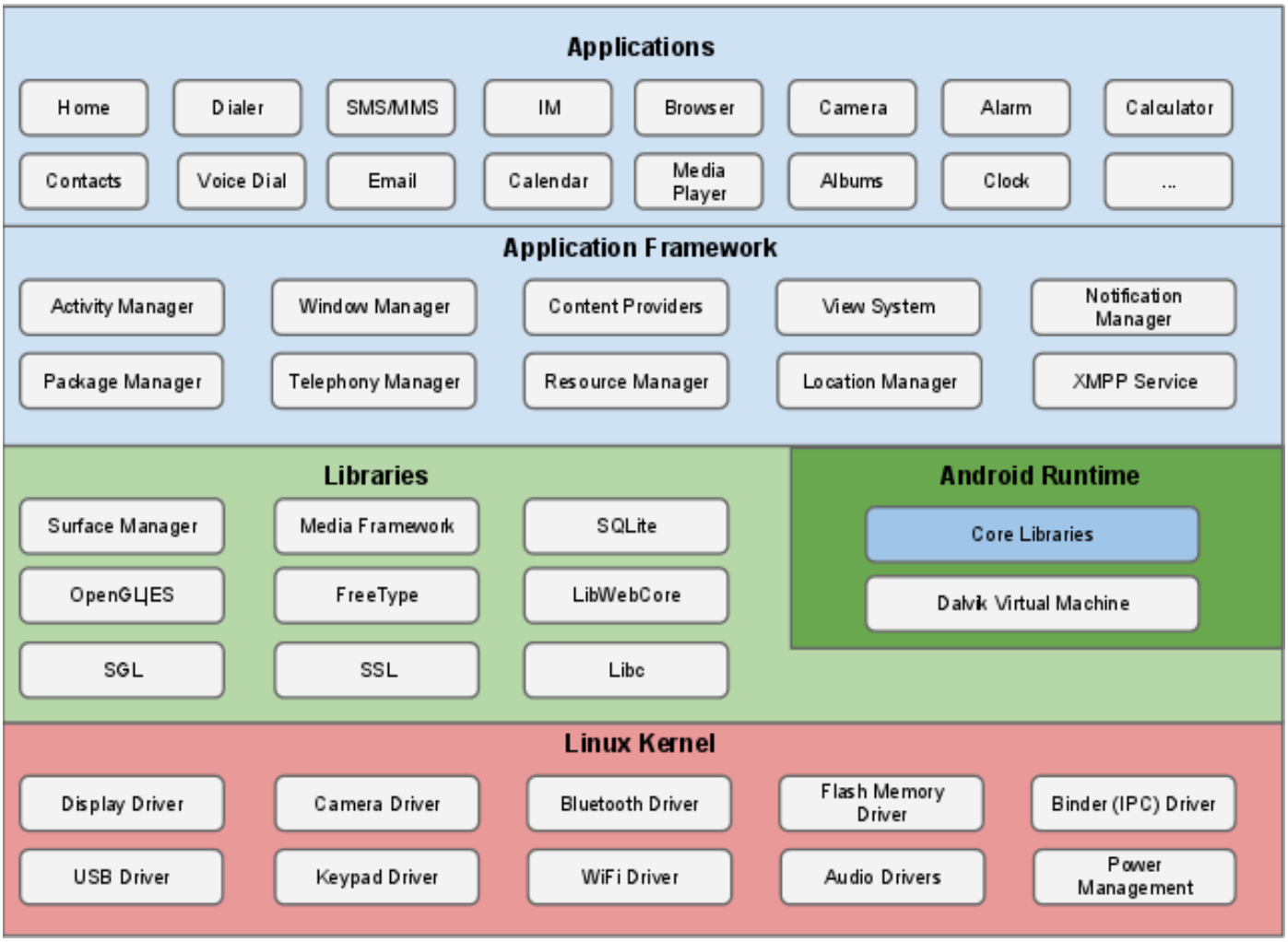
O SDK (Software Development Kit) é um kit disponibilizado pela Google inc. gratuitamente, é extremamente necessária para o desenvolvimento de aplicações para Android, incluindo as bibliotecas de APIs (Application Programming Interface), ferramentas como interfaces gráficas para desenvolvimento e um emulador para testes que simula um aparelho com a plataforma Android perfeitamente (GOOGLE inc., 2013). Junto ao SDK existe um *plug-in* chama ADT (Android Development Tool) que realiza a execução do emulador através da plataforma de desenvolvimento Eclipse (GOOGLE inc., 2013).

Atualmente todas as ferramentas como Eclipse, SDK, ADT e muitas outras estão disponíveis para download no site de desenvolvedores para Android (http://developer.android.com/index.html).

2.3 Arquitetura Android

Para o melhor desenvolvimento da aplicação é necessário o conhecimento sobre a arquitetura da plataforma em que se esta trabalhando, a arquitetura Android consiste em *kernel*, bibliotecas, *runtime*, *framework*, e aplicativos, conforme mostrado na figura 1.

Figura 1 - Arquitetura Android



Fonte: (GOOGLE inc., 2013).

2.3.1 Kernel Linux

Um sistema operacional deve gerenciar os dispositivos de entrada e saída, o processador, além de fornecer uma interface mais simples com o hardware para os usuários (TANEMBAU, 2003).

Com a crescente evolução e melhora dos hardwares de dispositivos móveis a necessidade do desenvolvimento de sistemas operacionais que consigam gerenciar estes dispositivos.

Assim o desenvolvimento de plataforma Android, baseando-se em um *kernel* Linux é uma excelente opção que alem de ser Open Source, segundo Brady (2013), Linux possui um ótimo gerenciamento de processos e memória.

A versão do *kernel* utilizado pelo Android é a Linux 2.6, sendo responsável pela segurança, processos, rede, gerenciamento de memória e *drivers*. O componente *driver* é muito importante pelo fato de garantir ao desenvolvedor uma grande abstração entre o hardware e o software, facilitando o acesso e o gerenciamento de dispositivos específicos do celular (GOOGLE inc., 2013).

2.3.2 Bibliotecas

As bibliotecas existem para o auxílio do desenvolvedor evitando a reescrita de código e facilitando as rotinas que dispõem de maior complexidade. Elas possuem códigos e dados que auxiliam na elaboração do projeto e podem deixá-lo mais rápido (GOOGLE inc. 2013).

O compilador coloca as bibliotecas compartilhadas diretamente em sua aplicação, não necessitando que desenvolvedor tenha que indicá-las manualmente e selecionando em qual aplicação deve rodar (GOOGLE inc., 2013).

O Android possui uma grande quantidade de bibliotecas para desenvolvimento como*: System C Library, Media Libraries, Surface Manager, Lib Webcore, SGL (*Scalated Graphical Library)*, 3D libraries, Freetype e SQLite.* Estas bibliotecas permitem a manipulação de banco de dados, imagens, sons, etc. (GOOGLE inc., 2013).

2.3.2.1 SQLite

SQLite é um SGBD (Sistema Gerenciador de Banco de Dados) embutido no Android, foi desenvolvido para ocupar o menor espaço possível sem perder o desempenho, normalmente indicado para dispositivos que possuem pouco armazenamento e processo, que é o caso de dispositivos celulares. Ainda sim possui uma grande gama de recursos, além de se criar tabelas convencionais como executar comandos SQL (Structured Querry Language), construir *trigers*, *views*, entre outras funcionalidades de um banco de dados. (GOOGLE inc., 2013).

No Android pode-se criar o banco de dados de várias formas, no entanto, a melhor recomendada, é utilizando a biblioteca do Android para SQLite, assim, a própria aplicação criará o banco de dados automaticamente (LECHETA, 2009).

2.3.3 Android RunTime

As aplicações desenvolvidas para Android são sempre interpretadas através de *byte-code,* como as aplicações Java, com a diferença de que Java gera códigos(. class) e os Android (.dex), estes códigos são interpretados pela DVM (*Dalvik Virtual Machine*) (PROJECT, 2013).

A DVM é um aperfeiçoamento da JVM (*Java Virtual Machine*), para melhor suprir as necessidades do Android (BORNSTEIN, 2013).

Caso a plataforma Android necessite ler um extensão (.class), a SDK automaticamente a converterá pra a extensão (.dex) através da DVM, assim, sendo possível a leitura (BORNSTEIN, 2013).

2.3.4 Application Framework

A *Application Framework* tem a funcionalidade de intermediar as funcionalidades da aplicação com as ferramentas nativas do Android, podendo adequá-los as necessidades do aplicativo a se desenvolver (GOOGLE inc., 2013).

Assim uma aplicação pode usar a lista de contatos, câmera, galeria de fotos, etc. A *Application Framework* fornece diversas ferramentas para o auxílio do programador, entre eles se destacam o provedor de conteúdo, gerenciador de

notificações e gerenciador de atividades. Assim proporcionando uma experiência mais atrativa e funcional para o usuário (GOOGLE inc., 2013).

2.4 Activity

O componente *activity* está entre os principais de uma aplicação para Android, pois é a tela de atividades que o usuário vai interagir. Para cada atividade executada haverá uma *acticity* com sua própria interface (GOOGLE inc., 2013).

Uma aplicação normalmente se compõe de várias *activitys* que se interligam, gerando um conjunto de atividades. Um *activity* é apresentado ao usuário ao início do aplicativo denominado como "*activity* principal", por sua vez pode-se chamar outras atividades e assim sucessivamente (GOOGLE inc., 2013).

Quando uma nova atividade é iniciada, a que estava sendo executada é interrompido,entra-se em uma pilha para preservar seus dados, caso o usuário volte a usá-la (GOOGLE inc., 2013).

Para o desenvolvedor é muito importante o conhecimento sobre o *activity* , pois nele se encontrará grande parte de seu trabalho e onde se alojará as atividades de sua aplicação.

2.4.1 Intent

Intent é um objeto que permite o envio de mensagens de comandos para outras aplicações do aparelho, assim, uma estrutura passiva leva uma descrição abstrata de uma tarefa a ser realizada (GOOGLE inc., 2013).

As informações passadas pelo objeto *intent* são de grande importância para o componente que a recebe, lhe mostra qual ação a ser tomada e os dados necessários para isso. Também existem informações de interesse para o sistema Android lhe indica qual o tipo de componente a receber o *intent*, e instruções de como iniciar uma atividade (GOOGLE inc., 2013).

Assim, um objeto de extrema utilidade, pode facilitar a manipulação de componentes como câmera do aparelho, mapas, navegação GPS, cliente de e-mails, e outras funcionalidades do aparelho.

2.5 Eclipse

O Eclipse é uma IDE (Integrated Development Enviroment) elaborada em Java patrocinado pela IBM (International Business Machines), que suporta a criação de aplicações em diversas linguagens dentre elas Java e C++ (ECLIPSE, 2002).

Além de ser uma IDE excelente, o Eclipse pode ser compatível com outras ferramentas exerce outras funções, como a ferramenta de controle de versão CVS(*Competitor Version System*), também pode executar internamente servidores para fazer a depuração de objetos distribuídos (ECLIPSE, 2002).

O Google inc. escolheu o Eclipse para ser a principal plataforma de desenvolvimento para Android, melhor auxilia no desenvolvimento, testes e na compilação dos projetos (Google inc., 2013).

2.6 Softwares Colaborativos

O trabalho em grupo permite pessoas trabalharem de uma forma mais eficiente, não sendo diferente para a informática. Atualmente a utilização de softwares colaborativos para melhores desempenhos esta cada vez maior. Estes softwares também podem ser chamados de *groupware*, está área é considerada pelos especialistas de extrema fragilidade devido à grande diferença entre os softwares colaborativos, sabendo que são adaptados para uma tarefa específica ou um determinado perfil de usuário (Ellis, 1991).

Uma definição para os *groupwares* é que são softwares para a comunicação de grupos de pessoas, procura a organização da informação, auxilia na coordenação e cooperação (Primo; Brambilla, 2013).

Para o melhor desenvolvimento de groupwares é existente modelos de desenvolvimento específicos como o modelo 3C, que visa às principais funcionalidades que um software colaborativo deve ter como comunicação, coordenação e cooperação (laurillau; Nigay, 2002). A comunicação é realizada na troca de mensagens, coordenação é o gerenciamento de pessoas, atividades e recursos disponíveis, cooperação a execução das tarefas em um espaço compartilhado (Ellis, 1991).

2.6.1 Modelo 3C

O modelo 3C se equivale ao modelo Clover, modelo onde são definidas três classes de funcionalidade: comunicação, coordenação e produção, no entanto produção se equivale à cooperação no modelo 3C (laurillau; Nigay, 2002). Estes modelos utilizam destas três dimensões para melhor atender aos requisitos dos *groupwares* que os modelos tradicionais não atendem adequadamente pelo fato de visarem à coordenação do grupo de desenvolvimento, assim o processo de colaboração e cooperação invertidos em relação com o modelo 3C (Ellis, 1991).

Busca-se sempre o objetivo de melhorar e facilitar o desenvolvimento de softwares colaborativos foi criado o RUP-3C-Groupware, que consiste em uma extensão do RUP (Rational Unified Process), no entanto foram incorporadas as boas praticas para o desenvolvimento de *groupwares*. Utilizado nas diferentes etapas de desenvolvimento de um software colaborativo, como: a análise dos domínios de classificação do groupware e seus elementos, e na construção dos componentes de montagem do groupware (Pimentel, 2006).

2.7 Software Colaborativo e Tecnologia Móvel

Sistemas colaborativos podem ser definidos como aplicações de suporte a grupos de pessoas que se comunicam e coordenam, para atingir um objetivo em comum. A ideia de apoia a colaboração através da computação remonta o ano de 1968, quando o pesquisador americano Douglas Engelbart demonstrou uma aplicação para compartilhamento de telas evolvendo duas pessoas, se comunicando através de uma rede com interfaces de áudio e vídeo (BARDINI, 2000). Os primeiros trabalhos na área de sistemas colaborativos foram para o suporte à realização de tarefas em conjunto por usuários em localidades diferentes, geralmente tinham por objetivos criar áreas de trabalho virtuais que poderiam ser utilizados para a colaboração (FARSHCHIAN, 2010).

Soluções eficazes para aplicações colaborativas em redes cabeadas são bem conhecidas, o mesmo não se aplica a colaboração de usuários entre dispositivos móveis. Os avanços na comunicação sem-fio e computação portátil trazem novos

desafios e necessidades para colaboração nos cenários móveis (NEYEM, 2008). Com conectividade intermitente, juntamente com fatores como contexto dos usuários dos dispositivos e do ambiente, passaram a ser fundamentais para o eficaz funcionamento dos sistemas colaborativos.

Um dos fatores positivos do uso de dispositivos móveis como plataforma para a colaboração é, maior alcance da tecnologia móvel em termos da base de usuários desses equipamentos nos dias atuais, que já ultrapassou o número de usuários de desktops. Outro fator benéfico é que em geral dispositivos móveis são artefatos pessoais, estando diretamente associados a seu usuário. Desta forma, podendo ser registrados nos dispositivos, informações que indiquem suas preferências pessoais e estabeleçam seu perfil. Finalmente, dispositivos móveis podem ser considerados sempre disponíveis, oferecendo a possibilidade da participação contínua em aplicações colaborativas (FARSHCHIAN, 2010).

2.8 Sincronização

O processo de sincronização permite trocar dados entre um *Pocket* PC e um PC desktop para atualização ou realização de cópias de segurança. Existem várias ferramentas que permitem a sincronização de dados com base em protocolos normalizados, definidos pela OMA (Open Mobile Alliance), que permitem assegurar que os dados existentes em ambos os lados são idênticos. Exemplos de tais ferramentas incluem o Hotsync para dispositivos Palm OS e o ActiveSync da Microsoft para dispositivos Windows Mobile, que sincronizam dados de agenda, contatos, notas textuais, mensagens de e-mail, ficheiros genéricos e documentos (TIFFANY, 2003).

No entanto, existem cenários de utilização em que se pretende outro tipo de sincronização dos dados, tal como no caso em que se utilizam bases de dados. Na realidade, com este tipo de aplicações é necessário um processo de sincronização mais sofisticado que permita assegurar a validade dos dados em ambos os lados da sincronização (FOX, 2003).

Visando solucionar o problema de sincronização foi criado um modelo semelhante ao modelo *peer-to-peer*, no modelo tomado como base não existe diferença

entre o cliente e o servidor, ambos podem reproduzir os dois papeis. Esse modelo na tecnologia móvel faz com que estes dispositivos tornem-se parceiros idênticos numa computação distribuída (MATEUS, 1998).

O modelo criado tem como base de sua comunicação o envio de e-mails contendo em anexo um arquivo de texto com os dados necessários para a atualização das bases locais dos usuários.

3 DESENVOLVIMENTO DO APLICATIVO

Neste tópico é apresentado o desenvolvimento do software quanto à análise feita sobre as ferramentas correlatas selecionadas.

3.1 Ferramentas Correlatas

Nesta seção será apresentada a análise das ferramentas que possam de alguma forma atender o problema proposto no trabalho, esta análise serviu de suporte para detecção de fatores cruciais para o desenvolvimento de uma ferramenta mais eficiente e que atenda melhor seu propósito.

Não foram encontradas aplicativos para Android ou softwares que tivessem o âmbito de gerenciar repúblicas estudantis em específico, mas que podem atender parcialmente os problemas encontrados pelo trabalho neste tipo de gestão.

Os softwares estudados foram: editores de planilhas de forma geral, Toshl Finanças, Minhas Economias, Minhas Contas, Personal Finances e Home Budget. Com exceção dos editores de planilhas que podem ser encontrados para diversas plataformas. Todos os softwares citados pertencem à plataforma Android, pelo fato de as demais ferramentas de softwares encontrados na pesquisa, que pertenciam a outras plataformas móveis, terem extrema semelhança com as analisadas, quando não a mesma pelo fato de o aplicativo estar disponível em diversas plataformas.

3.1.1 Apresentação das Ferramentas Correlatas

Nesta seção é apresentada a descrição das ferramentas selecionadas para a análise juntamente com suas principais características.

3.1.1.1 Toshl Finanças

Um aplicativo para a plataforma Android de administração financeiras pessoal. Vencedor do premio Europa 2013 para melhor comércio, finanças ou pagamentos Startup (Google Inc., 2013).

Principais Características:

- Repetindo as despesas e contas;
- Rendimentos de repetição;
- Torná-lo fácil de poupar dinheiro e as despesas de pista;
- Comparar a taxa de seus gastos com a época do mês;
- Orçamentos, diária, semanal, quinzenal, mensal, anual;
- Movimentar os fundos restantes para o próximo orçamento;
- Aprender com a sua história orçamento;
- Entrar despesas em qualquer moeda;
- Conversor de moeda por mais de 160 moedas;
- Definir a taxa de câmbio personalizado;
- Planejar suas despesas e contas com antecedência;
- Criar despesas repetir e obter uma sensação de contas do próximo mês;
- Começar o mês financeiro em qualquer dia;
- Navegar por qualquer período de tempo (Google Inc., 2013).

3.1.1.2 Minhas Economias

O Minhas Economias permite o controle financeiro, criação de orçamentos e planejamentos. Assim facilitando saber sobre os gastos, fazer sobrar dinheiro no final do mês e realizar sonhos. Gerencie as contas de diversos bancos, cartões de crédito, investimentos, financiamentos, aposentadoria. O Minhas Economias é seguro, anônimo e gratuito (Google Inc., 2013).

Principais Características:

- Verificar o saldo de todas as suas contas;
- Cadastrar transações de despesas, receitas e transferências;

- Programar repetição de transações;
- Lançar parcelamentos;
- Cadastrar lembretes para não atrasar contas;
- Cadastrar notas;
- Categorizar transações;
- Consultar e alterar as transações já cadastradas (Google Inc., 2013).

3.1.1.3 Minhas Contas

Aplicativo criado para facilitar o controle financeiro. Fácil de usar, permite compartilhar o conteúdo por correio eletrônico ou mensagem de texto, e visualizar todas as receitas, despesas e aplicações rapidamente. Organizar as contas por tipo e classe, e conheça os gastos. Saiba mensalmente o que acontece com o orçamento por meio do resumo e dos gráficos. Entenda como está o saldo e acompanhe a evolução das contas, prestações e aplicações (Google inc., 2013).

Principais Características:

- Resumo mensal do seu orçamento;
- Gráficos de comparação de contas;
- Backup completo no Cartão SD;
- Edição a qualquer momento de todas as informações;
- Organização da ordem das contas no resumo mensal;
- Alternar entre os meses por gestos;
- Pesquisa de contas criadas;
- Envio por e-mail e mensagem de texto;
- Exportar o resumo mensal para planilha em Excel;
- Bloquear com senha o acesso ao aplicativo;
- Aplicar juros a prestações e investimentos;
- Parcelar valores com aplicação de juros;
- Ordenar as contas por data;
- Lançamento do pagamento das contas de forma automática;

- Detalhe do saldo atual mostrando somente o que foi pago;
- *Widget* (quando aplicativo instalado na memória interna);
- Controlar as contas diariamente;
- Coloque o vencimento de sua conta no calendário do celular;
- Repita as contas diariamente, semanalmente, mensalmente ou anualmente (Google Inc., 2013).

3.1.1.4 Personal Finances

Personal Finances foi desenvolvido para fornecer ao utilizador através de uma representação gráfica de fácil compreensão as suas finanças pessoais. (Google inc., 2013).

Principais Características:

- Registro de categoria, data, beneficiário, montante e descrição;
- Moeda estrangeira;
- Atribuição automática de sinal negativo para categorias de despesa;
- Totais por despesas, receitas e saldo;
- Reconciliação de movimentos (Google Inc., 2013).

3.1.1.5 Home Budget

Agora disponível em *mobile* [*Android, iPhone / iPad*] e versões desktop [Windows, Mac OS], incluindo sincronização instantânea de dados entre as versões móveis / desktop. *HomeBudget* é um perseguidor de despesa integrada projetada para ajudar a controlar as despesas, receitas, contas em atraso e saldos de contas. Oferece suporte para elaboração de orçamentos e permite a análise das despesas e receitas, incluindo tabelas e gráficos (Google inc. 2013).

Principais Características:

- Criar / Editar / Excluir gastos e despesas recorrentes;
- Procurar entradas de despesas por mês, por categoria / data com a capacidade de drill-down;

- Opcionalmente associar uma conta do beneficiário e com uma conta;
- Criar / Editar / Excluir renda e lucro recorrente;
- Opcionalmente associar uma conta com a renda;
- Criar / Editar / Excluir contas e contas recorrentes;
- Suporte para Beneficiários, e pagamentos de trilha por beneficiário;
- Fazer o pagamento de contas ou uma como despesa ou como uma transferência;
- Vista Calendário e lista de contas;
- Criar / Editar / Excluir categorias de despesas e sub-categorias;
- Defina um orçamento e controlar as despesas na categoria / nível sub-categoria;
- Ícones associados com as categorias e reordenar a ordem de exibição categoria;
- Classificar as despesas em fixos, variáveis e discricionária e calcular o seu rendimento disponível (Google Inc. 2013);

3.1.1.6 Editores de Planilhas

Editores de planilhas são softwares utilizados para a edição de planilhas eletrônicas, capaz de personalizar uma planilha de diversas formas, como também pode ser usada como uma planilha comum, apenas para agrupamento e organização de informação como em uma tabela. Existindo diversos editores no mercado é possível inserir cálculos, gráficos e diversas outras funcionalidades em uma planilha, podendo ser utilizada para diversas tarefas, levando em consideração que editores se diferem em características e funcionalidades (Acessasp, 2013).

Principais Características:

- edição e formatação de planilhas eletrônicas;
- utilização de cálculos e gráficos;
- possibilidade de compartilhar planilhas (Acessasp, 2013).

3.1.2 Análise das Ferramentas Correlatas

As seleções de alguns fatores cruciais para a solução dos problemas propostos no trabalho servirão como requisitos para analisarmos os softwares e aplicativos como:

Comunicação - Meios de compartilhamento de informações entre outras pessoas existentes nos softwares.

Divisão de custo – Possibilidade de dividir os eventos financeiros do software entre outras pessoas.

Armazenamento – Armazenamento das informações do software e visualização posteriormente.

A tabela 1 definirá as características dos softwares em relação aos requisitos gerados.

Tabela 1 - Análise das Ferramentas Correlatas

	COMUNICAÇÃO	DIVISÃO DE CUSTO	ARMAZENAMENTO
Editores de Planilhas	Existente, necessidade de certo nível de conhecimento devido à utilização de ferramentas externas para o compartilhamento das planilhas.	Existente, dependente da configuração da planilha utilizada.	Armazenamento feito salvando as planilhas e suas alterações.
Toshl Finanças	Inexistente.	Inexistente.	Existente.
Minhas Economias	Inexistente.	Inexistente.	Existente.
Minhas Contas	Inexistente.	Inexistente.	Existente.
Personal Finances	Inexistente.	Inexistente.	Existente.
Home Budget	Existente, conectando-se com outras pessoas para compartilhamento de informações.	Inexistente.	Existente.

Fonte: Própria Autoria

3.1.3 Conclusão da Análise das Ferramentas Correlatas

As ferramentas analisadas podem cumprir alguns requisitos gerados a partir do problema proposto no trabalho, no entanto nenhuma pode cumprir de forma satisfatória todos os requisitos, a ferramenta *Home Budget* cumpriu de forma satisfatória o requisito de comunicação, sendo focado para residências familiares deixando ausente a divisão igualitária das dividas de uma casa, assim não cumprindo um dos principais requisitos para a resolução do problema.

Os editores de planilhas podem cumprir todos os requisitos com uma grande dependência de como o utilizador formatará suas planilhas, e utilizando formas externas aos editores para seu compartilhamento, essas atividades podem ser complicadas para muitas pessoas, além de não ter a garantia de um software que foi desenvolvido especificamente para solucionar estes requisitos. Todas as ferramentas são excelentes, o objetivo do trabalho não é desenvolver um software melhor que os analisados, mas sim desenvolver uma solução para os problemas propostos.

3.2 Desenvolvimento do Protótipo

Neste tópico será apresenta o desenvolvimento do protótipo que tem por objetivo auxiliar a administração de uma república de estudantes.

3.2.1 Requisitos

Foram utilizados três métodos para o levantamento dos requisitos, o método de etnografia, que consiste na elaboração de uma análise do ambiente em que o software desenvolvido será utilizado, observando o contexto social e organizacional no qual o software foi utilizado (VILLER, 2000). Método de entrevista e questionário, visando observar as verdadeiras necessidades do usuário final.

Análise do ambiente

As repúblicas estudantis não podem ser divididas em setores como uma empresa, mas para a análise deste ambiente é importante se destacar que serão observados os fatores referentes a gestão financeira pelos moradores. Alguns fatores foram selecionados para a análise como: Métodos utilizados para a gestão do grupo pelos moradores; ferramentas e métodos utilizados para a gestão da casa; frequência que é executada a comunicação das informações.

Gestão do grupo pelos usuários

Não é utilizado nenhum método específico para a gestão dos grupos pelos moradores de repúblicas estudantis, na maioria das vezes é selecionado um dos moradores para a organização e comunicação dos eventos referentes a casa para os outros moradores, não existindo sequência ou ferramenta específica para isto, normalmente utilizando redes sociais e emails. a distribuição é feita somente ao fim do mês ou na existência de um acontecimento incomum.

Métodos utilizados para a gestão da casa.

Os editores de planilhas são as ferramentas mais utilizadas para este tipo de gestão, também podendo ser utilizado métodos manuais para esse gerenciamento. A divisão das dívidas é feita de varias formas, dependendo de como foi projetada a planilha utilizada ou manualmente, mesmo na utilização de planilhas ocorrem falhas pelo fato de muitos acontecimentos não serem inseridos ou comunicados como: pagamento de uma dívida por um morador compra de algum produto, entre outros acontecimentos. Assim necessitando de verificações e reuniões para a averiguação dos eventos.

Frequência de comunicação

A frequência da comunicação é irregular, sendo transmitidas as informações conforme o encontro dos moradores, somente ao final do mês e em algumas ocasiões se utiliza algum meio de comunicação para o compartilhamento das informações da residência ou somente o valor a pagar de cada morador.

A partir da análise destas observações foram selecionados os principais requisitos para o desenvolvimento do software, estes requisitos serão confirmados perante entrevista e questionário aplicado aos moradores de repúblicas estudantis.

Entrevista

Foram entrevistadas sete moradores de repúblicas estudantis, com experiência neste tipo de moradia entre dois a quatro anos, sendo que dois dos sete entrevistados moraram em mais de uma república. Foram selecionados pontos principais necessários para esta entrevista, sendo explicitados em um roteiro semi-estruturado que se encontra no anexo B.

As entrevistas foram de suma importância para o levantamento dos requisitos, verificando as verdadeiras necessidades de uma república de estudantes, foram declarados em um âmbito geral pelos entrevistados que a comunicação das informações da moradia não obtém uma regularidade, assim dificultando os moradores de estarem cientes sobre os acontecimentos da casa. Em relação ao armazenamento das informações em todos os casos é utilizado de planilhas, no entanto apenas um morador é responsabilizado pela sua edição, não tendo uma frequência regular, sendo compartilhada apenas ao fim do mês. A divisão das dívidas na maioria dos casos não se utiliza a própria planilha, devido à falta de conhecimento para a edição dos cálculos necessários, sendo feita de forma manual ou com utilização de calculadora. Ao fim da entrevista foi aplicado um questionário que se encontra no anexo C.

Após analisado a entrevista e os questionários, foram gerados os documentos referentes aos requisitos do software, que estão situados no anexo A. Devido aos requisitos levantados, as principais funcionalidades que o software deve conter são: Transmissão dos dados para verificação de todos; controle dos eventos da república estudantil; divisão das dívidas de forma automática, visualização dos dados pelos usuários.

3.2.2 Ambiente de Desenvolvimento

Para o desenvolvimento de aplicativos para Android é necessário ter um ambiente de desenvolvimento configurado contendo JDK (*Java Development Kit*) e também o SDK (*Software Development Kit)*, instalados em uma IDE, a selecionada para este trabalho foi o Eclipse na versão 4.2.1. IDE que possibilita a instalação de varias ferramentas que auxiliam no desenvolvimento, uma ferramenta essencial para o desenvolvimento para a plataforma Android é o ADT (Android Development Tool), que também é utilizado neste trabalho.

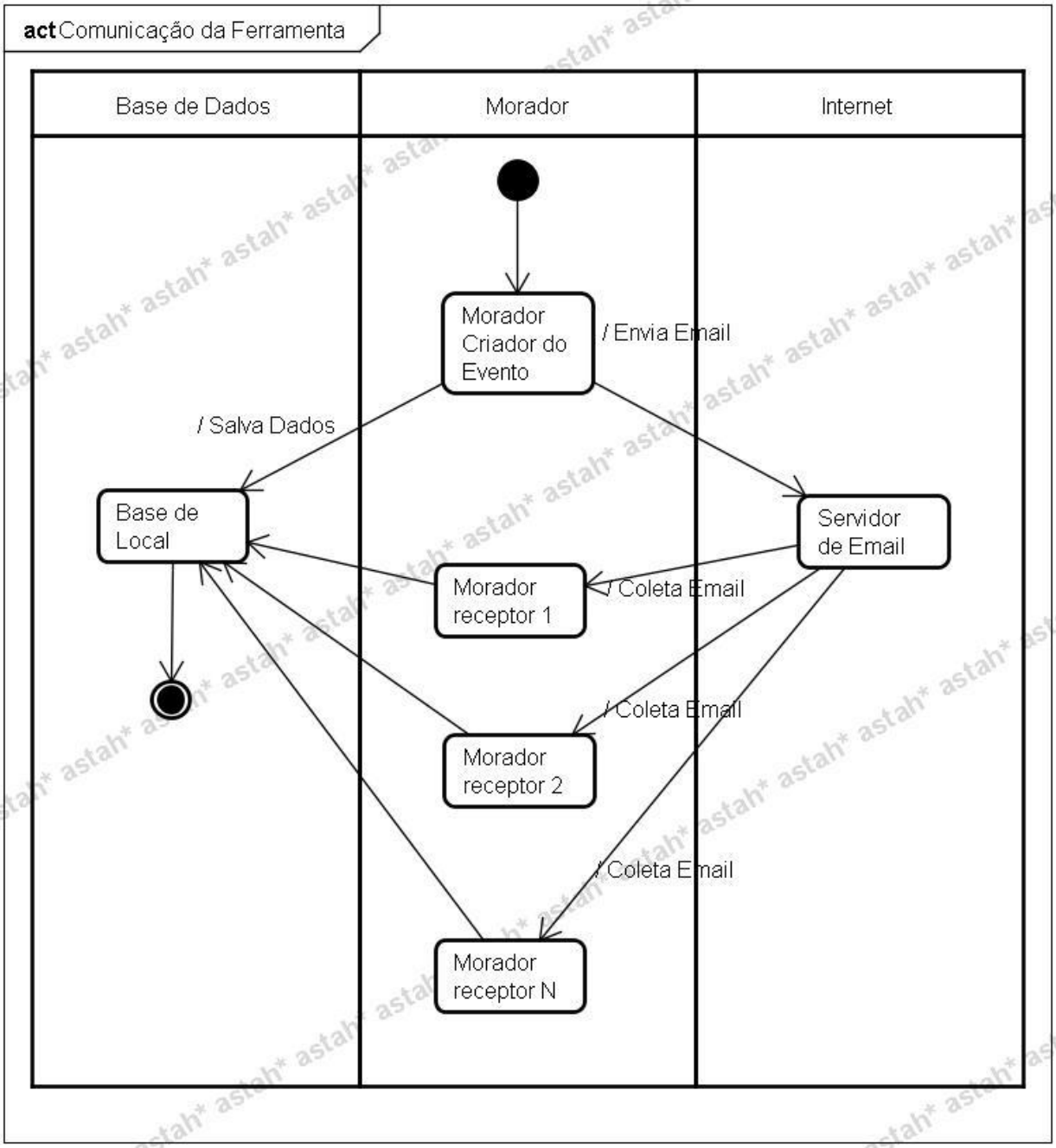
Para poder testar o protótipo desenvolvido foi utilizado a AVD (*Android Virtual Device*), proporciona à emulação de um dispositivo virtual Android, podendo-se escolher o modelo do aparelho a se simular e todas as suas especificações como:

- Um perfil de hardware (possui câmera, utiliza teclado, etc.);
- Qual versão da plataforma Android a executar;
- Emulador de cartão SD;
- Área de armazenamento na máquina de desenvolvimento.

3.2.3 Funcionamento da Comunicação do Protótipo

Neste Tópico é apresentado o funcionamento da comunicação por emails do protótipo, como mostrado na figura 2.

Figura 2 - Comunicação



Fonte: Própria Autoria.

A comunicação acontece através da troca de emails feita pelo software, neste email contem um arquivo de texto em anexo onde estão escritas as instruções a serem executadas para a atualização da base de dados de cada usuário. É utilizado o assunto do email para identificação do mesmo, onde são utilizadas as informações do

evento como chave para diferenciação entre eles. Esta chave é composta da palavra "RepAdminNovoEvento" as informações utilizadas são: o email do criador do evento, a data da criação deste evento e a chave primária referente a este evento.

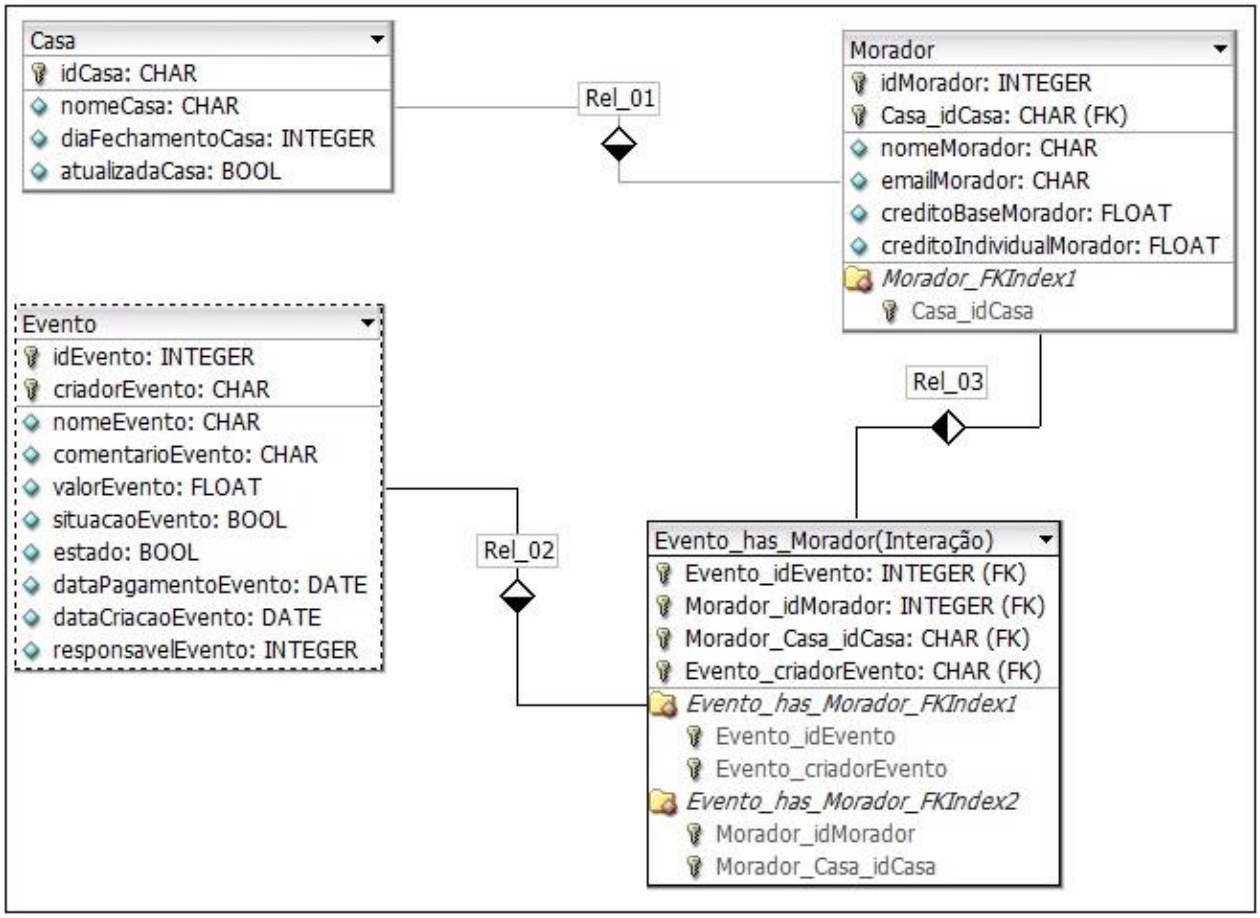
Ao se manipular um evento, criando, alterando ou excluindo, automaticamente é atualizada a base de dados do usuário manipulador. Em seguida é escrito o arquivo de texto que conterá as instruções para atualização da base de dados para os outros usuários, é gerado um email e anexado a ele o arquivo de texto que é enviado para os outros usuários, ao iniciarem o software utiliza o assunto do ultima email recebido referente ao software para verificar a existência de novos emails. Esta verificação se valida quando encontrados emails com a chave de verificação no assunto do email referente ao padrão selecionado para o software, a data deve ser igual ou posterior a do ultimo evento, o horário de envio deve ser superior e a chave primaria do evento maior ou igual.

Se encontrado um email não utilizado antes para atualizar a base de dados é recolhido o arquivo de texto que estará em anexo, executando as instruções contidas no arquivo. As atividades de verificação do email e utilização do anexo são executadas até não se encontrar novos emails.

3.2.4 Banco de Dados

Para o desenvolvimento do banco de dados foi utilizado o SQLite, motor de banco de dados nativo do Android. O banco de dados é composto por quatro tabelas, Casa, Morador, Evento e a tabela referente ao relacionamento entre moradores e eventos. O desenvolvimento do banco de dados seguiu o modelo entidade relacionamento, visado a melhor estrutura dos dados para facilitar atualizações no software e melhor tratamento das informações. A figura 3 exibe a estrutura geral do bando de dados.

Figura 3 - Estrutura Geral Banco de Dados



Fonte: Própria Autoria.

A tabela Casa contém as seguintes colunas:

- idCasa - Chave primária da tabela;
- nomeCasa - Nome da casa;
- diaFechamentoCasa - Dia o qual os moradores desejam que seja o fechamento do mês.
- atualizadaCasa - Verificador para se todos o moradores estão cadastrados, caso não estejam, o usuário fica impedido de criar novos eventos até atualização.

A tabela Morador contém as seguintes colunas:

- idMoradore - Chave primária da tabela;
- nomeMorador - Nome do morador;
- emailMorador - Email do morador;

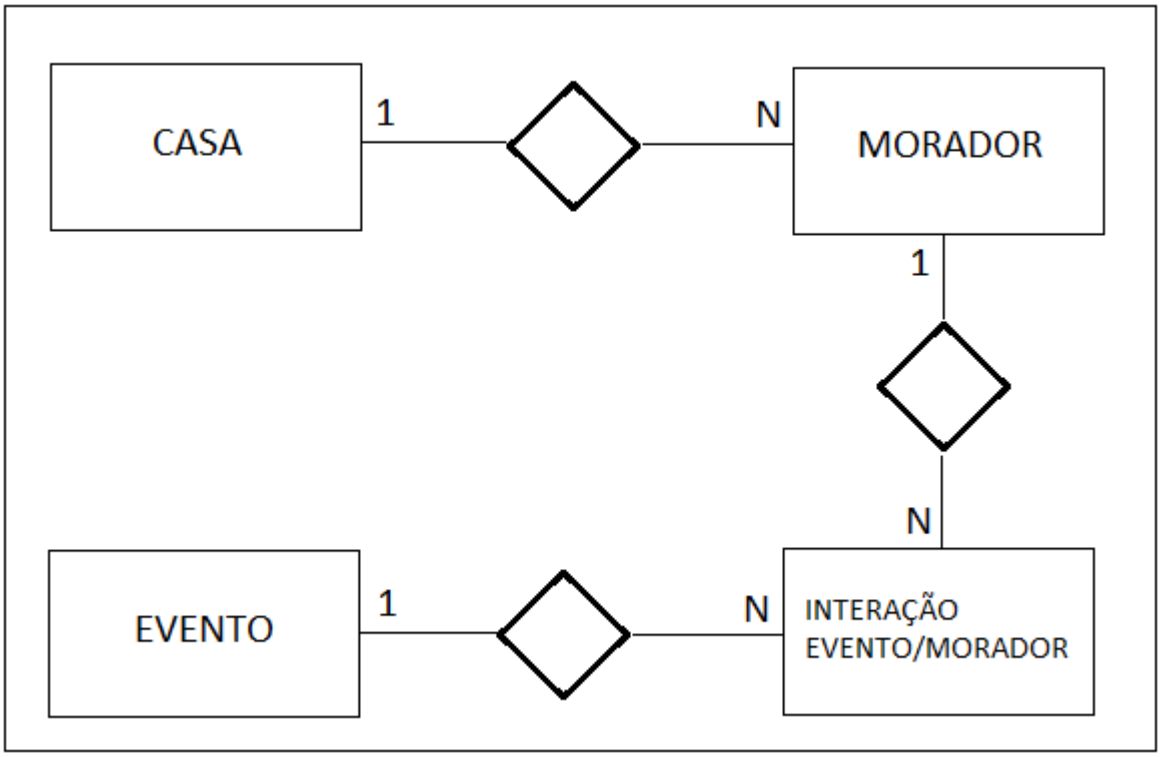
- creditoBaseMorador - Crédito caso o morador tenha pago algum evento da república estudantil;
- creditoIndividualMorador - Credito caso o morador tenho pago algum evento em que não são todos os moradores participantes;

A tabela Evento Contém as seguintes colunas:

- idEvento - Chave primário do evento;
- nomeEvento - Nome do evento;
- comentarioEvento - Comentário sobre o evento caso necessário;
- valorEvento - Valor a ser pago do evento;
- estadoEvento - Campo dedicado a indicar se o evento está válido, foi alterado ou excluído;
- situacaoEvento - Indica se o evento está pago ou não;
- dataPagamentoEvento – Data de pagamento do evento;
- dataCriacaoEvento – Data qual o evento foi craido;
- responsavelEvento – Indica qual morador esta responsável pelo evento caso algum morador ter sido responsabilizado;
- criadorEvento – indica qual morador criou o evento;

A figura 4 representa o diagrama de entidade relacionamento do banco de dados, onde a tabela "Casa" pode conter diversos moradores, no entanto a tabela "Moradores" pode conter apenas uma casa. É definido que cada morador pode participar de diversos eventos, porem os eventos podem conter dois ou mais moradores, assim necessitando de uma tabela intermediária para esta relação.

Figura 4 - Diagrama Entidade Relacionamento

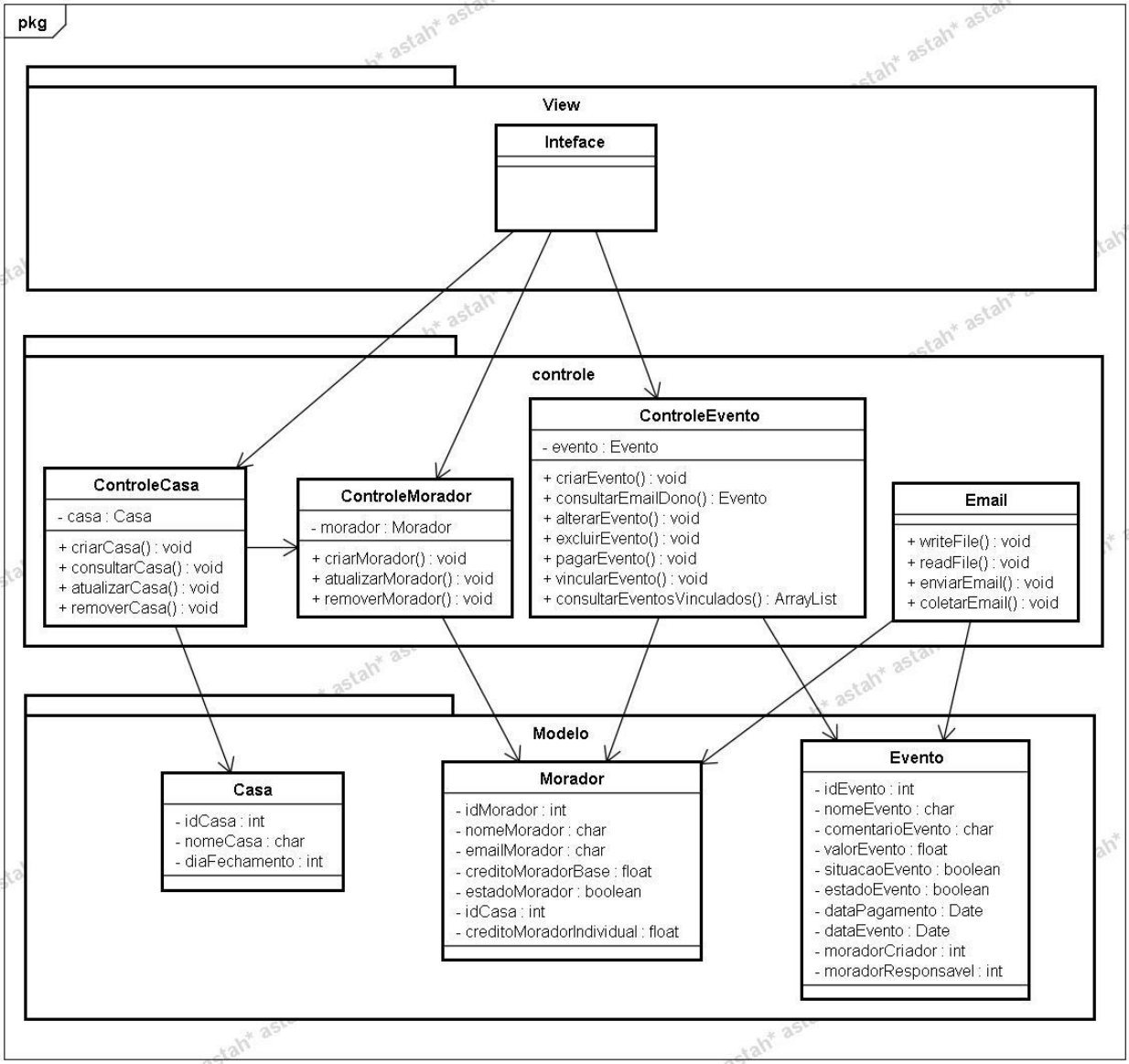


Fonte: Própria Autoria

3.2.5 Diagrama de Classes e Arquitetura

Na figura 5 e 6 é apresenta a estrutura geral do software, o diagrama de classes mostra as classes necessárias para o funcionamento do protótipo e suas interações e o diagrama de arquitetura mostra a arquitetura geral do software.

Figura 5 - Diagrama de Classes



Fonte: Própria Autoria.

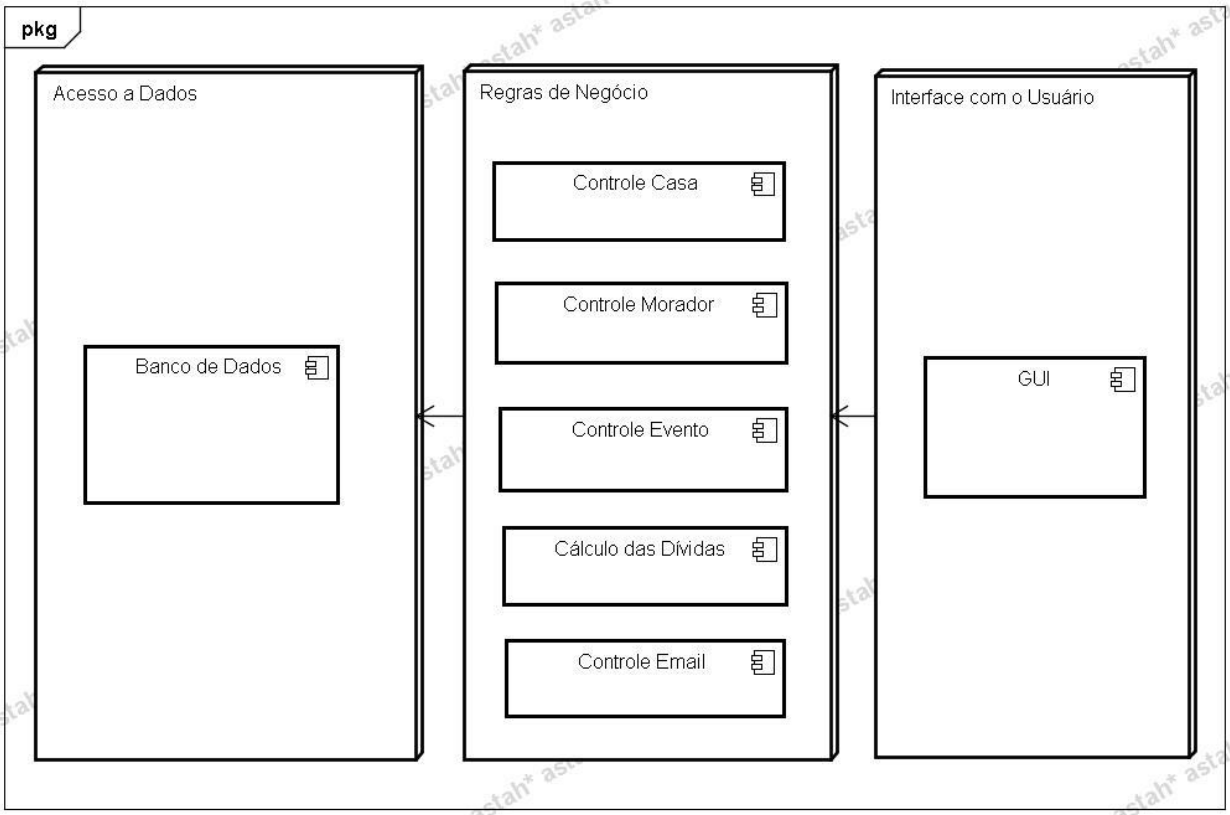


Figura 6 - Diagrama de Arquitetura

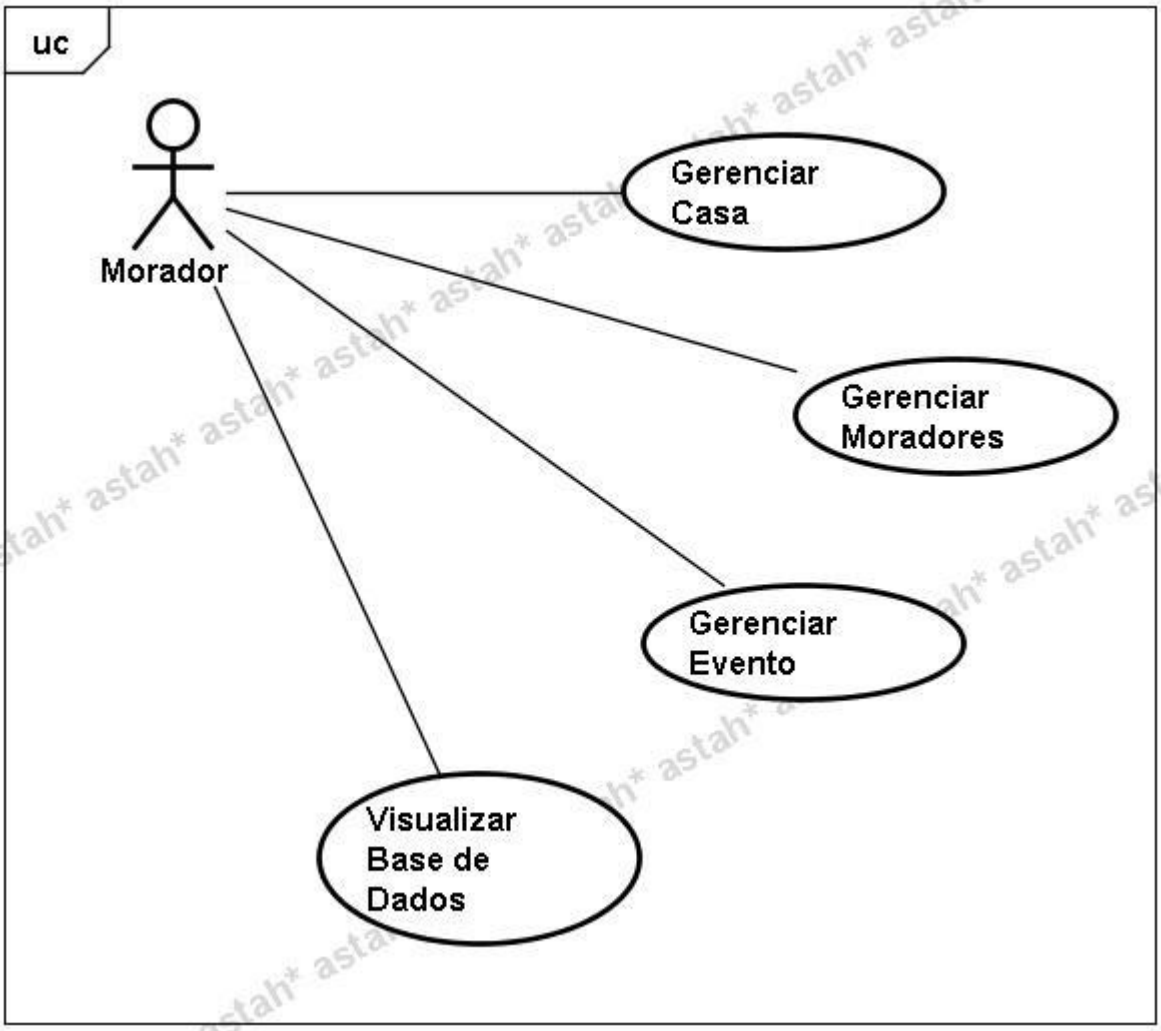
Fonte: Própria Autoria.

3.2.6 Casos de Uso

Nesta seção serão apresentados os casos de uso das principais funcionalidades do protótipo, juntamente com suas respectivas explicações.

3.2.6.2 Caso de Uso Geral

Figura 7 - Caso de Uso Geral

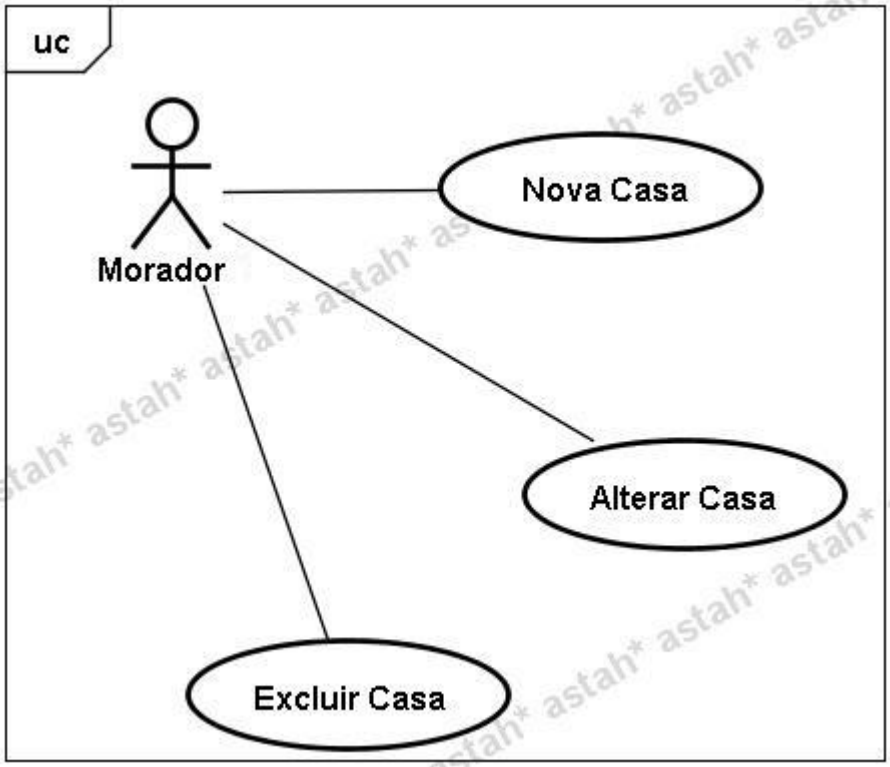


Fonte: Própria Autoria.

Na figura 7 é apresentado o caso de uso referente às atividades genéricas que o usuário poderá executar no aplicativo como, cadastro da casa, cadastro dos moradores, postarem eventos, visualizar base de dados. Nos próximos diagramas serão apresentadas as atividades de forma mais específica.

3.2.6.3 Caso de Uso das Atividades Referentes a Casa

Figura 8 - Caso de usa Casa



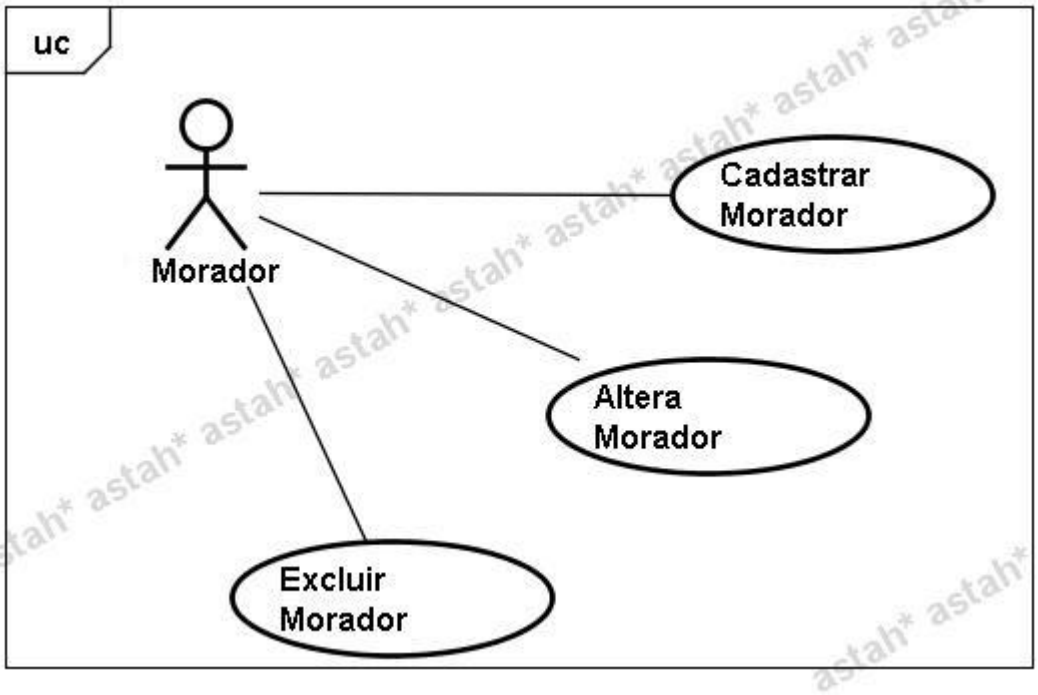
Fonte: Própria Autoria.

Na figura 8 é apresenta o uso referente a casa como:

- Nova Casa - Opção para o cadastro de uma nova casa, onde se deve ter extrema atenção nas informações cadastradas, devido a algumas delas terem relação direta com a contabilidade da casa. As informações serão nome da casa, valor do aluguel, dia de fechamento do mês.
- Alterar Casa - Esta opção permitirá o usuário de modificar informações da casa.
- Excluir Casa – opção que exclui todas as informações da casa.

3.2.6.4 Caso de Uso Referente aos Moradores

Figura 9 - Caso de Uso Moradores



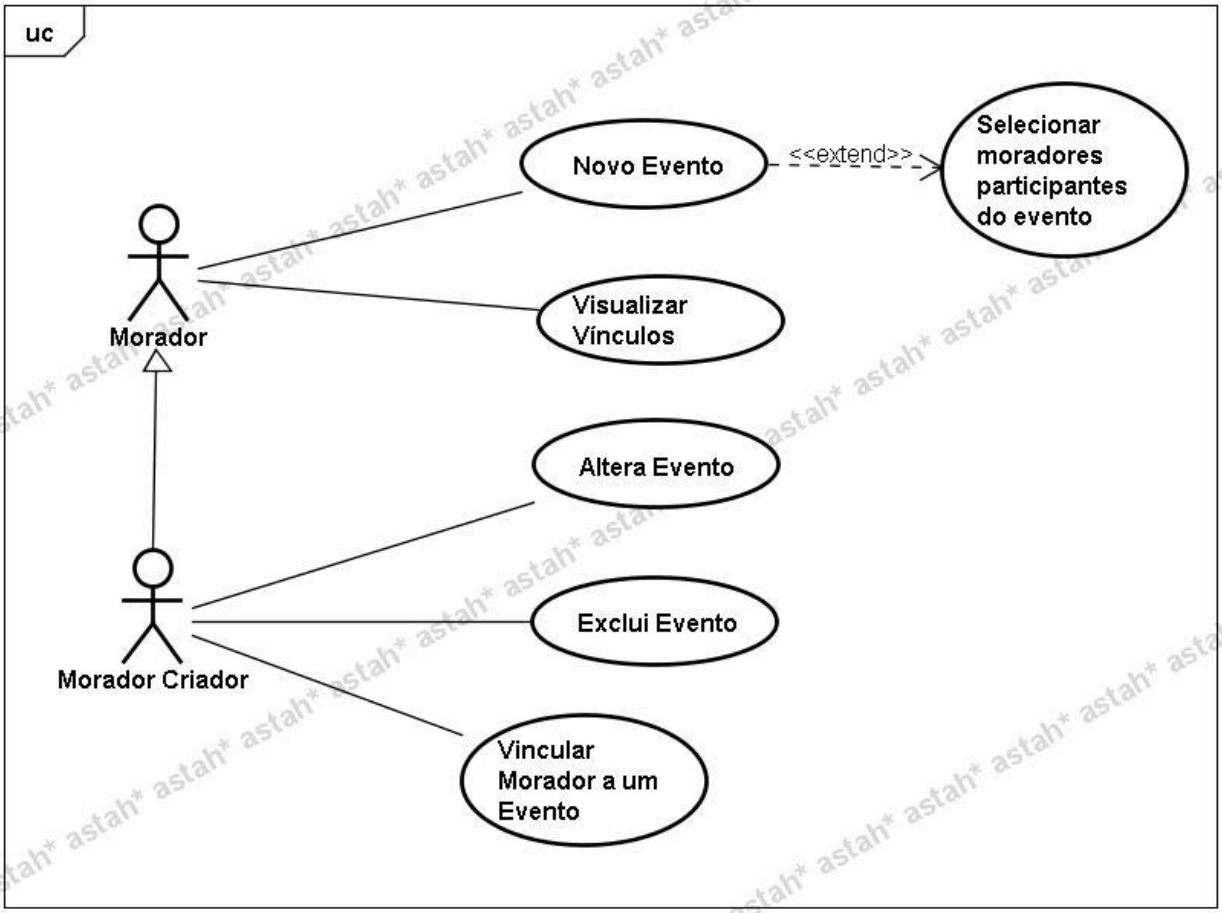
Fonte: Própria Autoria.

Na figura 9 é apresenta o uso referente aos moradores como:

- Cadastra Moradores – Permite cadastrar os moradores da república estudantil.
- Altera Moradores – Opção para alterar os dados de um morador.
- Exclui Morador – Exclui todos os dados de um morador.

3.2.6.5 Caso de Uso referente aos Eventos da Casa

Figura 10 - Caso de Uso Eventos



Fonte: Própria Autoria.

Na figura 10 é apresenta o uso referente aos eventos como:

- Novo Evento – Gera um novo evento para popular a base de dados dos moradores. Devem-se selecionar quais moradores participam deste evento.
- Altera Evento – Altera as informações necessárias de um evento.
- Exclui Evento – Exclui as informações do evento da base de dados.
- Vincular Morador a um evento – Vincula um morador a um determinado e vento, tornando-se responsável pelo mesmo.
- Visualizar vínculos – Destaca os eventos vinculados e seus respectivos responsáveis.

Observa-se que as atividades de alteração, exclusão e vinculação podem ser executadas somente pelo morador criador do evento em questão.

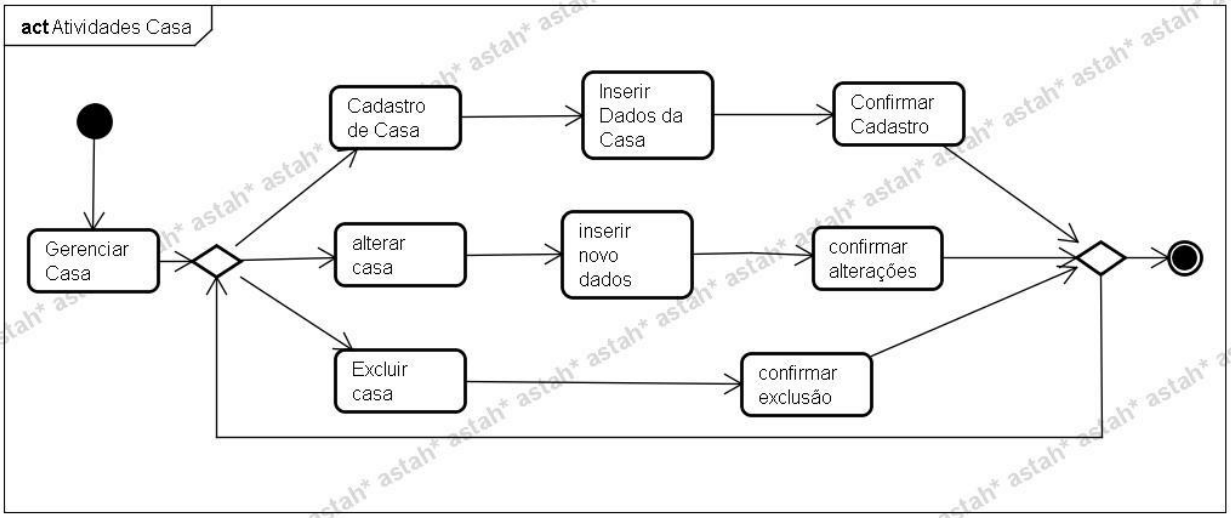
3.2.7 Diagrama de Atividades

Neste tópico será apresenta os diagramas de atividades das principais funcionalidades do protótipo, assim facilitando o entendimento de seu funcionamento.

3.2.7.2 Diagrama de Atividades da Casa

Na figura 11 é demonstrado o fluxo das atividades referentes a casa.

Figura 11 - Diagrama Atividade casa

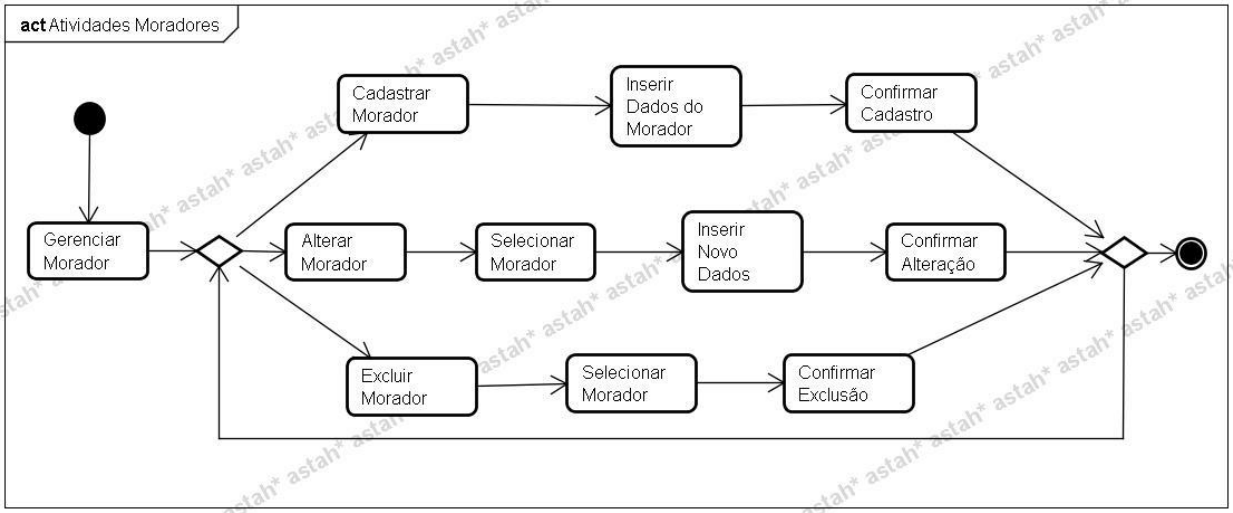


Fonte: Própria Autoria.

3.2.7.3 Diagrama de Atividades dos Moradores

Na figura 12 é demonstrado o fluxo das atividades referentes aos moradores.

Figura 12 - Diagrama Atividade Morador

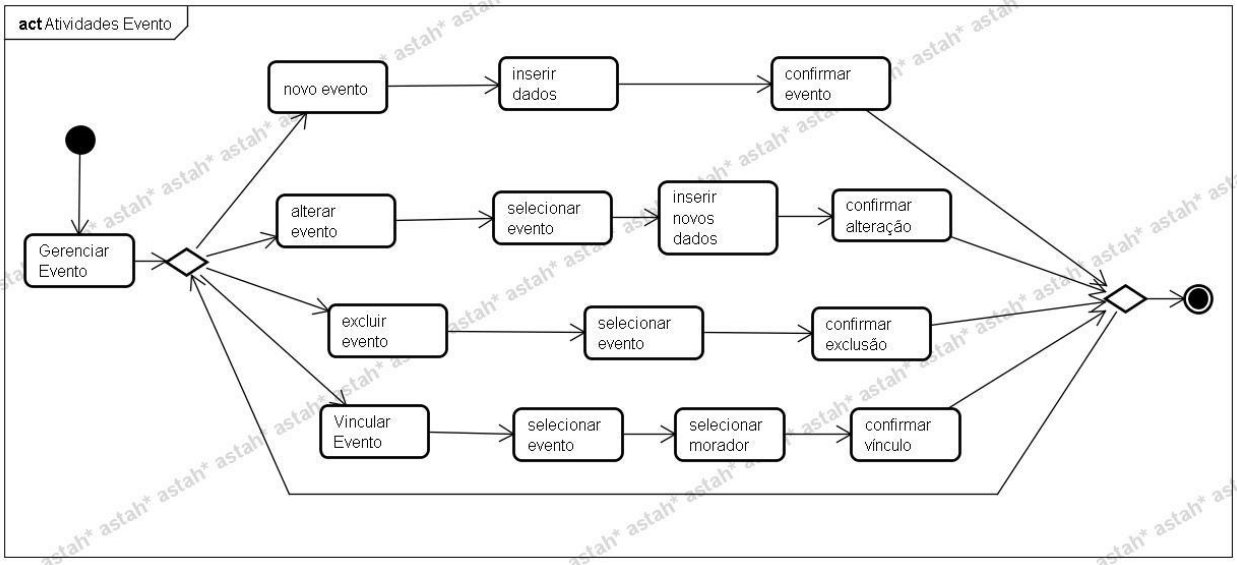


Fonte: Própria Autoria.

3.2.7.4 Diagrama de Atividades dos Eventos

Na figura 13 é demonstrado o fluxo das atividades referentes aos eventos.

Figura 13 - Diagrama Atividades Eventos

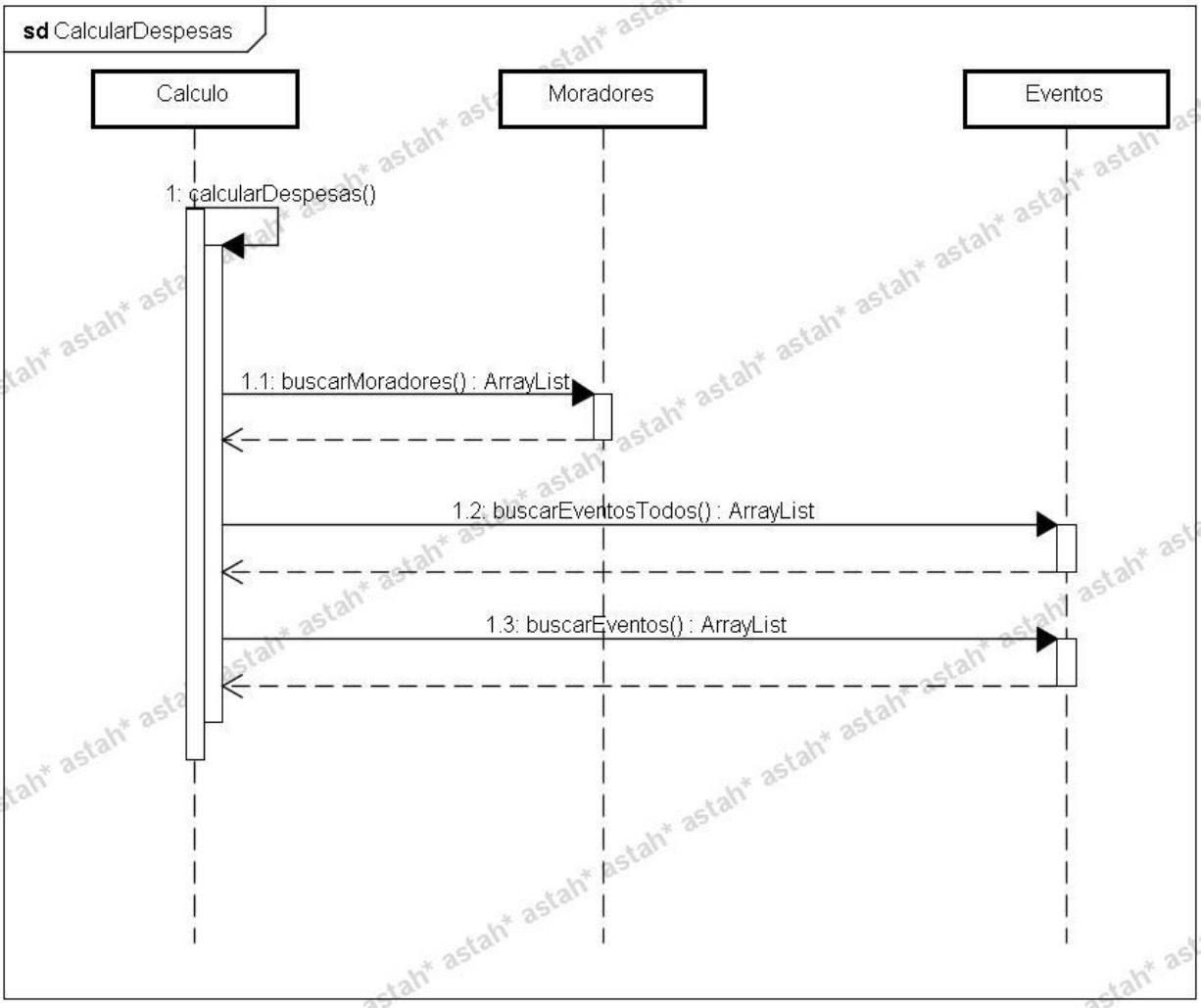


Fonte: Própria Autoria.

3.2.7.5 Cálculo das contas

Na figura 14 é demonstrado através de um diagrama de sequência como é feita a divisão das dívidas entre os moradores.

Figura 14 - Diagrama de Sequência Divisão das Dívidas



Fonte: Própria Autoria.

No diagrama de sequência “CalcularDespesas”,são apresentadas as entidades e sequencias de ações que acontecem na ferramenta para realizar o cálculo parcial das despesas da casa, e do total individual do usuário.

As sequências de ações são mostradas a seguir:

1:calcularDespesas() – para os eventos que todos moradores foram inclusos, se calcula de forma igual para todos, levando em consideração seus créditos, o restante dos eventos são calculados de forma individual, devido a inconstância de seus participantes. 1.1:buscarMoradores busca os dados dos moradores para a execução dos cálculos. 1.2:buscarEventosTodos seleciona somente os eventos que todos os moradores são participantes. 1.3:buscarEventos seleciona os eventos restantes, sendo eles os que não tem todos moradores como participantes.

Para os eventos onde não são todos moradores que participam se divide o valor total do evento entre o número de participantes do mesmo, fazendo isso para cada evento, somando um valor total. Observa-se a impossibilidade de calcular para todos os participantes, devido a não participação de todos moradores em todos os eventos.

3.2.8 Interface

A interface do protótipo é composta de quatorze layouts, onde dois destes são telas de transição, ou seja, compostas apenas por botões que te transferem para outras telas. As telas de transição são a tela principal e a tela referente às atividades da casa, como mostra a figura 15:

Figura 15 - Telas de Transição

Fonte: Própria Autoria.

As outras doze telas são telas onde existe alguma interação de informação com o usuário, são elas:

- Cadastro da casa;
- Alteração da casa;
- Manipulação dos moradores;
- Cadastro dos moradores;
- Alteração dos moradores;
- Manipulação dos eventos;
- Gerar novo evento;
- Alterar evento;
- Seleciona Moradores do Evento;
- Vincular responsável a um evento;
- Visualizar eventos vinculados;
- Visualizar eventos de meses anteriores.

3.2.8.2 Cadastro e Alteração da Casa

O layout de cadastro e de alteração da casa é idêntico, diferindo apenas pelo fato de que na alteração os campos veem preenchidos com os dados atuais da casa. É composto de um *textView* como título do layout, três *editText* para inserção das informações e um botão para confirmar o cadastro, como mostrado na figura 16:

Figura 16 - Cadastro e Alteração da Casa

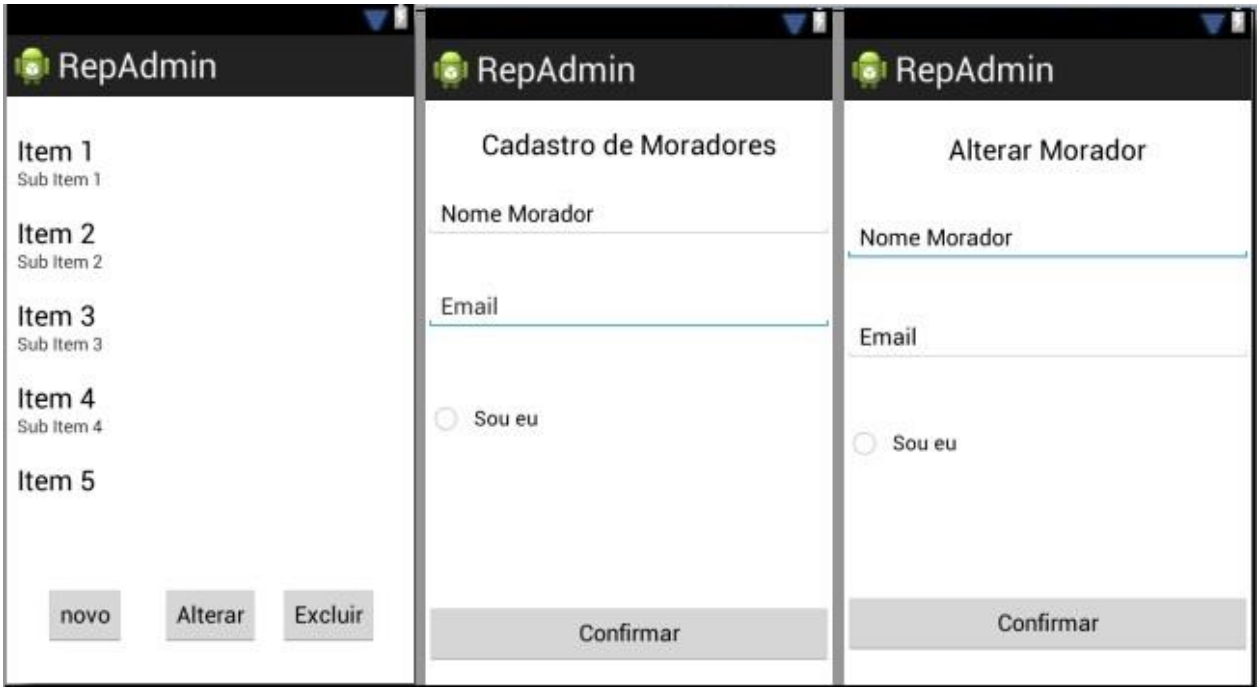
Fonte: Própria Autoria.

3.2.8.3 Manipulação dos Moradores

A manipulação dos moradores é composta de três layouts, o principal, composto por uma *listView*, onde são listados os moradores já cadastrados, onde pode ser selecionado o morador a se manipular, e três botões, para cadastrar novo morador, alterar um morador e excluir morador. As telas de cadastro de moradores e de alteração tem a mesma composição, um *textView* para o título dos layouts, e dois *editTexts*, para

inserção das informações, um *RadioButton* para o usuário identificar qual morador corresponde a ele e um botão para confirmar o cadastro ou alteração, como mostra a figura 17:

Figura 17 - Manipulação dos Moradores



Fonte: Própria Autoria.

3.2.8.4 Manipulação dos Eventos

Para manipular os eventos é utilizado de seis layouts, sendo que o principal é composto por um *listView*, para visualização dos eventos do mês atual e quatro botões, sendo eles para, criar um novo evento, alterar um evento, excluir um evento e vincular um evento a um morador. Os layouts de criação de um evento e de alteração de um evento são semelhantes, contendo, um *textView* para os títulos dos layouts, seis *editText* para a inserção das informações e um botão para confirmação, após a inserção das informações do evento, tanto na criação como na alteração de um evento, o usuário é transferido para a tela de seleção dos moradores participantes do evento em questão. O fator que diferencia é que não tela para um novo evento existe um

radioButton para verificar se o novo evento esta pago ou não, e na tela de alteração existe um botão com a função pagar o evento. Como mostra a figura 18:

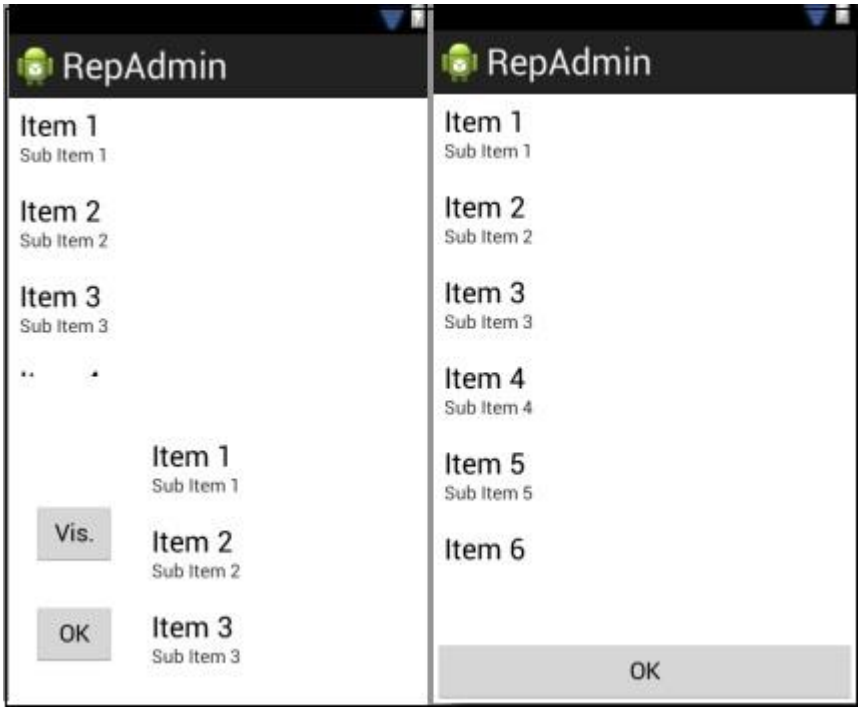
Figura 18 - Manipulação Evento

Fonte: Própria Autoria.

Existem dois layouts para a vinculação de um evento a um morador, o de vinculação e um para visualizar eventos já vinculados. A tela de vincular é composta por dois *listViews*, sendo um para listar os eventos do mês atual e o outro para listar os

moradores, e dois botões, para confirmar a vinculação e outro para levar a tela de visualização dos vínculos, sendo composta por um *textView* para lista todos os evento que foram vinculados e seus moradores responsáveis, assim podendo ser verificado vínculos de meses anteriores, e um botão para confirmação, para poder voltar a tela anterior. Como mostrado na figura 19:

Figura 19 - Telas de Vinculo



Fonte: Própria Autoria.

3.2.8.5 Visualizar Meses Anteriores

A visualização dos eventos de meses anteriores utiliza um único layout, ativado diretamente pela tela principal, contem, um *listView* para listar todos os eventos exceto os do mês atual, que não poderão ser manipulados, servindo somente para verificação, e um botão para confirmação de visualização, voltando a tela inicial, como mostra a figura 20:

Figura 20 - Visualização dos meses anteriores

Fonte: Própria Autoria.

3.2.9 Implementação

Neste tópico será explicado as principais partes dos códigos referentes às funções do protótipo, a implementação foi feite na linguagem JAVA, que é a mais adequada para desenvolvimento na plataforma Android.

3.2.9.2 Implementação Referente a Casa

Para a manipulação da casa são necessários três métodos, para cadastro, alteração e exclusão da mesma.

O primeiro método e o “criarCasa” com a finalidade de salvar os dados da casa no banco de dados, tem com parâmetro uma *String* para o nome da casa, um *Float* para o valor do aluguel e um *Integer* para o dia de fechamento do mês. Como mostra a figura 21:

Figura 21 - Método Cadastra Casa

```
public long criarCasa(String nome, float alugel, int diaFechamento ) {
    ContentValues values = new ContentValues();
    values.put(COLUNA_NOME_CASA, nome);
    values.put(COLUNA_ALUGUEL, alugel);
    values.put(COLUNA_DIA_FECHAMENTO, diaFechamento);

    return mDb.insert(TABELA_CASA, null, values);
}
```

Fonte: Própria Autoria.

Este método é chamado no *Activity* referente ou layout de cadastro da casa, assim que o botão confirmar é acionado, para que o método seja executado é verificado com uma busca SQL se não existe uma casa já cadastrada, utilizando o método ConsultarCasaExistente, como mostrado na figura 22:

Figura 22 - Método Consultar Casa Existente

```
public Cursor ConsultarCasaExistente() {

    return mDb.query(TABELA_CASA, new String[] {COLUNA_ID_CASA}, null, null, null, null, null);
}
```

Fonte: Própria Autoria.

Caso o cadastro seja bem sucedido, um caixa de diálogo aparecerá o alertando, caso já exista cadastro de uma casa outra caixa de dialogo aparecerá. A fim de não poluir o trabalho com códigos, apenas um exemple de criação de caixa de diálogo será apresentado, como na figura 23:

Figura 23 - Caixa de Dialogo Exemplo

```
private void confirma_mensg() {

    AlertDialog.Builder builder = new AlertDialog.Builder(this);

    builder.setTitle("Cadastro da Casa");

    builder.setMessage("Cadastrado Efetuado com Sucesso");

    builder.setPositiveButton("OK", new DialogInterface.OnClickListener() {
        public void onClick(DialogInterface arg0, int arg1) {


        }
    });

    AlertDialog alerta = builder.create();
    alerta.show();


}
```

Fonte: Própria Autoria.

Para alterar as informações da casa é utilizado o método “atualizarCasa”, que tem como parâmetro uma *String* para o nome, um *Float* para o aluguel e um *Integer* para o dia do fechamento do mês, como não é possível o cadastro de mais de uma casa, não a necessidade de comparar a variável de identificação referente a chave primária da tabela Casa, Assim alterando toda linha contendo sua chave primária diferente de nulo, como mostra a figura 24:

Figura 24 - Método de Alteração da casa

```
public boolean atualizarCasa(String casa, float aluguel, int diaFechamento) {
    ContentValues values = new ContentValues();
    values.put(COLUNA_NOME_CASA, casa);
    values.put(COLUNA_ALUGUEL, aluguel);
    values.put(COLUNA_DIA_FECHAMENTO, diaFechamento);

    return mDb.update(TABELA_CASA, values,COLUNA_ID_CASA + "=!",
            new String[] { String.valueOf(null) }) > 0;
}
```

Fonte: Própria Autoria.

O método utilizado para a exclusão da casa é o “removerCasa”, que elimina os dados das casa da base de dados, assim proporcionando o cadastro de uma nova casa. Como sempre haverá semente uma casa cadastrada, não necessidade de verificação do item a ser excluído através do atributo referente à chave primária da tabela Casa, assim excluindo qualquer linha da tabela que tiver sua chave primaria diferente de nulo, como na figura 25:

Figura 25 - Método de Exclusão da Casa

```
public boolean removerCasa() {

    return mDb.delete(TABELA_CASA, COLUNA_ID_CASA + "=!",
        new String[] { String.valueOf(null) }) > 0;
}
```

Fonte: Própria Autoria.

3.2.9.3 Implementação Referente aos Moradores

Para implementação dos eventos referentes a os moradores foram utilizados três métodos principais, podendo cadastrar alterar e excluir um morador.

Para cadastrar é utilizado o método “criarMorador”, onde é efetuada a inserção dos dados do morador no banco de dados, tendo como parâmetros, duas *Strings* consecutivamente para o nome do morador e uma para seu email, a também um parâmetro Booleano de verificação de qual morador é o usuário em questão, como mostrado na figura 26:

Figura 26 - Método de Cadastro de Morador

```
public long criarMorador(String nome, String email, boolean estado ) {
    ContentValues values = new ContentValues();
    values.put(COLUNA_NOME_MORADORES, nome);
    values.put(COLUNA_EMAIL, email);
    values.put(COLUNA_ESTADO_MORADOR, estado);

    return mDb.insert(TABELA_MORADORES, null, values);
}
```

Fonte: Própria Autoria.

Na figura 27 são apresentados os métodos de alteração e exclusão do morador, que são “atualizarMorador” e “removerMorador”. O método de atualização recebe os mesmo parâmetros do método de cadastro, diferenciando apenas pela passagem do parâmetro referente a chave primária do morador selecionado no *listView*, como o método de exclusão que somente recebe a chave primaria do morador como parâmetro.

Figura 27 - Alteração e Exclusão de moradores

```
public boolean atualizarMorador(long idMorador, String nome, String email, boolean estado ) {
    ContentValues values = new ContentValues();

    values.put(COLUNA_NOME_MORADORES, nome);
    values.put(COLUNA_EMAIL, email);
    values.put(COLUNA_ESTADO_MORADOR, estado);

    return mDb.update(TABELA_MORADORES, values, COLUNA_ID_MORADORES + "=?",
            new String[] { String.valueOf(idMorador) }) > 0;

}

public boolean removerMorador(long idMorador) {

    return mDb.delete(TABELA_MORADORES, COLUNA_ID_MORADORES + "=?",
        new String[] { String.valueOf(idMorador) }) > 0;
}
```

Figura 28: (Própria Autoria).

3.2.9.4 Implementação Referente aos Eventos

A manipulação dos eventos da república estudantil são as funcionalidades mais complexas do protótipo, demandando maior número de códigos e métodos, será simplificada sua atividades para melhor explicação e não poluição do trabalho com imagens de códigos fonte.

Para o criação de um novo evento é utilizado três métodos, o método “criarEvento”, que acrescentará as informações do novo evento no banco de dados do morador criador do evento, o método “*WriteFile*”, que escreverá o SQL de cadastro do novo evento em um arquivo de texto que será enviado para os outros moradores através do email e o método “EnviarEmail”, que anexará o arquivo ao email e o enviará.

Como mostra a figura 28, o método “criarEvento”, recebe todas as informações do evento como parâmetro, menos a data atual, que é coletada automaticamente do dispositivo criando um objeto do tipo “GregorianCalendar”, podendo assim manipular datas de diversas formas, e o estado do evento que será sempre “VÁLIDO” para um novo evento.

Figura 28 - Método Novo Evento

```
public long criarEvento(String nome, String comentario,float valor,
                        boolean situacao, int dia, int mes, int ano, String dono ) {

    GregorianCalendar atual = (GregorianCalendar) GregorianCalendar.getInstance();
    int diaC = atual.get(GregorianCalendar.DATE);
    int mesC = atual.get(GregorianCalendar.MONTH) + 1;
    int anoC= atual.get(GregorianCalendar.YEAR);

    ContentValues values = new ContentValues();
    values.put(COLUNA_NOME_EVENTO, nome);
    values.put(COLUNA_COMENTARIO, comentario);
    values.put(COLUNA_VALOR, valor);
    values.put(COLUNA_ESTADO_EVENTO,"VALIDO");
    values.put(COLUNA_SITUACAO, situacao);
    values.put(COLUNA_DIA, diaC);
    values.put(COLUNA_MES, mesC);
    values.put(COLUNA_ANO,anoC);
    values.put(COLUNA_DIAP, dia);
    values.put(COLUNA_MESP, mes);
    values.put(COLUNA_ANOP,ano);
    values.put(COLUNA_DONO_EVENTO, dono);

    return mDb.insert(TABELA_EVENTO, null, values);
}
```

Fonte: Própria Autoria.

Após o cadastramento do evento é chamado o método “*WriteFile*”, que escreverá o arquivo de texto que será mandado por email, recebendo a String SQL por parâmetro, como mostrado na figura 29:

Figura 29 - Método de Escrita do Arquivo de Texto

```
public boolean WriteFile(String text){
    try {
        Context context = null;
        FileOutputStream out = extracted(context).openFileOutput("Evento.txt",
                Context.MODE_APPEND);
        out.write(text.getBytes());
        out.write("\n".getBytes());
        out.flush();
        out.close();
        return true;

    } catch (Exception e) {
        Log.e(TAG, e.toString());
        return false;
    }
}
```

Fonte: Própria Autoria.

Tendo o arquivo de texto pronto é chamado o método de envio de email “EnviarEmail”, que formatará o mesmo e anexará o arquivo ao corpo do email, este método recebe seis parâmetros, onde cinco deles tem a função de criar o código de identificação do email para o protótipo, e o parâmetro restante o email para quem deve se enviar. O código de identificação e composto pelo nome do software e das palavras novo evento escrito tudo junto, seguido do email do criador do evento, data atual, e por fim a chave primária do evento, este código será o assunto do email, sendo de fácil verificação posteriormente. Como mostra a figura 30:

Figura 30 - Método de envio do Email

```
public void EnviarEmail (String email, int dia, int mes, int ano, int codigo, String emailEnv){


String s = email+" "+String.valueOf(dia)+" "+String.valueOf(mes)+
        " "+String.valueOf(ano)+" "+String.valueOf(codigo) ;


    Intent mailIntent = new Intent(Intent.ACTION_SEND);
    mailIntent.setType("text/Message");
    mailIntent.putExtra(Intent.EXTRA_EMAIL  , new String[]{emailEnv});
    mailIntent.putExtra(Intent.EXTRA_SUBJECT, "RepAdminNovoEvento"+s);
    mailIntent.putExtra(Intent.EXTRA_TEXT   , "");
    String FileName = "evento.txt";
    Calculator.generateReport(getActivity(), FileName);



    File attachment = (File) getActivity();
    if (!attachment.exists() || !attachment.canRead()) {
        Toast.makeText((Context) getActivity(),
                       "Attachment Error",
                       Toast.LENGTH_SHORT).show();
        System.out.println("ATTACHMENT ERROR");
    }
    else{
        Uri uri = Uri.fromFile(attachment);
        mailIntent.putExtra(Intent.EXTRA_STREAM, uri);
    }

    try {
       startActivity(Intent.createChooser(mailIntent, "Enviando Email..."));
    } catch (android.content.ActivityNotFoundException ex) {
        Toast.makeText((Context) getActivity(),
                   "O Cliente de Email não esta instalado.",
                   Toast.LENGTH_SHORT).show();
    }
}
```

Fonte: Própria Autoria.

Para se alterar o evento é utilizado três métodos, “consultarEmailDono” verificando se a solicitação de alteração esta sendo feito pelo mesmo morador que gerou o evento. O método “alteraEvento”, que faz uma alteração na coluna referente ao estado do evento que por padrão recebe o valor “VÁLIDO” para “ALTERADO”, utilizando a chave primária do evento, e por fim gerando um novo evento com as alterações utilizando o método de cadastro. Assim o evento alterado permanecendo para verificação, como mostrado nas figuras 31 e 32:

Figura 31 - Método de Verificação Dono do Evento

```
public Cursor consultarEmailDono() throws SQLException {

    Cursor mCursor =

    mDb.query(true, TABELA_MORADORES, new String[] { COLUNA_EMAIL}, COLUNA_ESTADO_MORADOR + "=?",
        new String[] { String.valueOf(true) }, null, null, null,
        null);
    if (mCursor != null) {
        mCursor.moveToFirst();
    }
    return mCursor;

}
```

Fonte: Própria Autoria.

Figura 32 - Método de Alteração do Evento

```
public boolean alteraEvento(int idEvento ) {


    ContentValues values = new ContentValues();

    values.put(COLUNA_ESTADO_EVENTO,"ALTERADO");


    return mDb.update(TABELA_EVENTO, values, COLUNA_ID_EVENTO + "=?",
            new String[] { String.valueOf(idEvento) }) > 0;

}
```

Fonte: Própria Autoria.

A exclusão do evento não o deleta da base de dados, e sim modifica a coluna da base de dado referente ao estado do evento para “EXCLUIDO”, passando como parâmetro a chave primária do evento, assim permanecendo para verificação, como mostra a figura 33:

Figura 33 - Método Excluir Evento

```
public boolean excluiEvento(int idEvento ) {


    ContentValues values = new ContentValues();

    values.put(COLUNA_ESTADO_EVENTO,"EXCLUIDO");


    return mDb.update(TABELA_EVENTO, values, COLUNA_ID_EVENTO + "=?",
            new String[] { String.valueOf(idEvento) }) > 0;

}
```

Fonte: Própria Autoria.

Os métodos de geração do arquivo de texto para o anexo do email, e para o envio propriamente dito utilizados pelas atividades de alteração e exclusão de eventos, são o mesmo utilizado para o cadastro, diferenciando pela String SQL escrita no arquivo de texto anexado ao email.

Para executar o pagamento de um evento é utilizado o método “pagarEvento”, modificando a coluna referente a situação do evento com “true”, utilizando como parâmetro a chave primária do evento selecionado, como mostra a figura 34:

Figura 34 - Método Pagar evento

```
public boolean pagarEvento(int idEvento ) {


    ContentValues values = new ContentValues();

    values.put(COLUNA_SITUACAO, true);


    return mDb.update(TABELA_EVENTO, values, COLUNA_ID_EVENTO + "=?",
        new String[] { String.valueOf(idEvento) }) > 0;

}
```

Fonte: Própria Autoria.

Feito isto, é feita uma atualização no morador que executou o pagamento acrescentando o valor do evento a seus créditos na coluna referente na tabela

“Moradores”. Os métodos de geração do arquivo de texto e envio de emails utilizados são os mesmos das outras atividades referentes a os eventos, alterando somente a String SQL utilizados para escrever o arquivo.

São utilizados dois métodos para a vinculação de responsáveis a um evento e para visualização dos eventos já vinculados. São eles “vinculaEvento” e “consultarEventosVinculados”. Para a execução da vinculação é passa como parâmetro uma String contendo o email do morador em questão e a chave primária do evento. E a visualização é composta por um busca no banco de dados que tem como restrição os eventos que a coluna referente ao responsável pelo evento esta diferente de nulo, como mostram as figuras 35 e 36:

Figura 35 - Método Vincular Evento

```
public boolean vinculaEvento(int idEvento, String email ) {


ContentValues values = new ContentValues();

values.put(COLUNA_RESPONSAVEL_EVENTO,email);


return mDb.update(TABELA_EVENTO, values, COLUNA_ID_EVENTO + "=?",
        new String[] { String.valueOf(idEvento) }) > 0;

}
```

Fonte: Própria Autoria.

Figura 36 - Método Visualizar Eventos Vinculados

```
public Cursor consultarEventosVinculados() throws SQLException {

    Cursor mCursor =

    mDb.query(true, TABELA_EVENTO, new String[]
            { COLUNA_NOME_EVENTO, COLUNA_RESPONSAVEL_EVENTO},
            COLUNA_RESPONSAVEL_EVENTO + "=!",
        new String[] { String.valueOf(null) }, null, null, null,
        null);

    return mCursor;

}
```

Fonte: Própria Autoria.

Todas as atividades referentes aos eventos utilizam do mesmo método de geração de arquivo de texto e envio de email, somente diferenciando a String SQL para Alteração do banco de dados.

3.2.9.5 Cálculo da Divisão das Dívidas

Para a divisão igualitária das dívidas da república é utilizado o método “consultarEventos”, selecionando os eventos utilizando como limitador a coluna referente ao estado do evento, necessariamente tendo que ser “VALIDO”, e a coluna referente a data do criação do evento, verificando se o mesmo é um evento do mês atual, como mostra a figura 37:

Figura 37 - Divisão das Dívidas

```
public Cursor consultarEventos() throws SQLException {

     GregorianCalendar atual = (GregorianCalendar) GregorianCalendar.getInstance();
     int mesC = atual.get(GregorianCalendar.MONTH) + 1;


    Cursor mCursor =

    mDb.query(true, TABELA_EVENTO, new String[] { COLUNA_VALOR}, COLUNA_MES + "=?",
         new String[] { String.valueOf("VALIDO"), String.valueOf(mesC) }, null, null, null,
         null);
    if (mCursor != null) {
         mCursor.moveToFirst();
    }
    return mCursor;

}
```

Fonte: Própria Autoria.

Feito isto, é somado todos os campos reverentes aos valores do evento e dividido pelo número de moradores existente, gerando um valor igualitário para cada um, sendo subtraído pelo valor referente aos créditos de cada morador, gerando um valor justo a se pagar para cada morador.

3.2.9.6 Visualização dos Meses Anteriores

A visualização é gerada a partir do método “consultarMesesAnteriores”, consistindo de um busca na tabela “Eventos” selecionando todos evento exceto os valor do mês atual, servindo apenas para verificação. Como mostra a figura 38:

Figura 38 - Buscando Meses Anteriores

```
public Cursor consultarEventosAnteriores() throws SQLException {

     GregorianCalendar atual = (GregorianCalendar) GregorianCalendar.getInstance();
    int mesC = atual.get(GregorianCalendar.MONTH) + 1;


    Cursor mCursor =

    mDb.query(true, TABELA_EVENTO, new String[] { COLUNA_NOME_EVENTO}, COLUNA_MES +"=!",
        new String[] { String.valueOf(mesC) }, null, null, null,
        null);

    return mCursor;

}
```

Fonte: Própria Autoria.

Todas as atividades quando concluídas com êxito exibem suas respectivas caixas de diálogos confirmando a atividade em questão.

4 VALIDAÇÃO DE USO DO PROTÓTIPO

Neste tópico será apresentada a validação de uso do protótipo, para esta avaliação foi selecionado a metodologia de qualidade de software de McCall, esta metodologia visa avaliar três principais características do software, sendo elas relacionadas à usabilidade do software, alteração do produto e transição do produto.

Foram utilizadas apenas as características relacionadas com o fator de usabilidade do software, tendo como foco os tópicos:

- Correção (o quanto o software satisfaz os objetivos);
- Confiabilidade (o quanto é preciso na execução de suas funções);
- Integridade (controle de acesso aos dados do software);
- usabilidade (esforço necessário de aprendizagem para o uso do software).

A execução do uso da ferramenta foi feita utilizando três dispositivos celulares, sendo eles, *Galaxy* Note1 e dois *Galaxys* S3, todos contendo a versão 4.0 do Android denominada *Icecream sandwich*, servindo perfeitamente para os teste pelo fato do protótipo ter sido desenvolvido tendo como requisito mínimo a versão 2.3 do Android.

4.1 Correção

O Protótipo foi desenvolvido especificamente para resolver os principais problemas existentes em uma república estudantil, como, comunicação entre os moradores, divisão igualitária das dívidas, armazenamento de fácil acesso das informações. Tendo suas funcionalidades com foco específico na resolução destes problemas, constata-se que o software satisfaz os requisitos necessários para a gestão de uma república.

4.2 Confiabilidade

Não foram executados testes para avaliar o protótipo desenvolvido, assim sendo impossível a constatação de exceções recorrentes, porem, através do uso do software observou-se o funcionamento preciso das funções se executado de forma corriqueira, ou seja, não geram ocasiões distintas das previstas dentro de uma repúblicas estudantil.

4.3 Integridade

O controle ao acesso as informações é feito na implementação do software, sendo possível a alteração dos eventos somente pelo usuário que o gerou, exceto se a alteração for a funcionalidade de pagamento do evento. Os dados referentes a casa e aos moradores podem ser alterados por todos, devido a estarem na base de dados interna de cada usuário.

4.4 Usabilidade

O software foi desenvolvido visando facilitar seu uso, não gerando interfaces rebuscadas, ou excesso de informações a serem inseridas. Porem, não foram feitos testes de uso por usuários finais para a verificação da facilidade de uso da ferramenta, ou o esforço necessário para o aprendizado da manipulação da mesma.

5 CONSIDERAÇÕES FINAIS

Neste tópico serão apresentada a conclusão do trabalho, suas dificuldades e o planejamento de trabalhos futuros.

5.1 Dificuldades

As dificuldades encontradas neste trabalho foram o pouco estudo cientifico sobre repúblicas estudantis e moradias semelhantes, escassa existência te material didático que se refere à manipulação de emails para programação Android e dificuldade em encontrar aplicativos para dispositivos moveis distintos, para uma melhor análise para geração de requisitos, devido a grande quantidade de softwares muito semelhantes.

5.2 Conclusão

Ao término deste trabalho conclui-se que existem diversas formas de administrar uma república de estudantes, porem não existe alguma ferramenta ou software que tenha como objetivo auxiliar essa gestão. Assim, o desenvolvimento deste protótipo não só pode auxiliar como proporcionar o desenvolvimento de outras ferramentas com o intuito de auxiliar estudantes e jovens em geral, tanto como universitários como em seu início de carreira profissional, também o desenvolvimento de ferramentas mais voltadas para a colaboração ao invés de ferramentas individuais.

O protótipo desenvolvido teve satisfatória funcionalidade, cumprindo todos os requisitos referentes à resolução dos problemas propostos no trabalho, tendo seu desenvolvimento descrito de forma a se entender seus principais pontos. Utiliza uma forma distinta para a troca de mensagens e comunicação do software, que dispensa a utilização de um servidor, se mostrando eficiente, porém, dependente do pleno funcionamento dos servidores de emails.

5.3 Trabalhos Futuros

Nesta seção serão apresentados possíveis trabalhos futuros recorrentes deste proposto. Devido ao foco em solucionar o problema proposto, não se atentou a outras possíveis funcionalidades que acrescentariam ao software como poder tirar fotos dos comprovantes de pagamentos, marcarem a localização nos mapas disponíveis em um dispositivo móvel entre outras funcionalidades. Devido a escolha do tipo de troca de mensagens pelos anexos dos emails, possibilita a comparação e avaliação deste com outros tipos de procedimentos de comunicação para o software analisando qual pode ser mais útil em quais tipos de ferramentas. Portanto, a proposta de trabalhos futuros é, o desenvolvimento de outras funcionalidades para acrescentar à ferramenta desenvolvida e a analise em comparação do método de comunicação utilizado no trabalho com outros métodos.

REFERÊNCIAS

Bardini, T. *Bootstrapping: Douglas Engelbart, coevolution, and the origins of personal computing*. Stanford University Press, Stanford, CA, USA, 2000.

Barre, R.. Manual de Economia Política. Rio de Janeiro: Fundo de Cultura, 1963.
Bornstein, D. . Dalvik virtual machine. Disponivel em: <http://www.dalvikvm.com/>. Acessado em 15/02/2013.

Brady, P. Anatomy and Physiology of an Android. Disponível em: <http://sites.google.com/site/io/anatomy--physiology-of-an-android>. Apresentação no evento \Google I/O Sessions Videos and Slides". 2008.

Eclipse inicial uma introdução ao popular IDE livre. Java Magazine, n. 4, ano 1,2002. Disponível em: <http://www.devmedia.com.br/post-8906-Artigo-Java-Magazine-04-Eclipse-inicial-uma-introducao-ao-popular-IDE-livre.html>. Acessado em: 15/02/2013.

Ellis, C.A; Gibbs, S.J; Rein, G. “Groupware: some issues and experiences”, Communications of the ACM, v. 34, n. 1, 1991.

Farshchian B. A; Divitini, M. *Handbook of Ambient Intelligence and Smart Environments. Chapter Collaboration Support for Mobile User in Ubiquitous Environments*. Springer Verlag, 2010.

Fernandes, C; Claro, D; Veiga, M. A vida dos estudantes universitários na Idade Média. Lisboa: Universidade de Lisboa, 2003. Disponível em: <http://www.educ.fc.ul.pt/docentes/opombo/hfe/momentos/medieval/estudantes/index.htm>. Acessado em: 19/02/2013.

Fox, D; Box, J. Building solutions with the Microsft .NET Compact Framework: Architecture and Best Practices for Mobile Development, (2003).

Google Inc. Android open source. Disponível em: <http://source.android.com. Acessado em 27/02/2013.

Google Inc. Application fundamentals. Disponível em: <http://developer.android.com/guide/topics/fundamentals.html>. Acessado em 27/02/2013.

Lakatos, E. M; Marconi, M. A. Fundamentos de Metodologia Científica. 7. Ed. – São Paulo: Atlas, 2010.

Laurillau, Y; Nigay, L. Clover architecture for groupware. Proceedings of the Conference on Computer-Supported Cooperative Work (CSCW 2002), 2002.

Lecheta, R. R. Google Android - Aprenda a criar aplicações para dispositivos móveis com o Android SDK. 2009.

Malanos, G. Teoria Econômica. Rio de Janeiro: Forum, 1967.

Mateus, G.; Loureiro, A. Introdução a Computação Móvel. Rio de Janeiro: NCE/UFRJ, 11ª Escola de Computação, 1998.

Naspolini, R. B. As primeiras faculdades de Direito: São Paulo e Recife. Florianópolis: 10 mai. 2008. Disponível em: <http://www.investidura.com.br/biblioteca-juridica/artigos/historia-do-direito/5-asprimeiras-faculdades-de-direito-sao-paulo-e-recife.html>. Acessado em: 27/02/2013.

Neyem, A; Ochoa, S. F; Pino, J. A.*Groupware: Design, Implementation, and Use*. In Briggs et al. Spinger Verlag, Berlin, Heidelberg, 2008.

Nielsenwire. U.S. Smartphone Market: Who's the Most Wanted? http://blog.nielsen.com/nielsenwire/?p=27418. Acessado em 29/02/2013.

Pimentel, M. RUP-3C-Groupware: um processo de desenvolvimento de groupware baseado o Modelo 3C de Colaboração. Tese de Doutorado, Departamento de Informática, Pontifícia Universidade Católica do Rio de Janeiro (PUC-Rio), 22 de março de 2006.

Primo, A; Brambilla, A.M. Social Software e produção do conhecimento. Disponível em: <http://hdl.handle.net/1904/17782> Acesso em : 20/09/2013.

Project, A. O. S. Android open source. Disponível em: <http://source.android.com>. Acessado em 15/02/2013.

Project, A. O. S. What is android? Disponível em: <http://developer.android.com/guide/basics/what-is-android.html>. Acessado em 29/02/2013.

Repolês, Maria Fernanda S. Início de pesquisa nas repúblicas de Ouro Preto: Como funcionam as relações de poder? In: MACHADO, Otávio Luiz (Org.). As repúblicas de Ouro Preto e Mariana: trajetórias e importância. Recife: Universidade Federal. 2007.

Sardi, Jaime A. Virtù et appetitus: aprendendo a conviver com o prazer. In: MACHADO, Otávio Luiz (Org.). As repúblicas de Ouro Preto e Mariana: Trajetórias e Importância. Recife: Universidade Federal de Pernambuco, 2000

Tanenbaum, A. S. (2003). Sistemas Operacionais Modernos. São Paulo. 2003.

Tiffany, R. SQL Server CE Database Development with the .NET Compact Framework. estados Unidos da América, APress, (2003).

Vilaça, M. L. C. Pesquisa e Ensino: Considerações e Reflexões Revista E-scrita. Volume 1. Número 2. Maio-Agosto de 2010.

Viller, S. e Sommerville, I. (2000): “Ethnographically informed analysis for software engineers.

ANEXO A

Levantamento de Requisitos

Analista:	Lucas Henrique Fernandes
Data Levantamento:	27/11/2013
Sistema:	Gerenciador de Repúblicas Estudantis

Solicitação

Deve ser feita a divisão das dividas entre todos os moradores chegado o fim do mês, levando em consideração se houve pagamento de alguma conta pelos usuários, assim sendo descontado o valor da conta ou das contas pagas pelo morador que o fez.

SITUAÇÃO ATUAL

Atualmente esta divisão é feita de forma manual ou utilizando editores de planilhas para esse controle, o que o torna instável, pelo fato de depender do conhecimento do usuário para a edição destas planilhas.

OBJETIVO A SER ATINGIDO

O Objetivo é fornecer aos usuários os valores referentes a cada morador automaticamente, levando em consideração cada atividade paga ou não pelos mesmos.

REQUISITOS

Identificador	Tipo	Descrição	Importância
1	F	O sistema deve coletar os valores dos eventos referentes ao o mês em questão, executando a somatória destes valores e dividindo pelo número de usuários cadastrados.	Fundamental
2	F	Deve-se subtrair do valor gerado pelo identificador 1, o valor em credito encontrado em cada morador.	Fundamental

Levantamento de Requisitos

Analista:	Lucas Henrique Fernandes
Data Levantamento:	27/11/2013
Sistema:	Gerenciador de Repúblicas Estudantis

Solicitação

Criar um módulo de troca de mensagens para o aplicativo utilizando emails entre os usuários para a atualização do banco de dados.

SITUAÇÃO ATUAL

Atualmente se é utilizado de diversas formas para a troca destas informações, sendo pessoalmente ou utilizando redes sociais e emails, não tendo um frequência fixa para isso.

OBJETIVO A SER ATINGIDO

O objetivo é disponibilizar as informações para todos os usuários sempre que houver um acontecimento referente a república estudantil, de forma a todos os moradores estejam cientes e participem desta gestão.

REQUISITOS

Identificador	Tipo	Descrição	Importância
1	F	Escrever as instruções de manipulação de um evento referente a república estudantil em arquivo de texto.	Fundamental
2	F	Anexar o arquivo de texto ao email que será enviado para os outros usuários.	Fundamental
3	F	Enviar emails para todos os moradores cadastros.	Fundamental

Levantamento de Requisitos

Analista:	Lucas Henrique Fernandes
Data Levantamento:	27/11/2013
Sistema:	Gerenciador de Repúblicas Estudantis

Solicitação

O sistema deve permitir o pagamento das dividas registradas, adicionando o valor desta dívida para o morador que a pagou.

SITUAÇÃO ATUAL

Atualmente o controle das dividas pagas ou não pagas é feito de forma manual ou através de controle por planilhas.

OBJETIVO A SER ATINGIDO

O objetivo é fornecer ao usuário a possibilidade de pagar as dividas da casa, classificando-a como paga e atribuindo seu valor como credito ao morador em questão, para a verificação quando feita a divisão das dívidas.

REQUISITOS

Identificador	Tipo	Descrição	Importância
1	F	O sistema deve marcar a situação do evento como pago.	Fundamental
2	F	Deve-se atribuir os valor da dívida paga ao morador que efetuou o pagamento.	Fundamental

Levantamento de Requisitos

Analista:	Lucas Henrique Fernandes
Data Levantamento:	27/11/2013
Sistema:	Gerenciador de Repúblicas Estudantis

Solicitação

Desenvolver um método que possibilite a vinculação de um morador a um evento já cadastrado, responsabilizando este morador pelo evento.

SITUAÇÃO ATUAL

Este tipo de vínculo normalmente é feito de forma verbal, não sendo feito registro destes vínculos.

OBJETIVO A SER ATINGIDO

Fornecer a possibilidade de responsabilizar um morador por um evento, sendo possível a visualização destes vínculos por todos usuários.

REQUISITOS

Identificador	Tipo	Descrição	Importância
1	F	O sistema de acrescentar as informações do evento o morador selecionado com seu responsável.	Fundamental
2	F	Fornecer a visualização destes vínculos a todos moradores.	Fundamental

Levantamento de Requisitos

Analista:	Lucas Henrique Fernandes
Data Levantamento:	27/11/2013
Sistema:	Gerenciador de Repúblicas Estudantis

Solicitação

Desenvolver uma interfaces e opções que permitam ao usuário a visualização dos dados reverente aos eventos da república estudantil.

SITUAÇÃO ATUAL

A visualização dos eventos é feita na maioria das vezes através de planilhas onde são armazenadas estas informações, as informações disponibilizadas varia devido a dependência do formado que a planilha é formatada.

OBJETIVO A SER ATINGIDO

Disponibilizar para o usuário a visualização dos dados atualizados sempre que for necessário.

REQUISITOS

Identificador	Tipo	Descrição	Importância
1	F	O sistema de fornecer a visualização dos dados sempre que necessário.	Fundamental

ANEXO B

Roteiro de Entreviste Semi-Estruturado

- Introdução;
- Tempo de moradia em repúblicas estudantis;
- Quantidade de repúblicas estudantis em que morou;
- Administração da república estudantil;
- Comunicação das informações;
- Divisão das dívidas;
- Pagamento das dívidas;
- Conclusão.

ANEXO C

Nome:_____
Universidade:_____
Curso:_____

Neste documento é apresentado um questionário para verificação dos requisitos levantados para o desenvolvimento do aplicativo de gestão financeira de repúblicas estudantis para a plataforma Android.

1 - A gestão financeira de repúblicas estudantis seria melhor executada se houvesse uma ferramenta onde é possível comunicar os acontecimentos referente a moradia de forma dinâmica, sendo compartilhada com todos os moradores?

_____
_____
_____
_____
_____
_____
_____

2 - O armazenamentos das informações referentes a administração de repúblicas estudantis seria melhor organizado e de fácil acesso se armazenadas em um único software, podendo ser visualizadas de um dispositivo móvel em qualquer localidade?

_____
_____
_____
_____
_____
_____

_____

_____

3 - A divisão das dívidas de uma república estudantil seria facilitada se o software executasse os cálculos, levando em consideração os moradores que executarão pagamentos e os que ainda não executaram, assim sendo uma divisão igualitária para todos os moradores? Não necessitando de cálculos manuais ou da configuração de planilhas para isso.

_____

_____

_____

_____

_____

_____

_____

_____

_____

_____

4 - a possibilidade de vincular um dos moradores como o responsável por determinado acontecimento referente a república estudantil, sendo possível a verificação de todos estes vínculos, ajudaria no controle desta moradia?

_____

_____

_____

_____

_____

_____ _____

Assinatura do requerente Assinatura do requerido