UNIVERSIDADE FEDERAL DO PIAUÍ – UFPI
CAMPUS SENADOR HELVÍDIO NUNES DE BARROS
CURSO DE BACHARELADO EM SISTEMAS DE INFORMAÇÃO

LOCASA: UM *APP* PARA CONTROLE DO PAGAMENTO DE ALUGUÉIS

GABRIEL DE LIMA LEAL

PICOS – PI
2016

GABRIEL DE LIMA LEAL

LOCASA: UM *APP* PARA CONTROLE DO PAGAMENTO DE ALUGUÉIS

Monografia apresentada ao Curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros como requisito final na aquisição do grau de Bacharel em Sistemas de Informação.
Orientador: Prof. Esp. Leonardo Pereira de Sousa.

PICOS – PI
2016

FICHA CATALOGRÁFICA
Serviço de Processamento Técnico da Universidade Federal do Piauí
Biblioteca José Albano de Macêdo

L435l Leal, Gabriel de Lima.

Locasa: um *APP* para controle de pagamento de aluguéis / Gabriel de Lima Leal.– 2016.
CD-ROM : il.; 4 ¾ pol. (56 f.)
Trabalho de Conclusão de Curso (Curso Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2016.
Orientador(A): Prof. Esp. Leonardo Pereira de Sousa

1. *Smartphones*-Aplicativo. 2.*Android*. 3.Dispositivos Móveis-Aluguéis. I. Título.

CDD 005.2

APLICATIVO ANDROID PARA AUXÍLIO NO CONTROLE DE PAGAMENTO DO ALUGUEL DE IMÓVEIS

GABRIEL DE LIMA LEAL

Monografia Aprovada como exigência parcial para obtenção do grau de Bacharel em Sistemas de Informação.

Data de Aprovação

Picos – PI, 20 de Janeiro de 2017

Leonardo Pereira de Sousa
Prof. Esp. Leonardo Pereira de Sousa
Orientadora

Filipe Fontinele de Almeida
Prof. Me. Filipe Fontinele de Almeida
Membro

Algeir P. Sampaio
Prof. Me. Algeir Prazeres Sampaio
Membro

Dedico este trabalho primeiramente à minha mãe Maria Elisa de Lima Leal, ao meu pai Manoel de Sousa Leal (*in Memorian*), à minha irmã Emanuella de Lima Leal que desde sempre foi inspiração, aos meus tios e tias que foram fonte de força em toda caminhada e grandes responsáveis por esta conquista.

AGRADECIMENTOS

Antes de tudo, agradeço a Deus por ter me guiado e dado forças em toda a jornada para que pudesse alcançar o objetivo final.

À minha mãe Maria Elisa de Lima Leal que me proporcionou a melhor orientação que podia ter, por ser um exemplo de mãe, mulher, profissional e pessoa.

À minha irmã Emanuella de Lima Leal por ser um norte em minha formação profissional e como pessoa.

Aos meus avós paternos e maternos, principalmente ao meu avô paterno Pedro Borges Leal por ser a melhor pessoa do mundo, e por me ensinar que temos que ser bons independente de tudo.

Aos meus tios e tias, em especial à minha tia Eva de Sousa Leal Silva por acreditar no meu potencial e proporcionar a melhor educação possível.

Aos meus primos e primas, por terem contribuído na minha educação como um todo, principalmente ao Erasmo Artur da Silva Junior pelo apoio técnico.

Aos meus amigos e companheiros de estrada, que sempre me auxiliaram e ajudaram nas dificuldades enfrentadas, principalmente à minha amiga Tatiele Veloso da Silva que sempre será presente em minha vida.

À minha namorada Larissa Ingrid de Araújo Sousa por sempre me auxiliar e sempre ser fonte de foco e perseverança.

A todo corpo docente do curso de Bacharelado em Sistemas de Informação do Campus Senador Helvídio Nunes de Barros em Picos pelas orientações.

Ao meu orientador Leonardo Pereira de Sousa por ser fonte de conhecimento e valiosas orientações.

A todos que contribuíram de alguma forma na jornada que se encerra nesta etapa.

A todos meu muito obrigado!

“A palavra comemorar remete quase sempre ao verbo festejar, entretanto, comemorar significa memorar com outros, ou, em outras palavras, lembrar junto.”

Mario Sérgio Cortella

RESUMO

A popularização de dispositivos móveis fez com que investimentos para a criação de *smartphones* se tornasse um mercado lucrativo. Com isso, sistemas operacionais mais robustos surgiram, fornecendo ao cliente experiência agradável em seu uso e execução de aplicações variadas, incorporando ainda mais tais aparelhos na vida cotidiana das pessoas. Em consequência à evolução deste mercado, o desenvolvimento de aplicações móveis também apresentou grande crescimento. Em uma era onde as pessoas necessitam enviar e receber e-mails ou mesmo realizar transferências bancárias a todo momento e de qualquer lugar este tipo de aplicação surgiu como agradável solução. Este trabalho apresenta o desenvolvimento de um aplicativo para dispositivos móveis com sistema operacional *Android* que auxilie locadores no controle de pagamento do aluguel de imóveis. Para a construção da aplicação foram utilizadas a modelagem UML, a ferramenta *Android Studio 2.1.2*, e o *SQLite*, banco de dados padrão do *Android*, para persistência dos dados. Após realização de testes constatou-se que a aplicação auxilia por completo a tarefa realizada pelo locador, dando-lhe funções de armazenamento e acesso às informações relevantes relacionadas a tal atividade.

Palavras-chave: *Smartphones*, *Android*, Dispositivos Móveis, Aluguel, Imóveis.

ABSTRACT

The popularization of mobile devices has made investments for the creation of smartphones to become a lucrative market. With this more robust operating systems have emerged, providing the customer enjoyable experience in its use and execution of varied applications, incorporating even more such devices into people's daily lives. As a consequence of the evolution of this market, the development of mobile applications also showed great growth. In an age where people need send and receive emails or even make bank transfers at any time and from anywhere this type of application emerged as a pleasant solution. This work presents the development of an application for mobile devices with Android operating system that helps owners in payment control of rental of real estate. For the construction of the application were used the modeling UML, the tool Android Studio 2.1.2, and the SQLite, Android standard database, for data persistence. After conducting tests it was found what the application assists completely the task performed by the lessor, giving him functions of storage and access to information relevant related to such activity.

Keywords: Smartphones, Android, Mobile Devices, For Rent, Properties.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

SI	Sistema de Informação
OS	*Operational System*
GPS	*Global Positioning System*
iOS	*iPhone Operating System*
UNIX	*Multi-User Exploration System*
TV	Televisão
ES	*Embedded Systems*
MSDN	*Microsoft Developer Network*
NFC	*Near Field Communication*
RIM	*Research in Motion*
BBM	*Blackberrymessenger*
EDGE	*Enhanced Date Rates GSM Evolution*
PDA	*Personal Digital Assistant*
OHA	*Open Handset Alliance*
SDK	*Software Development Kit*
NDK	*Native Development Kit*
HAL	*Hardware Abstraction Layer*
ART	*Android Runtime*
API	*Applications Programming Interface*
IDE	*Integrated Development Environment*
EUA	Estados Unidos da América
ADT	*Android Development Tools*
APK	*Android Package*
XML	*Extensible Markup Language*
SGBD	Sistema Gerenciador de Banco de Dados
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
JDK	*Java Development Kit*
JRE	*Java Runtime Environment*
SDK	*Software Development Kit*
ER	Entidade Relacionamento
MER	Modelo Entidade Relacionamento
APP	*Application*

CPF	Cadastro de Pessoa Física
MB	*Megabyte*
GB	*Gigabyte*
MHz	*Megahertz*
GHz	*Gigahertz*
RF	Requisitos Funcionais
RFN	Requisitos Não Funcionais
RN	Regras de Negócio

SUMÁRIO

1 INTRODUÇÃO

Os sistemas de informação surgiram devido à necessidade que as pessoas tinham de armazenar elementos importantes relacionados às suas atividades diárias. Antes da popularização dos computadores (e dos SI's) os dados eram armazenados em papel, ainda que apresentasse um método de fácil manipulação, exigia que existisse uma pessoa exclusivamente responsável em lidar com os arquivos, e atividades simples como atualização e recuperação dos dados eram muito dispendiosas.

Considera-se que a era da informação surgiu por volta do século XX. A globalização e consequentemente o aumento das transações entre empresas de todas as partes do mundo, ocasionou no aumento exponencial da quantidade de informações que auxiliavam na administração das organizações, e que deveriam ser armazenadas. Com isso o método de gestão das informações que era utilizado até o momento começou apresentar limitações operacionais devido ao grande volume de dados.

Na década de 70, com o surgimento dos microprocessadores, os computadores diminuíram drasticamente de tamanho, o que ajudou na sua popularização, surgiram como um divisor de águas na manipulação de grandes quantidades de informações. Sua popularização juntamente com a internet foi consolidada no século XX, possibilitando que a maioria das empresas implantassem um sistema de informação para auxiliar na gestão das informações e serviços oferecidos. Com isso puderam trabalhar de forma rápida e concisa nos dados importantes à gestão das organizações, o que pode determinar seu sucesso ou fracasso.

Os sistemas de informação passaram a fornecer recursos para manter grandes quantidades de dados disponíveis o tempo todo para manipulação precisa e segura. Diminuindo o tempo gasto na recuperação dos dados, a quantidade de pessoas necessária para manipulá-las, o custo envolvido, e por último e mais importante o tempo gasto na tomada de decisões.

A constante evolução da tecnologia além de oferecer recursos nunca antes disponibilizados, provocou também sua inserção na vida cotidiana das pessoas, tornando-se uma ferramenta comum. Com o passar dos anos os sistemas de informação deixaram de ser ferramentas que só eram utilizadas para armazenar dados e utilizados por empresas, e se tornaram sistemas que podem auxiliar qualquer pessoa em todas as atividades de seu dia a dia sem importar o grau de complexidade das mesmas, podendo ser de um simples despertador até ferramentas para auxílio de tomada de decisão como dito anteriormente.

Nos últimos anos o mundo da informática vem passando por uma revolução causada pelo movimento *mobile*. Aparelhos diminutos com um vasto oferecimento de recursos e poder

de processamento, dentre processamento de som, áudio e vídeo. Os grandes responsáveis por esse movimento são *smartphones* e *tablets*, que funcionam com sistemas operacionais mais sofisticados, que fazem com que possuam recursos que vão além de realizar uma simples ligação, como ter acesso a rede de computadores devido às funcionalidades disponíveis em seus *softwares.*

A crescente popularização de tais aparelhos gerou amplas demandas que deveriam ser supridas de alguma forma, desta maneira as grandes organizações começaram a fornecer investimentos maciços para o desenvolvimento de *softwares* para estes dispositivos, como sistemas operacionais mais sofisticados e aplicações.

1.1 Objetivo

Desenvolver uma aplicação para dispositivos móveis que execute sobre a plataforma *Android*, para auxiliar locadores no controle de pagamento do aluguel de imóveis.

1.2 Organização do Documento

O trabalho a ser apresentado a seguir encontra-se dividido em 6 (seis) capítulos:

- Capítulo 2 – Referencial Teórico: Onde é apresentada toda orientação teórica referente ao projeto. Contendo dados sobre o surgimento e ascensão dos dispositivos móveis, bem como informações sobre sistemas operacionais móveis e desenvolvimento *Android.*
- Capítulo 3 – Aplicativo LoCasa: Trata de como foi feito o levantamento de requisitos para a compreensão por completo da tarefa a ser auxiliada, e construção dos modelos sugeridos pela modelagem UML.
- Capítulo 4 – Funcionamento do Aplicativo: Este capítulo apresenta o funcionamento da ferramenta na visão do usuário, apresentando todas as funcionalidades do mesmo.
- Capítulo 5 – Testes: Contém informações relevantes aos testes feitos, configurações dos diferentes aparelhos onde o aplicativo foi testado, e resultados obtidos com base nos testes em campo.
- Capítulo 6 – Conclusões Finais: Apresenta as conclusões alcançadas mediante avaliação da ferramenta desenvolvida, também apresentando possíveis trabalhos futuros.

2 REFERENCIAL TEÓRICO

> O mercado corporativo mundial está em plena ascensão, e diversas empresas estão buscando incorporar aplicações móveis a suas rotinas de trabalho. O objetivo é tornar ágeis seus negócios e integrar as aplicações móveis com seus sistemas. As corporações visam lucro, sendo que os dispositivos móveis podem ocupar um importante espaço em um mundo onde a palavra "mobilidade" está cada vez mais conhecida e utilizada (LECHETA, 2010).

Os *smartphones* começaram a se popularizar a partir dos anos 2000, quando algumas empresas já faziam o uso de dispositivos chamados *Palmtops,* aparelhos que já possuíam características de um organizador pessoal, utilizavam o *Palm OS* como sistema operacional, projetado para fornecer interface gráfica agradável ao usuário e facilidade de uso com telas *touchscreen*. Mas como até então as grandes empresas não tinham interesse por tais aparelhos, o *software* apresentava limitações operacionais, como suporte a multitarefas e segurança defasada.

A transição de celulares comuns para ferramentas com vastos recursos (*smartphones)* fez com que tais aparelhos fossem bem aceitos pelas pessoas, provocando o crescimento do mercado *mobile*. Este fato induziu as organizações a investir nesse ramo de comércio emergente, e foi um dos principais fatores desencadeadores de grandes avanços relacionados a aplicações e dispositivos móveis.

2.1 Tipos de Sistemas Operacionais Móveis

A ascensão dos sistemas operacionais móveis mais sofisticados foi impulsionada pelo surgimento dos *smartphones*.

> Os *smartphones* são dispositivos móveis capazes de realizar grandes variedades de funções: filmar e reproduzir vídeos, acessar a *internet* banda larga por *Wi-fi* ou 3G, GPS, enviar e receber *e-mails*, ler e editar documentos em vários formatos etc (FLING, 2009).

Com a evolução de tais *softwares* surgiram o *Android* (da *Google*), o *iOS* (da *Apple*), o *Windows Phone* (da *Microsoft),* o *BlackBerry* (da *Research in Motion*), e alguns outros sistemas operacionais móveis que não obtiveram tanta popularidade como estes. Cada um possui sua particularidade, pontos positivos/negativos e arquitetura, mas todos possuem funcionalidades em comum como processamento multitarefa, conexão com redes e execução de aplicativos. Tais sistemas, hoje em dia, estão sendo estudados para que limitações como pouca capacidade de memória e tempo de bateria sejam resolvidos, para um melhor funcionamento.

Como citado anteriormente apenas alguns sistemas obtiveram a popularidade desejada, e alguns serão descritos com mais detalhes a seguir.

iOS

A *Apple* foi precursora no lançamento de dispositivos que contavam com sistemas operacionais mais sofisticados, em janeiro de 2007 disponibiliza no mercado o *iPhone* que funcionava com o *iPhone OS*, *software* de código fechado, que foi desenvolvido exclusivamente para *smartphones* da *Apple,* tal dispositivo obteve grande sucesso ficando quase três anos no domínio do mercado. O *software* foi chamado assim por 4 anos, e somente no lançamento de sua quarta versão que recebeu nova denominação, passando a chamar-se *iOS*.

O *iOS* é um sistema operacional da *Apple*, derivado do *Mac OS X*, lançado primeiramente para *iPhone*, idealizado e projetado originalmente para o *iPhone*, também é usado em outros aparelhos da empresa (APPLE, 2011).

Este sistema operacional é derivado do *Mac OS X* que possui como base o *UNIX*, foi desenvolvido inicialmente para o *smartphone iPhone* e logo em seguida adaptado para outros dispositivos móveis da *Apple* (*iPod Touch, iPad e Apple TV*). Feito para funcionar em telas sensíveis ao toque (*multitouch*), inicialmente não dava suporte à multitarefa só possuindo essa funcionalidade a partir de sua quarta versão. Foi um dos pioneiros na comercialização de *smartphones*, por este motivo é considerado um dos sistemas operacionais móveis mais antigos em desenvolvimento até hoje, já estando em sua décima versão.

> No *iPhone OS*, a arquitetura do sistema e as tecnologias são semelhantes às encontradas no *Mac OS X*. O *kernel* do *iPhone OS* é baseado em uma variante do mesmo *kernel* base que é encontrado no *Mac OS X*. No topo deste *kernel* estão as camadas de serviços que são utilizadas para implementar aplicações na plataforma (HUBSCH 2012).

A arquitetura do *iOS* é dividida em *Core OS* (Núcleo do sistema operacional), *Core Services* (Serviços oferecidos pelo sistema), *Media* (como o nome sugere, oferece serviços de mídia como áudio, vídeo, fotos e até o *OpenGL ES*), *Cocoa Touch* (responsável pelas interações com o usuário) (KAHNEY, 2008).

Como foi supracitado sua arquitetura é dividida em quatro camadas, descritas de forma mais detalhada a seguir:

- *Cocoa Touch*: Camada de mais alto nível composta por *frameworks* responsáveis em fornecer a infraestrutura necessária para a realização de operações no sistema operacional;

- *Media*: É responsável por fornecer um ambiente agradável ao usuário, onde encontram-se os recursos gráficos de áudio e vídeo encarregados de criar uma boa experiência e dispositivos móveis;
- *Core Services:* Possui componentes responsáveis em fornecer serviços fundamentais que são utilizados por todas as aplicações que executam no sistema;
- *Core OS:* Camada de mais baixo nível que serve como ponte entre o *hardware* e todas as outras camadas que o sistema possui.

Windows Phone

O *Windows Phone* trata-se de uma versão móvel do *Windows*, desenvolvido pela *Microsoft*, este sistema foi lançado em 2010 com intuito de substituir o *Windows Mobile* lançado em 2000 que não obteve o sucesso almejado. Possui como slogan "*Put people first*" (Colocando as pessoas em primeiro lugar) que deixa claro que o sistema foi desenvolvido voltado para o usuário doméstico.

Windows Phone 8 foi lançado em 11 de dezembro de 2012. "*Apollo*" é o nome de código da nova geração do sistema operacional, como confirmado num seminário da MSDN em agosto de 2011. A atualização adicionou ao sistema a tecnologia NFC (*Near Field Communication*) que permite transferir arquivos de maneira rápida à curta distância (BROCKSCHMIDT, 2012).

O *Windows Phone 8* apresentou como principal vantagem a possibilidade de outros fabricantes licenciarem seus aparelhos para utilizar o sistema operacional, desde que tais dispositivos dessem suporte às funcionalidades que dispõe.

Os *smartphones* da *Microsoft* oferecem recursos de compatibilidade com os aplicativos do *Office* para computadores, como *Word, Excel, Power Point*, dentre outros. Apresenta interface com o usuário chamada de Metro, onde os aplicativos ficam dispostos em um mosaico dinâmico, onde o usuário pode personalizar a posição e tamanho de cada ícone de determinado aplicativo.

BlackBerry

A RIM foi fundada por Mike Lazaridis em 1984, seus aparelhos obtiveram maior sucesso no âmbito empresarial, pois foram desenvolvidos especialmente para este ramo de atividade. Entrou no mercado americano em 1999 com o lançamento de uma espécie de *pager* bidirecional que permitia aos usuários o envio e recebimento de mensagens com outros aparelhos.

> O *BlackBerry* é um sistema operacional concebido pela empresa canadense RIM – *Research in Motion*. Ele integra diversas funções importantes e que foram integradas pela primeira vez em celulares chamados *smartphones*. O que o diferencia dos demais é que ele utiliza um serviço próprio de *e-mail RIM*, chamado BBM (*BlackBerrymessenger*). As mensagens e *e-mail* no envio e recepção chega até 200kbps, utilizando a tecnologia EDGE (BLACKBERRY – RIM, 2013).

Em seguida, no ano 2000, lançou seu primeiro dispositivo móvel comercializável, o *Blackberry RIM 857-957* que ainda não se tratava de um telefone, mas já oferecia aplicativos de um organizador pessoal (bloco de notas, calendário e outros). A RIM consolidou seu sucesso com o lançamento de seus primeiros telefones celulares, em 2002 lançou o *Blackberry 5810* que era um telefone integrado a um organizador pessoal, mas que ainda contava com alguns incômodos no seu uso, como ter que ficar com o fone de ouvido conectado ao aparelho a todo momento para que pudesse realizar ligações, barreira esta que foi sobreposta com o lançamento do *Blackberry 6210* lançado em 2003.

Os dispositivos móveis *Blackberry* apresentaram grande popularidade no meio empresarial, pois ofereciam produtos que possuíam cuidados especiais na segurança, o que interessava aos grandes empresários, até o próprio governo dos Estados Unidos utilizava aparelhos da RIM, estes motivos a fizeram uma das fabricantes de celular mais bem sucedidas do mundo, apresentando faturamentos bilionários.

Este cenário mudou em 2007 quando as vendas caíram devido ao lançamento do *iPhone*, dispositivo que oferecia as mesmas funcionalidades dos *Blackberry* e outros recursos que este não possuía, ficando por muitos anos às margens do mercado. A RIM pretende voltar ao foco da comercialização dos dispositivos móveis com o lançamento do *Blackberry 10*.

Android

Figura 1 – Ícone *Android*
Disponível em: < http://www-usr.inf.ufsm.br/~efilho/include/site.html >

A *Google* observando números alarmantes de popularidade dos *smartphones* decide entrar nesse novo ramo do mercado, e em 2005 compra a *Android Inc.*, pequena empresa do Vale do Silício que trabalhava no desenvolvimento de *softwares* para dispositivos móveis.

A *Android Inc.* já vinha trabalhando em um sistema operacional móvel de código aberto baseado no *kernel* do *Linux*, o *Android*, que trata-se de uma plataforma completa para dispositivos móveis, incluindo sistema operacional, *middleware* (*software* que funciona como mediador entre o sistema operacional e as aplicações que executam sobre o mesmo) e aplicações. A *Google* em 2008 lança o primeiro aparelho que funcionava com o *Android*, não apresentando, no primeiro instante, a popularidade desejada, fato transposto nos anos subsequentes.

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

Nos anos subsequentes a *Google* forma um grupo com mais de 30 empresas fabricantes de dispositivos móveis como celulares, PDA's e *Internet tablets*, a *Open Handset Alliance* (OHA), que é responsável por manter a plataforma *Android.*

> *Open Handset Alliance* é um consórcio de grandes empresas com o objetivo de popularizar e melhorar os dispositivos móveis. A Google é um dos membros do consórcio, continua responsável por controlar importantes etapas do desenvolvimento do sistema como a gerência do produto e a engenharia de processos (OPENHANDSETALLIANCE, 2011).

Um dos pontos determinantes para a popularidade deste sistema foi o fato de ser de código aberto o que gerou grande redução de custos, e possibilitou que empresas da OHA focassem seus investimentos e esforços no que mais agrada o mercado consumidor (velocidade, beleza e qualidade), fato que atualmente é visto nos aparelhos de diferentes marcas, que apresentam *layouts* diferentes, bem como forma de funcionamento dos dispositivos.

> O sistema operacional também está presente em mais de 56% dos *smartphones* em todo o mundo, tornando o *Android* uma das plataformas mais interessantes para o desenvolvimento de aplicativos (Gartner 2012). No *Android*, aplicações são desenvolvidas em linguagem de programação *Java*, utilizando o *Android SDK*, e executam na máquina virtual *Dalvik.* O *Android*, entretanto, possui suporte para o desenvolvimento de aplicações nativas, escritas em C e C++, através do *Android NDK* (BRAHLER, 2010).

Diariamente, segundo a Zeman (2013), cerca de 1,5 milhão de aparelhos que funcionam com o *Android* são ativados, e a *Google Play* (loja oficial de aplicativos para *android*) contabiliza mais de 50 bilhões de *downloads* oficiais mensalmente.

2.2 Arquitetura *Android*

Os sistemas operacionais móveis foram desenvolvidos possuindo particularidades em relação ao modo como seus recursos são oferecidos aos usuários e desenvolvedores, suas arquiteturas diferem entre si, e estas influenciam no funcionamento do sistema bem como no modo como os recursos para o desenvolvimento de aplicativos são disponibilizados.

O *Android* é um sistema operacional móvel desenvolvido para *smartphones* que dispõe de bibliotecas, interface gráfica e ferramentas para o desenvolvimento de aplicações. Constantemente este sistema é chamado de pilha de *softwares*, pois sua arquitetura é composta por cinco camadas empilhadas que vão desde o nível mais baixo (camadas mediadoras entre *software* e *hardware*) ao mais alto (aplicações nativas do sistema), como é demonstrado na Figura 2.

Do nível mais baixo ao mais alto, tem-se:

Kernel Linux: Possui um *software* baseado no sistema operacional *Linux 2.6*, que passou por várias modificações para se adequar às necessidades dos dispositivos móveis, com intuito de otimizar o uso de memória e melhorar o tempo de processamento. Estas mudanças também contam com novos dispositivos de *drivers*, melhor gerenciamento de energia e um recurso que permite o término de processos quando há pouco espaço de memória.

Hardware Abstraction Layer: O HAL é composto por vários módulos de biblioteca, cada qual implementa uma interface para um tipo específico de componente de *hardware*, como a câmera ou *bluetooth* (ANDROID DEVELOPERS, 2016). Esta camada é responsável por abstrair o *hardware*.

Android Runtime: No *Android*, aplicações escritas em *Java* são executadas em sua própria máquina virtual, que por sua vez é executada em seu próprio processo no *Linux*, isolando-a de outras aplicações e facilitando o controle de recursos (ANDROID DEVELOPERS, 2016). Cada aplicação é executada em uma instância da máquina virtual *Dalvik*, que foi melhorada exclusivamente para dispositivos móveis. Desenvolvida pelos engenheiros da *Google* com o objetivo de prover menor consumo de memória e execução isolada de processos.

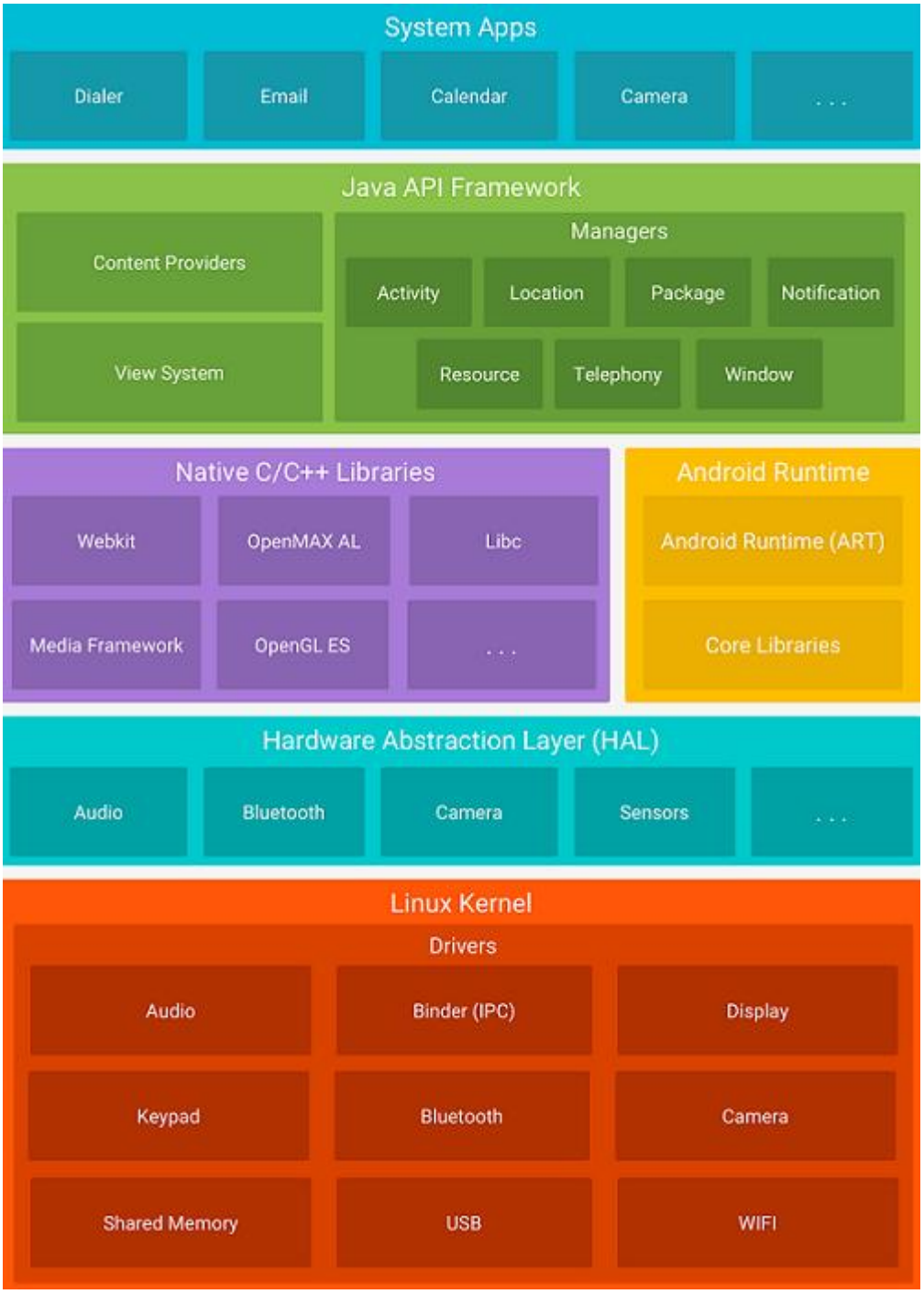

Figura 2 – A Pilha de Software do Android
Disponível em: < https://developer.android.com/guide/platform/index.html >

Native Libraries: Muitos componentes do sistema *Android*, tais como ART e HAL, são construídos a partir de código nativo que necessitam de bibliotecas nativas escritas em C e C++. A Plataforma *Android* fornece API's (*Applications Programming Interface*) para expor a funcionalidade de algumas dessas bibliotecas nativas para *apps* (ANDROID DEVELOPERS, 2016). Ou seja, fornece bibliotecas que dispõem de funcionalidades acessadas pela *framework* de aplicações para serem utilizadas no desenvolvimento de aplicativos.

Framework de Aplicações*:* Esta camada possui as API's que são interfaces de programação de aplicativos utilizadas no desenvolvimento de aplicações *Android.* Como citado anteriormente, cada arquitetura possui suas peculiaridades, uma delas é como são apresentadas as ferramentas para o desenvolvimentos de aplicações para determinada plataforma, e a *Framework* de Aplicações busca simplificar a reutilização de processos e deixar a complexidade alheia ao desenvolvedor.

System Apps: E por fim, estando no topo da pilha de *softwares*, a camada de aplicações dispõe das aplicações nativas do sistema operacional, despertador, calendário, teclado e todos os outros.

2.3 *Android Studio*

Figura 3 – Android Studio
Disponível em: < https://developer.android.com/studio/features.html >

> O *Android Studio* é o ambiente de desenvolvimento integrado (IDE) oficial para o desenvolvimento de aplicativos *Android* e é baseado no *IntelliJ IDEA*. Além do editor de código e das ferramentas de desenvolvedor avançadas do *IntelliJ*, o *Android Studio* oferece ainda mais recursos para aumentar sua produtividade na criação de aplicativos *Android* (ANDROID DEVELOPERS, 2016).

Foi anunciado em Maio de 2013, na conferência Google I/O, evento para programadores organizado pela Google na cidade de São Francisco-EUA. Sua versão beta (ainda em fase de testes disponibilizada ao público) só foi apresentada em junho do ano seguinte, e a primeira compilação estável aconteceu seis meses depois. É disponibilizado de forma gratuita para várias plataformas.

O *Android Studio* dispõe de novos recursos para o desenvolvimento de aplicativos e foi apresentado como alternativa ao *Eclipse ADT*. Várias funcionalidades foram agregadas à ele para que a construção de aplicações se tornasse mais fácil, e a realização dos testes acontecesse de forma mais rápida, como *Instant Run*, emulador rápido, desenvolvimento unificado para diferentes dispositivos, modelos de projetos e integração com *GitHub.*

O *Android Studio* oferece um ambiente unificado para o desenvolvimento de aplicativos pata telefones e *tablets Android* e dispositivos *Android Wear*, *Android TV* e *Android Auto* (ANDROID DEVELOPERS, 2016).

> Por padrão, o *Android Studio* configura novos projetos a serem implantados no emulador ou em um dispositivo físico com apenas alguns cliques. Com o *Instant Run*, podem ser implementadas alterações em métodos e recursos existentes de um aplicativo em execução sem compilar um novo APK. Dessa forma, as alterações no código podem ser visualizadas quase instantaneamente (ANDROID DEVELOPERS, 2016).

Cada projeto no *Android Studio* possui um ou mais módulos que podem ser do aplicativo *Android*, de bibliotecas ou do *Google App Engine*, e é divido em pastas para proporcionar ao desenvolvedor acesso rápido aos principais arquivos e códigos fonte da aplicação. Cada módulo possui a pasta *manifests* que contém o arquivo AndroidManifest.xml, a pasta *java* onde ficam todos os códigos fonte na linguagem *Java*, e a pasta *res* onde estão todos os recursos que não são códigos, que ficam os *layouts XML* da aplicação, *strings* e imagens utilizadas no projeto. Como pode ser visto na Figura 4.

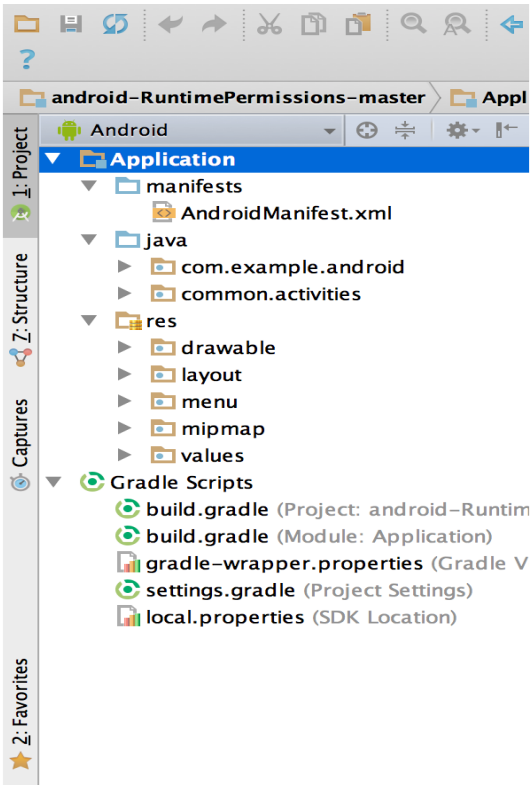

Figura 4 – Os Arquivos do Projeto na Visualização do Android
Disponível em: < https://developer.android.com/studio/intro/index.html >

2.4 SQLite

Os bancos de dados surgiram com a necessidade de armazenamento de informações para posterior acesso, levando em consideração que era um incômodo ter que inserir os dados que seriam usados no sistema toda vez que o mesmo fosse reiniciado. Tal motivo impôs a necessidade de utilização de uma ferramenta para persistência de informações, tornando obsoletos os sistemas que não utilizavam tais recursos.

Inicialmente os dados eram armazenados em sistemas de arquivos, tornando mais complexas várias tarefas como o gerenciamento dos dados, definição da localização e outras, já que a aplicação cliente era a responsável por tudo isso, o que acarretava perda de produtividade. Em solução a estes problemas surgiram os SGBD's, que são sistemas gerenciadores de bancos de dados responsáveis por fazer toda a manipulação do banco, retirando essa responsabilidade da aplicação cliente, simplificando alguns processos como os de inserção, exclusão e acesso aos dados.

> No contexto de aplicações para dispositivos móveis, a escassez de recursos computacionais, tais como memória, capacidade de armazenamento inerentes a dispositivos móveis pessoais (celulares e PDA's), combinado com a falta de suporte, faz com que os desenvolvedores para estes tipos de dispositivos não desfrutem dos benefícios de um SGBD (DEVMEDIA, 2016).

Em contrapartida a este problema, o sistema operacional *Android* oferece uma ferramenta de persistência de dados nativa a ele, o *SQLite* trata-se de um banco de dados escrito na linguagem C, de código aberto, responsável em fornecer uma poderosa biblioteca baseada em SQL, que funciona como um mini sistema gerenciador de banco de dados, tem apresentado popularidade entre os desenvolvedores de aplicações móveis, principalmente os que trabalham com *Android* e *iOS*.

Muitas aplicações usam o modelo cliente/servidor o que traz empecilhos ao seu uso, como dificuldade na instalação dos sistemas que o utilizam e obrigatoriedade de assistência completa pela empresa detentora do mesmo. Já o *SQLite* que é distribuído junto a aplicação torna o processo de instalação mais simples e rápido. Mesmo fornecendo uma poderosa biblioteca é um SGBD desenvolvido para atender as necessidades de dispositivos móveis e com isso se adequa às limitações apresentadas pelos mesmos, devido a esse fator não dispõe de todas funcionalidades oferecidas pelos grandes SGBD's.

2.5 Usabilidade

Em consequência à popularização dos *smartphones* o mercado de desenvolvimento de aplicações apresentou grande crescimento mesmo enfrentando dificuldades devido às limitações de tais dispositivos. Inicialmente os desenvolvedores procuravam assemelhar as ferramentas móveis o máximo possível das utilizadas em *desktops*, o que proporcionava ao usuário uma experiência dispendiosa e desagradável. Com isso foi constatado que este tipo de aplicação deveria receber um olhar diferente para que fosse tão aceitável quanto o próprio aparelho móvel.

O mercado de desenvolvimento de aplicações foi bombardeado com milhares de aplicativos, mas quantidade não implica em qualidade, e muitas destas ferramentas foram esquecidas pelos usuários por não oferecer experiência prazerosa em seu uso. Observando esse fato foi concluído que as mesmas deveriam seguir diretrizes para proporcionar manipulação satisfatória e agradável.

Existem vários tipos de testes, um deles é o de usabilidade, que consiste em uma técnica para avaliar a facilidade de uso das aplicações ao pedir para um grupo de usuário utilizar a aplicação e observar como estes se comportam em relação a elas (BALLARD, 2007).

Estes testes podem ser feitos de duas formas, em laboratório ou no campo. Os testes em laboratório utilizam emuladores ou dispositivos propriamente ditos, oferecem baixo custo, mas apresentam limitações, pois não levam em consideração variáveis importantes como a influência do ambiente externo. Já as experimentações feitas em campo, o usuário comum sofre influência do ambiente externo na utilização da ferramenta.

> Usabilidade é um dos componentes de aceitabilidade de um sistema e afirma que basicamente se refere à questão de se saber se o sistema é bom o suficiente para satisfazer todas as necessidades e exigências dos usuários. A usabilidade é dividida em cinco componentes: Facilidade de Aprendizagem, Facilidade de Memorização, Eficiência no Uso, Poucos Erros e Satisfação Subjetiva (NIELSEN, 1993).

Pode ser dito que as aplicações móveis possuem necessidades especiais em relação aos processos aos quais dão suporte. As atividades realizadas nos mesmos devem ser finalizadas de forma rápida, ou seja, o usuário deve atingir seu objetivo no menor tempo possível. O tempo de aprendizado da ferramenta deve ser curto, não exigindo grandes esforços para posterior uso.

3 APLICATIVO LOCASA

Nos últimos anos o ramo da tecnologia entrou na era *mobile*, onde aparelhos que executam aplicações móveis estão ganhando cada vez mais espaço no mercado. Fato este que é causado pela simplicidade fornecida aos usuários. Atualmente as informações possuem caráter dinâmico, as pessoas visam facilidade e rapidez em suas tarefas, como enviar e receber *e-mails* de qualquer lugar a qualquer momento.

A crescente popularização dos *smartphones* fez com que fossem inseridos na vida cotidiana das pessoas, de tal forma que é mais difícil encontrar uma pessoa que não utiliza os mesmos para auxiliar em seus afazeres diários do que o contrário. Isso se deve ao motivo de disporem de aplicações que auxiliam em suas tarefas, das mais simples às mais complexas.

Visando a inserção dos dispositivos móveis no dia a dia das pessoas, empresas de todos os ramos de negócio enxergaram a necessidade de fornecer esta facilidade aos seus clientes. Graças a isso, atualmente é possível realizar transferências bancárias ou mesmo fazer compras utilizando um aparelho móvel.

Assim como em todo campo de negócio, a gestão de imóveis possui vários sistemas de informação para seu auxílio, *softwares* que foram desenvolvidos para ser utilizados em computadores de mesa, fato que limita o locador a utilizá-los apenas onde a máquina está localizada. Deste modo, o desenvolvimento de um aplicativo que faça a gestão desta tarefa (desde cadastros ao controle dos aluguéis) se apresenta como uma ótima solução em tempos de era *mobile*.

3.1 Construção da Aplicação

O aplicativo LoCasa é descrito de forma detalhada nas seções a seguir, onde serão explanados a problemática a ser resolvida, recursos utilizados no desenvolvimento e testes realizados.

O sistema operacional *Android* foi avaliado para que a ferramenta de desenvolvimento fosse determinada a fim de proporcionar uma construção mais ágil da aplicação. O projeto foi desenvolvido no sistema operacional *Linux Ubuntu 15.10* de 64-bit, e a *framework* utilizada no projeto foi o *Android Studio* 2.1.2.

O *Android Studio* necessita de três *softwares* para que funcione corretamente e permita que os aplicativos sejam testados, são eles: JDK, JRE e SDK (responsável por fornecer as API's). O *Android Studio* fornecido para *Linux* não exige que o SDK seja instalado

separadamente, o que facilita sua configuração, quanto aos demais *softwares* o *download* e instalação devem ser realizados separadamente.

A ferramenta LoCasa fornece recursos para o armazenamento e manipulação de dados relevantes ao controle de aluguéis (inquilinos, imóveis e aluguéis). Para ser testada a eficiência da aplicação, foram realizados testes em dispositivos físicos com sistema operacional *Android*, que possuem diferentes poder de processamento e memória disponível, para que fosse constatado se a aplicação funciona normalmente em todos.

3.1.1 Requisitos do Sistema

Para que os requisitos fossem levantados de modo a dar suporte às reais necessidades dos possíveis usuários, entrevistas foram realizadas em busca das funcionalidades que auxiliariam de fato no controle do pagamento do aluguel de imóveis. Com base nas informações obtidas, foi possível traçar um roteiro de desenvolvimento da aplicação para que as dificuldades na gestão desta tarefa fossem transpostas de forma efetiva, e que proporcionasse facilidade nos processos envolvidos.

> Uma das primeiras fases de engenharia de um *software* consiste no Levantamento de Requisitos. Nesta etapa, o engenheiro de *software* busca compreender as necessidades do usuário e o que ele deseja que o sistema a ser desenvolvido realize. Isto é feito, sobretudo, por meio de entrevistas, nas quais o engenheiro tenta compreender como funciona hoje em dia o processo a ser informatizado e quais os serviços o cliente precisa que o *software* forneça (GILLEANES, 2007).

Requisitos são objetivos ou restrições estabelecidas por clientes e usuários do sistema que definem as diversas propriedades do sistema (JAIR, 2000). A fase de levantamento de requisitos é feita em conjunto com o cliente, para que o mesmo possa expressar suas necessidades em relação às funcionalidades do *software*.

De acordo com Sommerville (2003), a fase de levantamento e análise de requisitos é composta pelos seis passos a seguir:

- Compreensão do domínio: Os analistas devem desenvolver sua compreensão do domínio da aplicação;
- Coleta de requisitos: É o processo de interagir com os *stakeholders* do sistema para descobrir seus requisitos. A compreensão do domínio se desenvolve mais durante essa atividade;
- Classificação: Essa atividade considera o conjunto não estruturado dos requisitos e os organiza em grupos coerentes;

- Resolução de conflitos: Quando múltiplos *stakeholders* estão envolvidos, os requisitos apresentarão conflitos. Essa atividade tem por objetivo solucionar esses conflitos;
- Definição das prioridades: Em qualquer conjunto de requisitos, alguns serão mais importantes do que outros. Esse estágio envolve interação com os *stakeholders* para a definição dos requisitos mais importantes;
- Verificação de requisitos: Os requisitos são verificados para descobrir se estão completos e consistentes e se estão em concordância com o que os *stakeholders* desejam do sistema.

Após feita a análise das informações, os requisitos funcionais, não funcionais e regras de negócio foram avaliados para que pudessem ser inseridos no escopo do desenvolvimento da aplicação, e em seguida documentados. A seguir todos eles serão detalhados, de forma a entender mais facilmente o funcionamento do aplicativo LoCasa.

- Requisitos Funcionais

Os requisitos funcionais definem como o *software* deve funcionar mediante determinadas entradas. Na Quadro 1 são detalhados os requisitos funcionais do sistema a ser construído, contando com identificador, descrição e possíveis dependências que o mesmo venha a ter.

Quadro 1 - Requisitos Funcionais do Aplicativo LoCasa

Identificador	**Descrição**	**Dependente de**
RF01	Fornecer funções para armazenar dados referentes ao locador (cadastro, alteração e exclusão das informações).	RFN01.
RF02	Fornecer funções para armazenar dados referentes aos inquilinos (cadastro, alteração e exclusão das informações).	RF01, RF11, RFN01, RFN02.
RF03	Fornecer funções para armazenar dados referentes aos aluguéis (cadastro, alteração e exclusão das informações).	RF01, RF02, RF05, RF11, RN01, RN02, RN04, RFN01, RFN02.
RF04	Fornecer funções para armazenar dados referentes às listas de espera (cadastro, alteração e exclusão das informações).	RF01, RF02, RF05, RF11, RN05, RFN01, RFN02.
RF05	Fornecer funções para armazenar dados referentes aos imóveis (cadastro, alteração e exclusão das informações).	RF01, RF12, RFN01, RFN02.

Identificador	Descrição	Dependente de
RF06	Pesquisar aluguéis pagos/não pagos.	RF01, RF03, RF11, RFN03.
RF07	Fornecer relatório com informações relevantes (quantidade de pagos e pendentes) aos pagamentos de aluguéis.	RF01, RF11, RF03.
RF08	Alterar *status* do aluguel para defini-lo como pago.	RF01, RF11, RF03.
RF09	Alterar imóvel de inquilino caso seja desejado.	RF01, RF11, RF03, RN02.
RF10	Pesquisar lista de espera por imóvel.	RF01, RF02, RF04, RF05, RF11.
RF11	O sistema deve fornecer uma função de *login*, para que só o usuário autorizado tenha acesso às informações (locador).	RF01

- Requisitos Não Funcionais

Requisitos não funcionais de um sistema são as características implícitas a ele, estão relacionados aos níveis de desempenho, qualidade, robustez, segurança, dentre outras propriedades que o mesmo deve possuir. O Quadro 2 demonstra os requisitos não funcionais do aplicativo LoCasa.

Quadro 2 – Requisitos Não Funcionais do Aplicativo LoCasa

Identificador	Descrição	Categoria	Depende de
RFN01	Consistência e persistência dos dados, sendo esta feita por meio de um SGBD, padrão ou não.	Manutenibilidade.	
RFN02	Dados no sistema só podem ser alterados pelo administrador (locador).	Segurança de acesso.	RF01, RF11.
RFN03	Sigilo das informações armazenadas no *software*.	Segurança.	RF01, RF11.
RFN04	Suporte a dispositivos com diferentes tamanho de telas.	Portabilidade.	
RFN05	O *software* deve ser de fácil utilização, focando no conforto e facilidade no aprendizado de suas funcionalidades.	Usabilidade.	

- Regras de Negócio

As regras de negócio como o próprio nome informa, são as limitações impostas pelo meio comercial onde será utilizada a ferramenta proposta. O Quadro 3 mostra as regras que o aplicativo LoCasa deve seguir.

Quadro 3 – Regras de Negócio do Aplicativo LoCasa

Identificador	**Descrição**	**Depende de**
RN01	2 inquilinos não podem alugar o mesmo imóvel.	RF03, RFN01.
RN02	Inquilino inadimplente não pode solicitar mudança de imóvel.	RF09.
RN03	Inquilino já inicia contrato com aluguel pago (*status*: pago).	RF03.
RN04	As listas de espera devem ser divididas por imóvel.	RF04.
RN04	Após dia 25 de cada mês o software avisa ao usuário que o mês está acabando e orienta-o a verificar os aluguéis.	RF03.

Com base no levantamento de requisitos foi possível a elaboração de modelos que representam o funcionamento do *software* e quais funcionalidades o mesmo deve suprir.

3.1.2 Modelagem UML

Em toda área de atuação quando um projeto será desenvolvido são feitos vários modelos para que possa ser elaborado um roteiro correto de execução de atividades. Todo projeto, seja ele grandioso ou não, necessita de uma modelagem antes de ser iniciado.

A linguagem UML tornou-se, nos últimos anos, a linguagem padrão de modelagem de software adotada internacionalmente pela indústria de Engenharia de *Software* (GILLEANES, 2007).

A linguagem de modelagem unificada não se apresenta como uma linguagem de programação, é responsável por fornecer vários diagramas a serem utilizados para elaboração de modelos de determinado *software,* para que este possa ter sua lógica entendida de forma fácil e rápida. A seguir será apresentada toda a modelagem do aplicativo desenvolvido.

- Diagrama de Caso de Uso

O diagrama de caso de uso é responsável por demonstrar como o sistema vai reagir de acordo com as entradas de determinado usuário (ator). É responsável por apresentar as funcionalidades que o usuário manipulará ao utilizar o aplicativo.

A Figura 5, por meio do diagrama, tem por objetivo mostrar o funcionamento geral da aplicação. Os atores são as pessoas que irão utilizar o sistema de forma direta, o aplicativo LoCasa em questão possui um único ator, o locador (responsável por gerir a tarefa de controle).

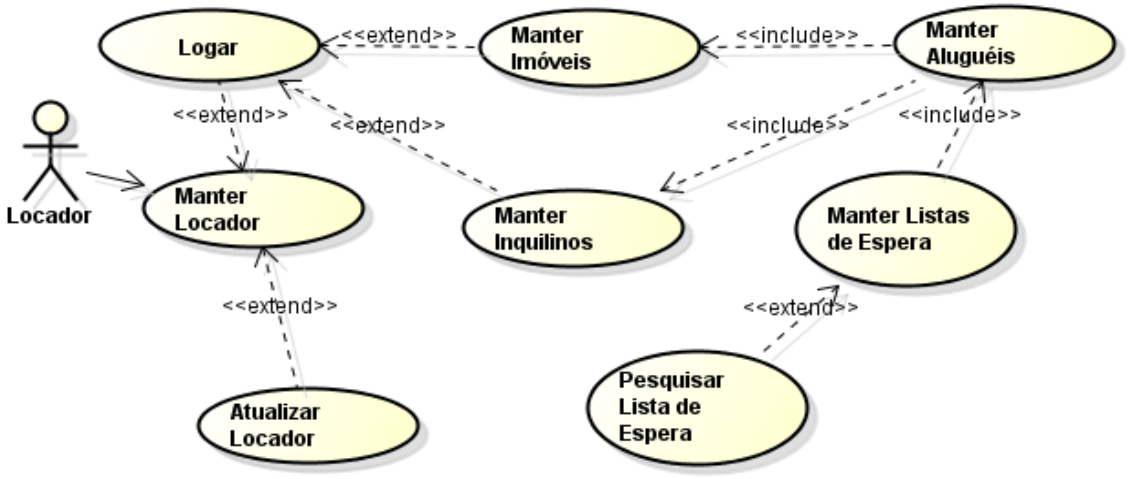

Figura 5 – Diagrama de Casos de Uso
Fonte: O autor (2016)

Os atores são representados pelos bonecos, como dito anteriormente, o aplicativo possui somente um (locador), a seta que o liga a determinado caso de uso demonstra vínculo entre eles. O locador será o administrador do sistema e responsável por inserir e manipular os dados. Para manter o controle de acesso às informações, o aplicativo dispõe de *login* que será detalhado a seguir, juntamente com todos os outros casos de uso. Os casos de uso também possuem relação entre si, o que é representado pelas setas pontilhadas e direcionadas entre os mesmos. A relação <<*extend*>> interliga dois casos de uso por meio da relação de extensão, isso acontece quando a realização de um caso de uso é extensível a outro, ou seja, quando o primeiro é executado o segundo pode ou não ser executado em seguida. Já a relação <<*include*>> define a obrigatoriedade de execução de um caso de uso quando outro for executado.

A documentação dos casos de uso apresentados na Figura 5 será demonstrada a seguir, com as informações relevantes para a execução dos mesmos. As funcionalidades do sistema serão detalhadas a seguir seguindo uma sequência de execução das tarefas na utilização da aplicação.

Para melhor entendimento dos casos de uso do sistema, a seguir os mesmos serão descritos detalhadamente.

Quadro 4 – Documentação do Caso de Uso Cadastro de Locador

Nome do Caso de Uso	**Cadastrar Locador**
Caso de Uso Geral	
Ator Principal	Locador
Ator Secundário	
Resumo	Determina quais dados serão necessários para que o cadastro do locador seja efetuado.
Pré-Condições	

Nome do Caso de Uso	**Cadastrar Locador**
Pós-Condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
	1. O sistema solicita os dados do locador que deseja se cadastrar;
2. Locador insere os dados;	
	3. Verifica se já existe locador cadastrado;
	4. O sistema insere o locador no banco de dados.
Restrições/Validações	O locador só é inserido no banco de dados caso não houver outro locador cadastrado. O sistema só pode ter um único administrador.

Após feito o cadastro do locador, o mesmo pode entrar de fato nas opções do aplicativo, ou atualizar seus dados quando quiser.

Quadro 5 – Documentação do Caso de Uso *Login*

Nome do Caso de Uso	***Login***
Caso de Uso Geral	
Ator Principal	Locador
Ator Secundário	
Resumo	O locador ao efetuar *login* terá acesso a todas as funcionalidades do aplicativo para inserção e manipulação dos dados.
Pré-Condições	Locador deve estar cadastrado.
Pós-Condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
	1. O sistema solicita senha e *login* do locador;
2. O locador insere sua senha e *login* fornecidos na hora do cadastro;	
	3. O sistema verifica se os dados correspondem aos armazenados no banco de dados;
	4. O sistema libera o acesso às funções ao locador.
Restrições/Validações	Se senha e/ou *login* não forem corretos o sistema não libera o acesso às funções.

Quadro 6 – Documentação do Caso de Uso Atualizar Locador

Nome do Caso de Uso	**Atualizar Locador**
Caso de Uso Geral	
Ator Principal	Locador
Ator Secundário	

Nome do Caso de Uso	**Atualizar Locador**
Resumo	Fornece opções para modificar os dados do locador que foram inseridos em seu cadastro.
Pré-Condições	Locador deve estar cadastrado.
Pós-Condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
	1. O sistema solicita senha e *login* do locador;
2. O locador insere sua senha e *login* fornecidos na hora do cadastro;	
	3. O sistema verifica se os dados correspondem aos armazenados no banco de dados;
	4. O sistema libera o formulário de cadastro de locador ao usuário.
Restrições/Validações	Se senha e/ou *login* não forem corretos o sistema não libera o acesso ao formulário.

Caso o locador escolha acessar as funções de inserção de informações no aplicativo ele irá se deparar com todas as funções listadas na tela de seu aparelho.

Quadro 7 – Documentação do Caso de Uso Manter Imóveis

Nome do Caso de Uso	**Manter Imóveis**
Caso de Uso Geral	
Ator Principal	Locador
Ator Secundário	
Resumo	Descreve quais são os dados necessários para que os imóveis sejam armazenados.
Pré-Condições	Locador deve estar *logado* no sistema.
Pós-Condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
	1. O sistema solicita os dados referentes ao imóvel a ser cadastrado;
2. O locador insere os dados;	
	3. O sistema verifica se há campos vazios ou se o imóvel já existe;
	4. O sistema insere o imóvel no banco de dados.
Restrições/Validações	Caso o imóvel já exista ele não será inserido no banco novamente.

Quadro 8 – Documentação do Caso de Uso Manter Inquilinos

Nome do Caso de Uso	**Manter Inquilinos**
Caso de Uso Geral	
Ator Principal	Locador

Nome do Caso de Uso	**Manter Inquilinos**
Ator Secundário	
Resumo	Descreve quais são os dados necessários para que os inquilinos sejam armazenados.
Pré-Condições	Locador deve estar *logado* no sistema.
Pós-Condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
	1. O sistema solicita os dados referentes ao inquilino a ser cadastrado;
2. O locador insere os dados;	
	3. O sistema verifica se há campos vazios ou se o inquilino já existe;
	4. O sistema insere o inquilino no banco de dados.
Restrições/Validações	Caso o inquilino já exista ele não será inserido no banco novamente.

Quadro 9 – Documentação do Caso de Uso Manter Aluguéis

Nome do Caso de Uso	**Manter Aluguéis**
Caso de Uso Geral	
Ator Principal	Locador
Ator Secundário	
Resumo	Descreve quais são os dados necessários para que os aluguéis sejam armazenados.
Pré-Condições	Locador deve estar *logado* no sistema.
Pós-Condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
	1. O sistema solicita os dados referentes ao aluguel a ser cadastrado;
2. O locador insere os dados;	
	3. O sistema verifica se há campos vazios ou se aquele determinado imóvel já está alugado, ou se aquele inquilino já aluga outro imóvel;
	4. O sistema insere o aluguel no banco de dados.
Restrições/Validações	Caso o imóvel e/ou o inquilino não existam, ou o imóvel já está alugado, ou o inquilino já possui um aluguel em seu nome o aluguel não é inserido no banco de dados.

Quadro 10 - Documentação do Caso de Uso Manter Lista de Espera

Nome do Caso de Uso	**Pesquisar Lista de Espera**
Caso de Uso Geral	
Ator Principal	Locador
Ator Secundário	
Resumo	Descreve quais são os dados necessários para que as listas de espera sejam armazenadas.
Pré-Condições	Locador deve estar *logado* no sistema.
Pós-Condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
	1. O sistema solicita os dados referentes à lista de espera a ser cadastrada;
2. O locador insere os dados;	
	3. O sistema verifica se há campos vazios, se o imóvel fornecido está alugado, se o inquilino fornecido existe e se já está em outra lista de espera;
	4. O sistema insere a lista de espera no banco de dados.
Restrições/Validações	Se o imóvel não estiver alugado ou inquilino e/ou imóvel não existirem o sistema informa e não insere a lista de espera ao banco de dados.

Quadro 11 – Documentação do Caso de Uso Pesquisar Lista de Espera

Nome do Caso de Uso	**Pesquisar Lista de Espera**
Caso de Uso Geral	
Ator Principal	Locador
Ator Secundário	
Resumo	Fornece ao locador a função de procurar pelo código do imóvel se o mesmo possui inquilino em lista de espera.
Pré-Condições	Locador deve estar *logado* no sistema.
Pós-Condições	
Fluxo Principal	
Ações do Ator	**Ações do Sistema**
	1. O sistema solicita o código do imóvel para que inicie a busca;
2. O locador insere os dados;	
	3. O sistema busca no banco de dados por listas de espera que constem com determinado código de imóvel;
	4. O sistema retorna quantas pessoas estão na lista de espera daquele imóvel e imprime as informações da lista.
Restrições/Validações	Caso o imóvel não possua lista de espera nada é visualizado.

Após ter sido feita a análise de quais processos o *software* deve dar suporte, ou seja, funções que o mesmo deve possuir para que o objetivo final fosse alcançado, a modelagem do banco de dados pôde ser feita.

3.1.3 Diagrama de Entidade e Relacionamento

O Modelo Entidade e Relacionamento (também chamado ER, ou simplesmente MER), como o nome sugere, é um modelo conceitual utilizado na Engenharia de *Software* para descrever os objetos (entidades) envolvidos em um domínio de negócios, com suas características (atributos) e como elas se relacionam entre si (relacionamentos) (DEVMEDIA, 2016).

Este diagrama permite que se faça o modelo da estrutura que o banco de dados deve possuir, com isso é possível inferir as relações que existirão entre as entidades bem como o funcionamento do banco. As entidades são representadas pelos retângulos, ao seu redor, ligadas a elas por retas com um círculo na extremidade, estão seus atributos. As relações entre as entidades são representadas pelos losangos que as interligam.

O aplicativo LoCasa apresenta cinco entidades relacionadas entre si, como pode ser visto na Figura 6. A entidade “Locador” é responsável pelo armazenamento das informações do usuário que irá administrar o sistema, é a única pessoa que possui acesso às informações armazenadas no aplicativo.

A entidade “Aluguel” que se relaciona com as entidades “Locador”, “Imóvel” e “Inquilino”, possui como atributos as informações que fazem a ligação entre um “Inquilino” e um “Imóvel”. As entidades “Inquilino” e “Imóvel” são as tabelas responsáveis por armazenar informações sobre as pessoas que solicitam um aluguel e sobre os imóveis disponíveis, respectivamente. Por fim, se relacionando com “Inquilino” tem-se a entidade “Lista de Espera” que é solicitada pelo inquilino.

O diagrama abaixo foi construído utilizando uma ferramenta denominada *brModelo*, que fornece ao usuário todos os recursos necessários à criação de um DER completo.

Com a modelagem do banco de dados construída, o mesmo pôde ser implementado, o *SQLite* foi escolhido como a ferramenta de persistência dos dados do *app* (levando em consideração as limitações que o mesmo apresenta).

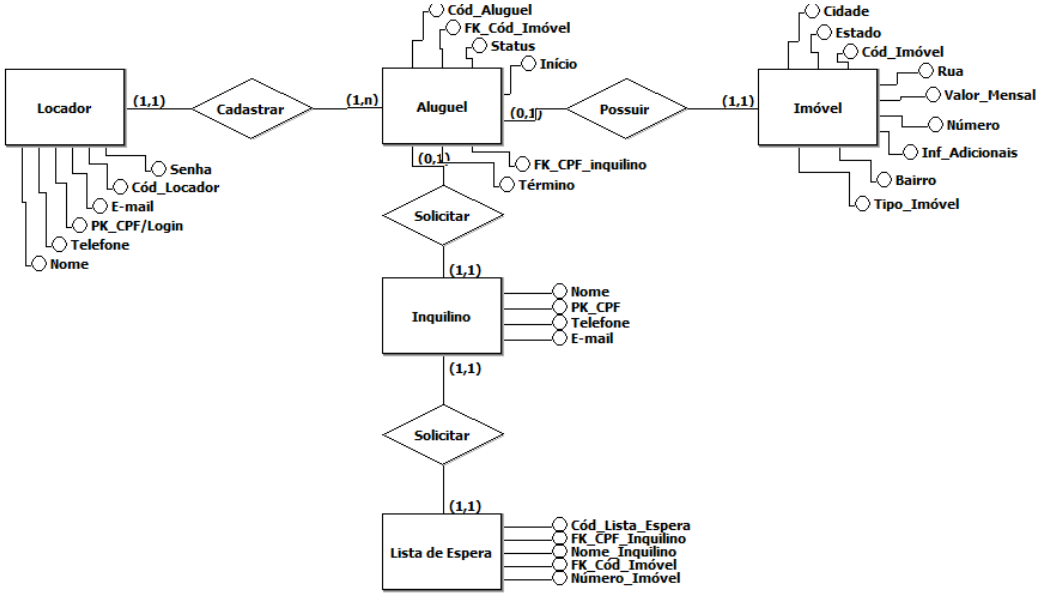

Figura 6 – Diagrama Entidade e Relacionamento
Fonte: O autor (2016)

3.1.4 Criação do Banco de Dados

Após feito o levantamento de requisitos, o desenvolvimento do aplicativo tornou-se mais objetivo também menos propenso a erros, e sua construção foi iniciada. Como foi escolhido o banco de dados padrão fornecido pelo *Android*, sua implementação foi feita no próprio *Android Studio*, utilizando duas classes em sua construção, e para que os dados pudessem ser acessados foram utilizadas as classes "Repositório", como pode ser visto na Figura 7.

Cada classe utilizada na aplicação que necessite em algum momento armazenar ou recuperar informações contidas no banco de dados possui sua respectiva classe repositório. Estas classes são responsáveis em armazenar, atualizar ou excluir todas as informações existente no banco do aplicativo.

As classes dos pacotes "*Database*" e "Domínio" em conjunto, são responsáveis por toda construção e manipulação do banco de dados.

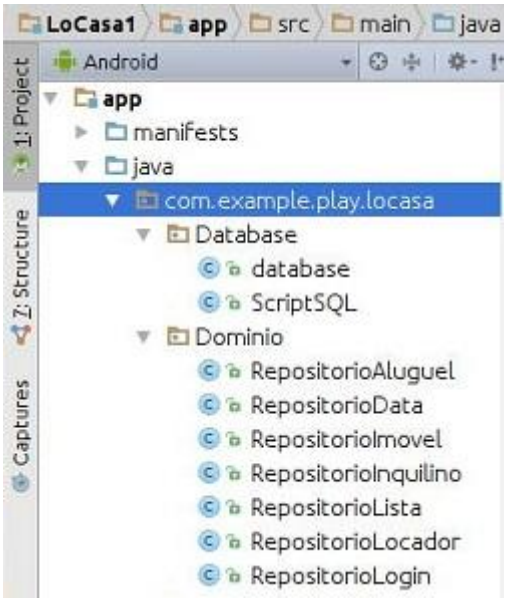

Figura 7 – Classes Utilizadas na Construção e Manipulação do Banco de Dados
Fonte: O autor (2016)

A classe "*ScriptSQL*" é responsável pela codificação do banco em linguagem SQL, possui todo *script* utilizado para a criação das tabelas (entidades) que o aplicativo possui. Como pode ser visualizado na Figura 8, cada tabela é construída por meio de uma função encarregada em unir todas *strings* em uma única (por meio do recurso *sqlbuilder.append*) e retorná-la quando a função for chamada.

```
public class ScriptSQL {

    public static String getCreateImoveis(){

        StringBuilder sqlbuilder = new StringBuilder();

        sqlbuilder.append("CREATE TABLE IF NOT EXISTS IMOVEL( ");
        sqlbuilder.append("_CODIGO VARCHAR(5) NOT NULL PRIMARY KEY, ");
        sqlbuilder.append("CIDADE VARCHAR(20), ");
        sqlbuilder.append("ESTADO VARCHAR(2), ");
        sqlbuilder.append("RUA VARCHAR(30), ");
        sqlbuilder.append("NUMERO VARCHAR(7), ");
        sqlbuilder.append("BAIRRO VARCHAR(30), ");
        sqlbuilder.append("TIPO VARCHAR(15), ");
        sqlbuilder.append("SACADA VARCHAR(3), ");
        sqlbuilder.append("VALOR VARCHAR(10) ");
        sqlbuilder.append("); ");

        return sqlbuilder.toString();

    }
```

Figura 8 – Modelo de Função Para Criação de Determinada Tabela
Fonte: O autor (2016)

As funções definidas em "*ScriptSQL*" são chamadas na classe "*Database*" que é responsável em manipular os comandos SQL propriamente ditos. Até então o banco de dados não foi criado, isso só acontece quando o aplicativo executa a tela principal pela primeira vez (*activity* principal).

O *SQLite* por não ser um SGBD poderoso apresenta limitações em relação à linguagem SQL, uma delas é não dar suporte total ao uso de chaves estrangeiras, necessárias na implementação do aplicativo. Para transpor essa barreira a checagem de chave estrangeira é feita de forma subentendida, ou seja, na entidade que possui a chave estrangeira é adicionado um campo para armazená-la, e a informação adicionada a este campo é checada em sua respectiva tabela para manter a consistência das informações.

3.1.5 Implementação do Aplicativo

A ferramenta *Android Studio* possui uma organização padrão dos projetos criados (Figura 4), no início fica situado o arquivo "*AndroidManifest.xml*" responsável por apresentar informações importantes para execução do código do projeto. Logo abaixo encontram-se as classes *Java* que possuem toda a codificação nesta linguagem, onde são manipuladas as informações inseridas pelo usuário. Em seguida, está a pasta "*res*" que possui tudo o que é utilizado no *layout* do aplicativo, parte com a qual o usuário vai lidar diretamente. E por último, "*Gradle Scripts"* encarregado de fornecer recursos para compilação do aplicativo.

Após construído, o aplicativo LoCasa possui as classes *Java* expostas na Figura 9, que são responsáveis por implementar todas as regras de negócio da aplicação, bem como a manipulação dos dados inseridos em todas as telas.

Figura 9 – Classes *Java* do Aplicativo LoCasa

Fonte: O autor (2016)

Logo abaixo encontra-se a pasta "*res*" que possui todas as telas da aplicação, cada tela é escrita em linguagem XML, o *Android Studio* apresenta recursos para proporcionar manipulação rápida e fácil do *design* do aplicativo. Todo arquivo XML possui sua respectiva classe em *Java*.

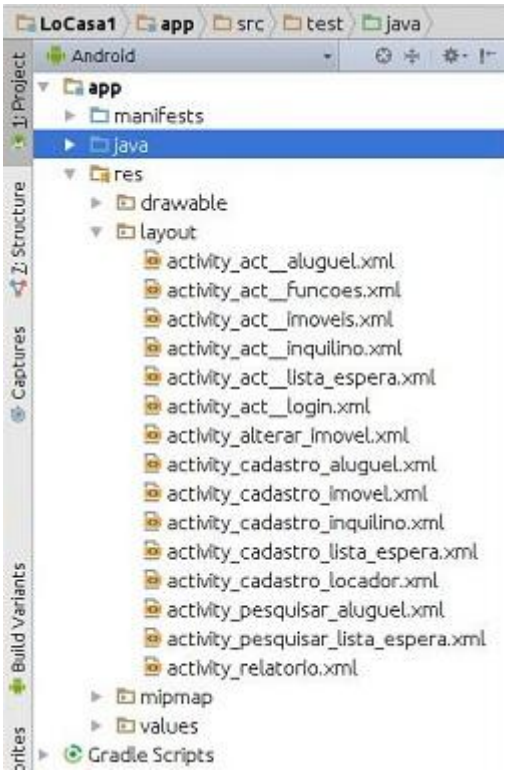

Figura 10 – Pasta "*res*" do Aplicativo LoCasa
Fonte: O autor (2016)

4 FUNCIONAMENTO DO APLICATIVO

O aplicativo LoCasa possuirá somente um usuário, responsável pelo cadastro de todos os dados, bem como acesso e recuperação de tais informações. Ao utilizar o aplicativo pela primeira vez o usuário encontra a tela de boas vindas representada pela Figura 11, que dispõe dos botões “Entrar” responsável por verificar se *login* e senha digitados estão corretos para que o locador seja redirecionado à tela das demais funções do *app*, e o botão “REALIZAR CADASTRO” que direciona o locador à respectiva tela de seu cadastro, para que logo após faça seu *login*.

Figura 11 – Tela Inicial da Aplicação sem Locador Cadastrado
Fonte: O autor (2016)

Ao utilizar o botão “REALIZAR CADASTRO”, o usuário será direcionado a tela de cadastro, representada pela Figura 12, que apresenta três botões referentes à manipulação dos dados do locador.

O usuário precisa preencher impreterivelmente todos os campos, caso algum deles fique vazio o sistema não efetuará o cadastro, e irá emitir uma mensagem informando ao usuário da existência de espaços em branco.

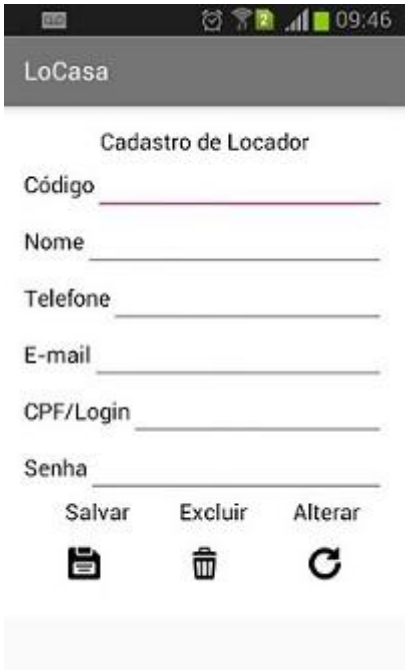

Figura 12 – Tela de Cadastro do Locador
Fonte: O autor (2016)

Após realizado seu cadastro, quando o locador utilizar o aplicativo outra vez, a tela inicial será a representada pela Figura 13, não apresentando mais o botão "REALIZAR CADASTRO", e sim o botão "ATUALIZAR LOCADOR" que da mesma forma do botão "Entrar", verifica se *login* e senha digitados estão corretos para que o locador possa ser redirecionado a tela representada pela Figura 13.

Figura 13 – Tela Inicial da Aplicação com Locador Cadastrado
Fonte: O autor (2016)

Com cadastro feito, o locador pode inserir seus dados e entrar na aplicação, ao efetuar *login* é direcionado à tela que dispõe de todas as funções oferecidas pelo aplicativo, representada pela Figura 14.

Figura 14 – Tela Funções Oferecidas pelo Aplicativo
Fonte: O autor (2016)

A tela acima apresenta ícones representativos das funções às quais dão suporte, ao efetuar um *click* sobre os ícones de “Imóveis”, “Inquilinos”, “Aluguéis” e “Lista de Espera” o locador é direcionado às telas de listagem e busca de itens já inseridos no banco de dados, representados pelas Figuras 15, 16, 17 e 18, respectivamente.

Smartphones com *displays* menores, a princípio, sofrem com a dificuldade de acesso às funcionalidades inferiores, principalmente em telas que possuem muitas funcionalidades na vertical, algumas chegam a não ser visualizadas pelos usuários.

Fato que foi resolvido adicionando uma simples *tag* ao arquivo XML da respectiva tela que deseja adicionar a barra de rolagem, para que o usuário possa dirigir-se às funções inferiores sem problemas.

Figura 15 – Tela Listagem e Busca de Imóveis
Fonte: O autor (2016)

Figura 16 – Tela Listagem e Busca de Inquilinos
Fonte: O autor (2016)

Figura 17 – Tela Listagem e Busca de Aluguéis
Fonte: O autor (2016)

Figura 18 – Tela de Listagem e Busca de Listas de Espera
Fonte: O autor (2016)

Ao selecionar a opção de Aluguéis, quando a tela da Figura 17 é iniciada, como o vencimento do aluguel é ao final do mês, o sistema verifica se há aluguéis cadastrados e se o mês está próximo ao fim, se estas duas verificações forem confirmadas o sistema emite uma mensagem na tela alertando o locador e orientando-o a verificar os aluguéis pendentes.

Figura 19 – Tela Alerta ao Locador Sobre o Fim do Mês
Fonte: O autor (2016)

As Figuras 15, 16, 17 e 18 apresentam o mesmo conjunto de botões com as mesmas funcionalidades, o ícone "+" situado no canto superior direito direciona o usuário à tela de cadastro da referida opção escolhida, como é representado pelas Figuras 20, 21, 22 e 23 respectivamente, possibilitando ao mesmo cadastrar novos dados. O botão representado pelo ícone de lupa, fornece a opção de buscar um imóvel, inquilino, aluguel ou lista de espera pelo seu código, ao digitar o código de tal dado e efetuar um *click* sobre o botão de busca, a informação será recuperada do banco de dados e impressa na tela. Por fim, o botão "Listar Todos" que como o próprio nome sugere, é responsável por recuperar informações de todos os dados que estão armazenados no banco e os imprimir na tela. Outra funcionalidade agregada a estas telas, é a possibilidade de *click* sobre qualquer item impresso na listagem fazendo com que o mesmo seja enviado a tela de cadastro com seus dados preenchidos nos respectivos campos, o que facilita a alteração de informações e dificulta uma eventual exclusão acidental de dados.

Figura 20 - Tela Cadastro de Imóveis
Fonte: O autor (2016)

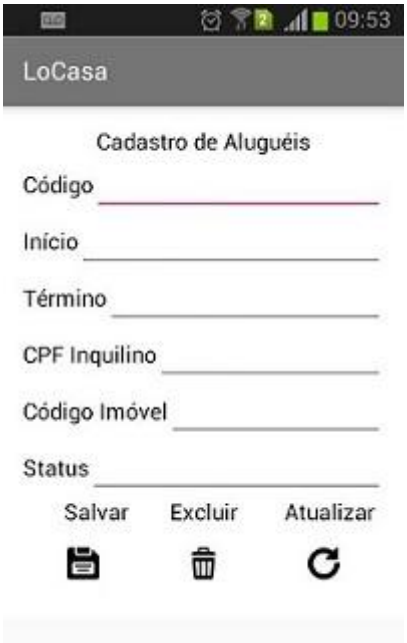

Figura 22 – Tela Cadastro de Aluguéis
Fonte: O autor (2016)

Figura 21 - Tela Cadastro de Inquilinos
Fonte: O autor (2016)

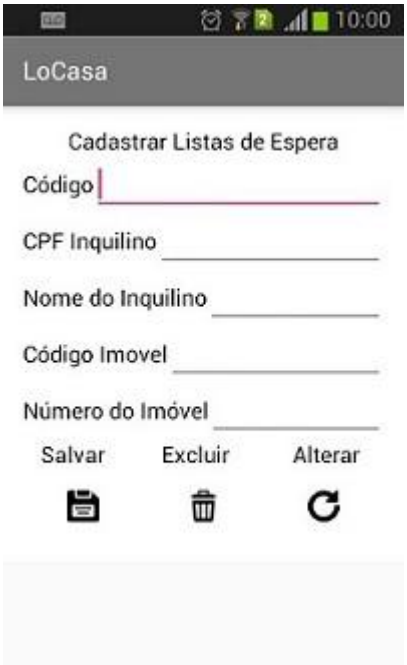

Figura 23 - Tela Cadastro de Lista de Espera
Fonte: O autor (2016)

O sistema não aceita campos em branco e código/CPF repetidos, cada processo é efetuado com rapidez e comprovado com mensagem de confirmação. Ao tentar excluir ou alterar dados, o sistema exibe uma janela de confirmação ao usuário, exigindo que ele comprove a intenção de atualização de informações.

O aplicativo também fornece ao usuário a opção de pesquisar por aluguéis de acordo com seus *status* (“Pesquisar Aluguel”), como pode ser visualizado na Figura 24, o locador insere a situação dos aluguéis que procura e ao efetuar um *click* sobre o ícone de busca, o sistema irá recuperá-los e lista-los na tela.

Figura 24 – Tela de Busca por Aluguéis Pagos ou Pendentes
Fonte: O autor (2016)

Muitos dos *softwares* para gestão de imóveis não contam com funções de lista de espera, o aplicativo LoCasa além de dar suporte a esta, ainda permite que o usuário procure se determinado imóvel possui uma fila de espera (“Lista de Espera por Imóvel”), de forma análoga à busca de aluguéis pelo *status*, a busca da fila de um determinado imóvel é feita pelo seu código, como pode ser visto na Figura 25.

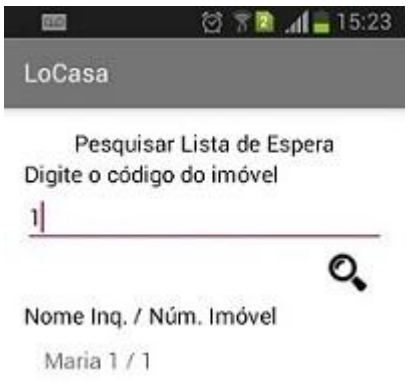

Figura 25 – Tela de Pesquisar Lista de Espera por Imóvel
Fonte: O autor (2016)

O usuário também pode fazer a troca de imóvel do aluguel de determinado inquilino caso este venha a solicitar (“Mudar Inquilino de Imóvel”), para que isso seja feito, na tela representada pela Figura 26 são digitados o código do imóvel atual do aluguel e o código do imóvel ao qual vai mudar, para que a modificação seja feita o inquilino não pode estar

inadimplente, caso esteja o sistema informa que a mudança não foi efetuada devido sua situação.

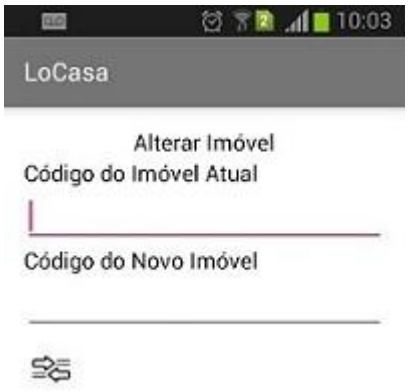

Figura 26 – Tela de Alteração de Imóvel de Determinado Aluguel
Fonte: O autor (2016)

E em desfecho às funcionalidade do aplicativo, tem-se o "Relatório" que possibilita ao usuário consultar a quantidade de aluguéis pagos e pendentes, conforme Figura 27, até o momento.

Figura 27 – Tela Relatório
Fonte: O autor (2016)

O principal objetivo do desenvolvimento de aplicações móveis é simplificar ao máximo qualquer tarefa, tornando seus processos mais ágeis e mantendo sua solidez. O aplicativo LoCasa é uma aplicação móvel que supre todas as necessidades do controle do aluguel de imóveis, fornecendo alguns recursos que vão além em relação aos oferecidos pelos *softwares* convencionais (*desktop*), provendo ao usuário a possibilidade de manipular seus dados a qualquer hora e lugar.

5 TESTES

O aplicativo LoCasa foi testado em quatro dispositivos móveis diferentes, para que seu funcionamento fosse analisado perante distintas disposições de recursos. O Quadro 12 a seguir detalha as configurações dos aparelhos utilizados.

Quadro 12 – Dispositivos Físicos Utilizados nos Testes

Modelo	**Versão do SO**	**Processador**	**Memória RAM**
Samsung Galaxy Ace	2.3.6	832MHz	290MB
Samsung Galaxy Win	4.1.2	1.2GHz	845MB
Lenovo Vibe K10	5.1.1	1.5GHz Octa Core	2GB
Motorola Moto G 2°	6.0	1.2 GHz Quad Core	1GB

O aplicativo não apresenta limitações geradas por diferentes tamanhos de tela. A mínima versão do *Android* necessária para execução da ferramenta é 2.1, abrangendo mais de 95% dos dispositivos ativos. A versão desenvolvida até o momento possui 4.23MB, espaço fornecido até pelos dispositivos mais simples.

5.1 Testes em Campo do Aplicativo LoCasa

Com o objetivo de obter impressões de usuários reais em relação à usabilidade do aplicativo, o mesmo foi testado por dois possíveis clientes futuros que possuem experiência no ramo de locação de imóveis. Para que conclusões fossem tiradas dos testes dois questionários foram respondidos pelos usuários com seus respectivos pontos de vista em relação à usabilidade do aplicativo LoCasa. Um dos questionários foi elaborado pelo desenvolvedor do projeto, a fim de obter informações sobre a presença de aplicações voltadas à locação de imóveis e sobre o aplicativo apresentado (Apêndice A).

O segundo questionário foi elaborado com base no trabalho apresentado por Rafael (2013), onde é mostrado que existem diversos tipos de questionários padronizados utilizados na obtenção de respostas sobre a usabilidade de *softwares* baseadas no pós-teste dos mesmos. Em contraponto, nenhum dos questionários era voltado especificamente para aplicações móveis (com *smartphones touchscreen)*, assim sendo, propôs um questionamento padrão a ser utilizado na inferência de resultados específicos relacionados às aplicações móveis, devido ao fato de o questionário apresentar perguntas repetitivas foram selecionadas somente doze (Apêndice B).

Após feita avaliação sobre as repostas colhidas nos questionários, pôde-se concluir que ainda existe uma carência no que diz respeito ao desenvolvimento de aplicações móveis direcionadas à administração de imóveis. Os testes em campo foram satisfatórios, o aplicativo mostrou-se funcional atendendo às necessidades dos usuários bem como a mobilidade desejada pelos mesmos.

6 CONCLUSÕES FINAIS

O aplicativo LoCasa foi construído com o intuito de auxiliar locadores de imóveis na gestão do pagamento de aluguéis, permitindo-lhes manter o cadastro de imóveis, inquilinos, aluguéis e listas de espera, bem como a manipulação de tais informações para posterior acesso e/ou modificação. Fornecendo ao usuário acesso rápido e fácil aos dados, proporcionado pela utilização de um aplicativo.

Em todo o documento foram apresentados os recursos utilizados na construção do *app*, desde a escolha do sistema operacional móvel utilizado, ferramenta de construção e recursos para persistência das informações. Para que as reais necessidades dos possíveis usuários fossem atendidas, os diagramas de caso de uso e entidade relacionamento foram elaborados, a fim de que fosse entendido o funcionamento completo do sistema, para que o mesmo pudesse ser construído.

O aplicativo encontra-se em versão funcional, podendo ser utilizado no controle do pagamento de aluguéis, entretanto, está passível a atualizações visando melhorias no *design* para torná-lo mais agradável e implantação de novas funcionalidades.

6.1 Trabalhos Futuros

Como trabalhos futuros propõe-se:

- Implementação de funcionalidade que permita ao inquilino a escolha da data de vencimento de seu aluguel;
- Aperfeiçoamento do *layout*;
- Agregar o *Google Drive* à aplicação para que os dados fiquem salvos na nuvem.

Como mencionado em sessões anteriores, o aplicativo está passível a atualizações. Visando melhoria de sua usabilidade, bem como evolução no *layout* a ser apresentado aos usuários, e aperfeiçoamentos no que diz respeito ao armazenamento e segurança das informações contidas no *software*.

REFERÊNCIAS

ANDROID DEVELOPERS. **The developer's guide.** Disponível em: <http://developer.android.com/guide/index.html>. Acesso em: 20 de out. de 2016.

APPLE. **Apple Developer.** Disponível em: <http://developer.apple.com>. Acesso em: 10 de nov. de 2016.

BALLARD, B. **Projetando a experiência do usuário móvel.** 2007.

BLACKBERRY – RIM – RESEARCH IN MOTION. Disponível em: < http://br.blackberry.com/software/smartphones/blackberry-10-os.html?LID=br:bb:software:smartphonesoftware:blackberry10os&LPOS=br: bb:software />. Acesso em: 25 de nov. de 2016.

BRAHLER, S. **Analysis of the Architecture.** 2010.

BROCKSCHMIDT, K. **Microsoft Press Programming Windows 8 Apps with HTML, CSS, and JavaScript.** Microsoft Corporation.

DEVMEDIA. **Modelo Entidade Relacionamento (MER) e Diagrama Entidade-Relacionamento (DER).** Disponível em: <http://www.devmedia.com.br/modelo-entidade-relacionamento-mer-e-diagrama-entidade-relacionamento-der/14332>. Acesso em: 30 de nov. de 2016.

DEVMEDIA. **SQLite no Android.** Disponível em: <http://www.devmedia.com.br/sqlite-no-android/19201>. Acesso em: 09 de nov. de 2016.

FLING, B. **Mobile design and development.** Páginas 01-12. 2009.

GARTNER. **Gartner diz que as vendas mundiais de celulares caíram 2% no primeiro trimestre de 2012.** Disponível em: <http://www.gartner.com/it/page.jsp?id=2017015>. Acesso em: 06 de dez. 2016.

GUEDES, G. T. A. **UML - Uma Abordagem Prática.** 3° Edição, São Paulo: Editora Novatec, 2007.

HUBSCH, E. **Uma abordagem corporativa do desenvolvimento de aplicações para dispositivos móveis.** Disponível em: <www.fatecsp.br_dti_tcc_tcc00065>. Acesso em: 08 de nov. de 2016.

KAHNEY, L. **A cabeça de Steve Jobs.** Tradução Maria Helena Lyra. Rio de Janeiro, 2009.

LECHETA, R. R. **Google Android – Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** 1° Edição, São Paulo: Editora Novatec, 2009.

LECHETA, R. R. **Google Android – Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** 2° Edição, São Paulo: Editora Novatec, 2010.

LEITE, J. C. **Análise e Especificação de Requisitos.** Disponível em: <https://www.dimap.ufrn.br/~jair/ES/c4.html>. Acesso em: 21 de nov. de 2016.

NIELSEN, J. **Engenharia de Usabilidade.** 1993.

OLIVEIRA, R. J. **Proposta de um questionário pós-teste para medir usabilidade de aplicativos de celulares *touchscreen.*** Universidade Federal de Santa Catarina. Florianópolis, 2013.

OPENHANDSETALLIANCE. Disponível em: <http://www.openhandsetalliance.com/oha overview.html>. Acesso em: 01 de dez. de 2016.

SOMERVILLE, I. **Engenharia de Software.** Tradução Maurício de Andrade. 6° Edição, São Paulo: Editora Addison-Wesley, 2003.

ZEMAN, E. **Google tem 1,5 milhão de Androids ativados por dia.** Disponível em: <http://www.itforum365.com.br/conectividade/dispositivos/google-tem-15-milhao-de-androids-ativados-por-dia>. Acesso em: 03 de dez. de 2016.

APÊNDICES

APÊNDICE A – Questionário Elaborado pelo Desenvolvedor do Projeto

Perguntas	Usuário 1	Usuário 2
1. Trabalha ou trabalhou na área há quanto tempo?	Trabalha, há mais de 2 (dois) anos.	Trabalhou, por 1 (um) ano e meio.
2. Utiliza algum *software* em auxílio a tarefa?	Não.	Não.
3. Já viu algum aplicativo com este propósito?	Não.	Não.
4. Já utilizou algum aplicativo para auxiliar esta tarefa?	Não.	Não.
5. Qual seria a vantagem da utilização de uma aplicação móvel?	Acessibilidade e organização.	Mobilidade, acessar dados dos clientes de qualquer lugar.
6. O que acha da implementação de um aplicativo para controle de aluguéis?	Boa ideia, pois facilita no atendimento ao cliente.	Ótima idéia.
7. Que funcionalidade do aplicativo apresentado lhe chamou mais atenção?	Cadastro de clientes.	Lista de Espera.
8. O aplicativo LoCasa atende às necessidades?	Sim, pois fornece o suporte necessário.	Sim.

APÊNDICE B – Questionário Padrão Voltado às Aplicações Móveis

Perguntas	Usuário 1	Usuário 2
1. Foi fácil inserir dados no aplicativo?	Sim.	Sim.
2. Foi fácil aprender usar o aplicativo?	Sim.	Sim.
3. Foi fácil navegar no aplicativo?	Sim.	Sim.
4. Usaria o aplicativo com frequência?	Sim.	Sim.
5. O aplicativo impõe dificuldade em seu uso?	Não.	Não.
6. Se sentiu no comando do aplicativo?	Sim.	Sim.
7. As tarefas executadas no aplicativo necessitam de muito tempo?	Não.	Não.
8. É fácil corrigir algum dado incorreto inserido?	Sim.	Sim.
9. Se sentiu frustrado ao usar o aplicativo?	Não.	Não.
10. Recomendaria o aplicativo a outras pessoas?	Sim.	Sim.
11. Gostou de usar o aplicativo?	Sim.	Sim.
12. Alguma sugestão ao desenvolvedor?	Disponibilizar mais recursos.	Funcionalidade para emissão de boletos.

TERMO DE AUTORIZAÇÃO PARA PUBLICAÇÃO DIGITAL NA BIBLIOTECA "JOSÉ ALBANO DE MACEDO"

Identificação do Tipo de Documento

( ) Tese
( ) Dissertação
(X) Monografia
( ) Artigo

Eu, **Gabriel de Lima Leal**, autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação **LoCasa: Um *app* para controle do pagamento de aluguéis** de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 31 de Janeiro de 2017.

Assinatura