UNIVERSIDADE DO EXTREMO SUL CATARINENSE – UNESC

CURSO DE CIÊNCIA DA COMPUTAÇÃO

RAFAEL SILVA DE BITENCOURT

SOFTWARE ANDROID PARA CORRETORES DE IMÓVEIS

CRICIÚMA

2013

RAFAEL SILVA DE BITENCOURT

SOFTWARE ANDROID PARA CORRETORES DE IMÓVEIS

Trabalho de Conclusão de Curso, apresentado para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC.

Orientador: Prof. Esp. Gilberto Vieira da Silva

CRICIÚMA

2013

RAFAEL SILVA DE BITENCOURT

SOFTWARE ANDROID PARA CORRETORES DE IMÓVEIS

Trabalho de Conclusão de Curso aprovado pela Banca Examinadora para obtenção do grau de Bacharel no curso de Ciência da Computação da Universidade do Extremo Sul Catarinense, UNESC, com Linha de Pesquisa em Computação Móvel.

Criciúma, 25 de novembro de 2013.

BANCA EXAMINADORA

Prof. Gilberto Vieira da Silva – Esp. – (UNESC) – Orientador

Prof. Fabricio Giordani – Esp. – (UNESC)

Prof. Gustavo Bisognin – MSc. – (UNESC)

Dedico este trabalho a minha mãe, que foi quem mais me incentivou e apoiou.

AGRADECIMENTOS

Agradeço primeiramente aos meus pais João Batista e Ester, por terem me criado com educação e apoiado durante todo este curso, sempre incentivando e torcendo pelo meu crescimento pessoal e profissional.

A minha namorada Camila, que conheci no início do curso e vem me acompanhando e apoiando desde então, sendo paciente e confortando quando mais precisei.

Ao meu orientador Gilberto e a professora Merisandra, por me auxiliarem no desenvolvimento do trabalho.

E a todos os amigos que de alguma forma me ajudaram, sendo diretamente no curso ou, até mesmo, com momentos de descontração neste período.

RESUMO

As empresas de tecnologia, de um modo geral, estão buscando cada vez mais a mobilidade, a fim de se destacar com o aumento da necessidade de comunicação e utilização da informação a todo momento. Esta busca tem gerado concorrência entre grandes empresas e, como consequência, o desenvolvimento de plataformas móveis avançadas. A Plataforma Android vem se destacando no mercado, com interfaces simplificadas e funcionais, além de auxiliar os desenvolvedores para a criação de aplicativos robustos, com grande número de documentação sobre o código e também APIs que permitem, com facilidade, a integração de aplicações com diversas tecnologias e recursos dos dispositivos móveis atuais, como GPS e câmera. Outro mercado que vem crescendo é o imobiliário, com o aumento da busca por imóvel próprio. O crescimento neste mercado também gerou concorrência entre os corretores, que têm que encontrar novas técnicas de venda para se diferenciarem no mercado. Vendo esta necessidade dos corretores de imóveis, as vantagens da Plataforma Android e a falta de aplicação móvel que permita a gerência e apresentação de imóveis, foi proposto neste trabalho o desenvolvimento de uma aplicação utilizando a plataforma Android, visando a gerência de imóveis. A fim de oferecer funcionalidade na aplicação, foram utilizados diferentes recursos integrados a aplicação, como: GPS, banco de dados SQLite, câmera, serviço Google Maps e serviço Google StreetView. Desta forma o trabalho resulta em uma aplicação para corretores de imóveis de código livre e robusta, sendo disponibilizada aos acadêmicos interessados em realizar melhorias ou utilizar como base para novos projetos.

Palavras-chave: Imóveis. Corretor. Plataforma Android. Aplicação móvel.

ABSTRACT

Technology companies are, in a general way, searching more for mobility, having a differential with the increasing demand of communication and information in every moment. This search has been generating competition among big companies and, as a result of it, the developing of mobile advanced platforms. The Android platform has been overlapping other ones in this market, with simple and functional interface, apart from helping developers to create powerful apps, with the great documentation about the code and also APIs that allow, with ease, app´s integration with several technologies and resources of the mobile devices available, as GPS and camera. Another growing market is the housing market, with the increasing search for the own property. The growth of the market also increased competition among brokers, that had been finding new sales techniques to be different on the market. Observing this need of Real estate brokers, the advantages of Android platform and the lack of mobile application that allows management and house presentation, it was proposed on this work the developing of an application using Android, aiming the houses management. To offer functionality on this app, it was used a lot of integrated resources as: GPS, data bank SQLite, camera, Google Maps services and Google StreetView service. As a result of this work, Real state brokers have a new strong free coded app being released to students to improve developing or use it for new projects.

Keywords: Properties. Realtor. Android platform. Mobile application.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

AAC	*Advanced Audio Coding*
API	*Application Programming Interface*
AVC	*Advanced Volume Control*
CRECI	Conselho Regional de Corretores de imóveis
CRECISP	Conselho Regional de Corretores de imóveis de São Paulo
DCI	Diário Comércio Indústria e Serviços
GPS	*Global Positioning System*
HTML	*HyperText Markup Language*
MP3	*Moving Picture Experts Group 1 Audio Layer 3*
MPEG-4	*Moving Picture Experts Group Phase 4*
MVC	*Model-view-controller*
NDK	*Native Development Kit*
NFC	*Near Field Communications*
OHA	*Open Handset Alliance*
OS	*Operating System*
SCIESP	Sindicado dos Corretores de Imóveis no Estado de São Paulo
SDK	*Software Development Kit*
SMS	*Short Message Service*
SQL	*Structured Query Language*
UML	*Unified Modeling Language*
URI	*Uniform Resource Identifier*
XML	*Extensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

O mercado imobiliário está apresentando um crescimento no Brasil e fatores que têm forte influência são o aumento do crédito para pessoas físicas e das faixas salariais, que dão às famílias um maior poder aquisitivo (CINTRA, 2011; LINDENBERG FILHO, 2006).

Este crescimento desperta o interesse de investidores e também pela profissão de corretor de imóveis. Até mesmo profissionais de outras áreas estão se especializando para trabalharem como corretor de imóveis, devido às boas perspectivas de lucros que este mercado vem oferecendo (CRECISP, 2012; PAES, 2011).

O aumento de profissionais na área aumenta também a concorrência. Esta competitividade leva o cliente a escolher o corretor melhor preparado. Então, corretores menos experientes devem buscar métodos que sejam um diferencial para poderem entrar no mercado. Hoje a utilização de tecnologia na apresentação dos imóveis pode ser um grande diferencial (LINDENBERG FILHO, 2006).

Conforme buscas em sites como Google Play (GOOGLE INC., [2013?]a), hoje há aplicativos móveis de empresas que permitem que clientes possam realizar consultas de imóveis em seu banco de dados, porém nenhum destes aplicativos é para uso específico do corretor, impossibilitando que ele gerencie seu cadastro de imóveis e apresente a seus clientes.

Então, neste trabalho será desenvolvida uma aplicação móvel destinada a corretores de imóveis utilizando a plataforma Android, que segundo Ableson, Sem e King (2011, tradução nossa), está revolucionando o mercado global de telefonia celular por ser uma plataforma *open source* e com diversos recursos poderosos, permitindo o desenvolvimento de aplicações com muitas funcionalidades.

1.1 OBJETIVO GERAL

Desenvolver uma aplicação, utilizando a plataforma Android, para auxiliar corretores de imóveis na gerência de imóveis e na apresentação à clientes.

1.2 OBJETIVOS ESPECÍFICOS

Os objetivos específicos do projeto consistem em:

a) analisar a API Maps da Google e apresentar suas principais funções para manipular mapas;
b) verificar provedores de localização para os dispositivos android;
c) analisar como capturar imagens da câmera do aparelho com a programação Android e apresentar a solução;
d) aplicar o banco de dados SQLite que é nativo do Android;
e) analisar a função do corretor de imóveis e o mercado imobiliário;
f) desenvolver uma aplicação para auxiliar corretores de imóveis na gerência de imóveis e na apresentação a clientes.

1.3 JUSTIFICATIVA

Aplicações móveis englobam novas tecnologias e conceitos que algumas empresas ainda não conhecem, sendo que uma aplicação móvel pode trazer diversos benefícios para qualquer negócio, como: diferenciar-se de empresa concorrente, eficiência na produtividade e mobilidade no negócio (MALLICK, 2003).

Antes da implementação de uma aplicação móvel deve ser analisado em que plataforma será desenvolvida, buscando sempre o que há de melhor no mercado. Segundo pesquisas apresentadas no site Teleco (TELECO, [2012?]), o principal sistema operacional móvel da atualidade é o Android. Este possui código aberto e ainda pode ser utilizado por dispositivos de diferentes fabricantes, o que é uma vantagem para desenvolvedores e empresários que desejam investir nesta plataforma, pois não ficam limitados a uma única marca (LECHETA, 2010).

O Android conta com diversos recursos e aplicações nativas, como o Google Maps. Além disso, programadores de todo o mundo podem contribuir com o desenvolvimento desta tecnologia criando novas funcionalidades.

Outro ponto positivo é a possibilidade de desenvolver aplicações utilizando a linguagem Java e seus recursos, sendo Java uma linguagem de programação bem difundida no mercado (LECHETA, 2010).

Devido a esta diversidade de benefícios da plataforma, ela já está sendo bastante utilizada no mercado empresarial. No site Google Play (GOOGLE INC., [2013?]a), onde são publicadas as aplicações desenvolvidas para esta, pode ser encontrado com facilidade aplicações de empresas a fim de obter mobilidade, expandindo a divulgação de produtos e serviços.

Com base nas vantagens apresentadas dos dispositivos móveis e da plataforma Android e na dificuldade do corretor de imóveis em apresentar seu produto com mobilidade, propõe-se o desenvolvimento de uma aplicação Android que permita ao corretor obter vantagem competitiva em relação a concorrência, pois o software permitirá ao corretor apresentar informações detalhadas do imóvel, como localização através do Street View e fotos que possam dar mais detalhes sobre seus produtos sem a necessidade inicial de levar o comprador até o imóvel.

1.4 ESTRUTURA DO TRABALHO

O trabalho é composto por 7 capítulos. O primeiro faz uma breve descrição sobre o tema abordado, especifica o objetivo geral e os objetivos específicos e argumenta uma justificativa para o problema proposto.

O crescimento do mercado imobiliário foi apresentado no capitulo 2, assim como a função do corretor de imóveis, método para venda, informações importantes de imóveis e necessidade de tecnologia na área.

O sistema operacional Android é abordado no capítulo 3, onde é descrito evolução, estrutura e principais características deste sistema.

O capitulo 4 descreve sobre os principais recursos do Android utilizados no trabalho, que são: Google Maps, Street View, provedores de localização, câmera e banco de dados SQLite.

No capítulo 5 são apresentados alguns trabalhos relacionados ao proposto, passando uma breve descrição do que foi desenvolvido em cada um.

O sistema proposto é apresentado no capítulo 6, ao qual foi detalhado a metodologia empregada, os recursos aplicados e o desenvolvimento da aplicação.

No capítulo 7 estão as conclusões obtidas com os resultados obtidos, como também sugestões de trabalhos futuros e dificuldades encontradas.

2 MERCADO IMOBILIÁRIO

O consumo das famílias tem aumentado exponencialmente junto com a economia brasileira. Um dos fatores que tem permitido este crescimento é o aumento de crédito para pessoas físicas além do aumento das faixas salariais (CINTRA, 2011).

Segundo Lindenberg Filho (2006), este novo poder aquisitivo das famílias tem fomentado o mercado imobiliário, e a demanda por novos profissionais tem aumentado a cada ano no Brasil.

Este é um dos fatores que despertam o interesse pela profissão, devido às boas perspectivas de lucros que este mercado vem oferecendo (CRECISP, 2012).

A expansão do mercado imobiliário brasileiro atrai investidores que querem aproveitar a valorização, que vem sendo constante, neste segmento. Profissionais de outras áreas também estão se especializando para trabalharem como corretor de imóveis, a fim de desfrutar dos benefícios desta evolução (PAES, 2011).

O aumento no número de corretores de imóveis é percebido pelo número de inscritos no CRECI. Em 2011 o CRECISP já possuía mais 114 mil inscritos. Em 2009 foram 9152 novos profissionais e em 2010 foram 9835 (CRECISP, 2011).

Como a aquisição de um imóvel é um investimento financeiramente alto, os clientes que desejam comprar procuram o maior número de informações possíveis, com objetivo de obter segurança no negócio. Estas informações podem ser encontradas nos diferentes meios de comunicação social ou internet, apresentando diversas alternativas. Desta forma, o cliente acaba recorrendo ao corretor de imóveis para intermediar a aquisição devido ao número excessivo de informações que pode deixá-lo confuso (LINDENBERG FILHO, 2006).

2.1 INTERMEDIAÇÃO IMOBILIÁRIA

Segundo Brasil (1978, p. 1), a Lei n° 6.530 que disciplina o exercício da profissão diz que, “compete ao corretor exercer a intermediação na compra, venda, permuta e locação de imóveis, podendo, ainda, opinar quanto à comercialização imobiliária”.

Intermediação imobiliária é a ação de aproximar o comprador do imóvel ao vendedor e conduzir a transação para o sucesso. Então, as principais atividades do corretor de imóveis são (CRECISP,[2013?]; RAPOSO; HEINE, 2004):

a) obter informações detalhadas sobre aquisição, venda, locação, avaliação, preço, financiamentos, etc.;
b) firmar contrato relativo aos serviços prestados;
c) combinar preço e condições da transação;
d) manter-se atualizado com relação ao perfil do mercado imobiliário;
e) analisar a documentação do imóvel, dando ciência a inquilinos e/ou compradores;
f) agendar visitas ao imóvel, apresentando-o ao cliente;
g) orientar todo cliente que queira investir em imóveis.

Deve-se ter atenção no contato entre o corretor e o cliente, pois o marketing imobiliário consiste basicamente no relacionamento com os clientes. Então, se bem trabalhado este contato, será um diferencial que gerará mais clientes fiéis e satisfeitos à empresa ou profissional (COMIN, 2003).

Quando o corretor encontra um pretendente à compra de imóvel, ele deve fazer uma análise junto ao comprador para saber o tipo de imóvel que ele deseja, como por exemplo, sua localização, quando deseja obter o imóvel, entre outros fatores. Após o recolhimento destas informações, o corretor verifica entre os imóveis disponíveis, o que se aproxima do perfil desejado pelo cliente (COMIN, 2003).

Lindenberg Filho (2006) recomenda que o corretor de imóveis utilize uma metodologia de vendas com cinco fases, a saber: abordagem, entrevista, apresentação/demonstração, objeção e fechamento. Veremos o que ele diz sobre as fases apresentação/demonstração e objeção, que é o foco neste trabalho:

a) na fase de apresentação e demonstração devem ser evidenciados de maneira direta e com clareza os benefícios que o cliente pode obter com o imóvel oferecido. Esta apresentação deve levar em consideração os principais critérios de avaliação, apresentados na tabela 1;
b) na fase da objeção é que o cliente se manifesta com sua opinião e/ou solicita mais detalhes sobre o imóvel. O corretor deve estar aberto para estas objeções e ter argumentação para responder todas as dúvidas que ainda restaram.

Para poder satisfazer e ter uma relação comercial ampla, o corretor deve ter um mínimo de informações para que lhe permita estar no nível do cliente (LINDENBERG FILHO, 2006).

Os clientes têm receios na compra do imóvel, como, de ser enganado ou de errar na decisão. Desta forma o corretor de imóveis deve apresentar o maior número de informações possíveis sobre o imóvel, a fim de passar segurança ao cliente e a credibilidade de que está tomando a decisão correta (LINDENBERG FILHO, 2006).

Segundo o presidente do CRECISP, José Augusto Viana Neto, hoje os clientes estão mais exigentes, cobrando informações detalhadas do imóvel que desejam adquirir, conforme apresentado na tabela 1. Assim o corretor de imóveis precisa estar muito mais preparado, tanto educacionalmente como tecnologicamente, para poder atender com agilidade a questionamentos de seus clientes (CRECISP, 2012).

Tabela 1 - Critérios de avaliação.

Critérios de avaliação mais usados no processo decisório	
Área do terreno	Distribuição interna
Ano de construção	Acabamento
Localização	Infra-estrutura
Preço	Despesas de condomínio
Metragens	Serviços disponíveis
Número de dormitórios	Número de unidades
Orientação solar	Número de garagens

Fonte: Lindenberg Filho (2006).

Destes critérios, o que tem maior influência é a localização do imóvel (LINDENBERG FILHO, 2006).

O preço do imóvel deve ser uma das últimas informações a ser passada para o cliente. Primeiro deve ser apresentado informações que atraiam seu interesse. Se trabalhar com valores no início das negociações, o cliente pode perder o interesse sem ter conhecido o imóvel (LINDENBERG FILHO, 2006).

A mudança no perfil do corretor que vem ocorrendo está atraindo profissionais de faixas etárias cada vez mais novas (CRECISP, 2012).

Segundo Odil Baur de Sá, presidente emérito do SCIESP, os jovens têm mais conhecimento tecnológico, assim têm mais facilidade para ingressar na

profissão, pois o acesso à informação é mais rápido com o uso da tecnologia (CRECISP, 2012).

O aumento de profissionais na área aumenta também a concorrência. Esta competitividade leva o cliente a escolher o corretor mais preparado. (LINDENBERG FILHO, 2006).

Com esta competitividade nota-se a necessidade de os corretores se diferenciarem para poderem se destacar na profissão. Com o ingresso de profissionais cada vez mais jovens nesta área e com maior conhecimento tecnológico, a aplicação móvel pode ser utilizada como meio para apresentar seu produto de forma diferenciada.

Desta forma, apresento o Sistema Operacional Android como alternativa para encapsular um aplicativo de alto nível que pode auxiliar os corretores na efetivação das vendas dos imóveis.

3 SISTEMA OPERACIONAL ANDROID

O desenvolvimento para dispositivos móveis é muito limitado se comparado com o desenvolvimento para desktop, pois o programador não tem a mesma liberdade para trabalhar com o sistema operacional e interagir com os hardwares do dispositivo, como no desenvolvimento desktop. Os fabricantes destes dispositivos utilizavam sistemas operacionais proprietários de código fechado para proteger seus “segredos” de hardware. Até hoje, com o avanço e diversos recursos que existem nos dispositivos móveis, muitos fabricantes ainda utilizam sistema operacional proprietário. Mas o lançamento do Android, em novembro de 2007, mudou este cenário (DIMARZIO; POLO, 2008, tradução nossa).

O Android é uma plataforma que está revolucionando o mercado global de telefonia celular. É a primeira plataforma *open source* para aplicação móvel que influenciou os principais mercados de telefonia móvel de todo o mundo (ABLESON; SEM; KING, 2011, tradução nossa).

O Android chegou ao mundo dos sistemas operacionais móveis, em uma época em que os dispositivos utilizavam diversos sistemas proprietários, como Symbian OS, Microsoft’s Windows Mobile, Mobile Linux, iPhone OS (baseado no Mac OS X) e Moblin (da Intel). Nenhum destes se tornou o padrão para dispositivos móveis, pois as APIs disponíveis e ambientes para desenvolvimento de aplicações móveis são muito restritas (HASHIMI; KOMATINENI; MACLEAN, 2010).

A chegada do Sistema Operacional Android encantou os consumidores, mas a grande diferença ocorreu para os desenvolvedores, que antes não tinham muitas alternativas para o desenvolvimento móvel, pois os sistemas operacionais existentes eram proprietários e restringiam aplicativos de terceiros. Com o Android o desenvolvedor tem liberdade para implementar suas aplicações que podem utilizar recursos cada vez mais poderosos do *hardware* móvel (MEIER, 2010, tradução nossa).

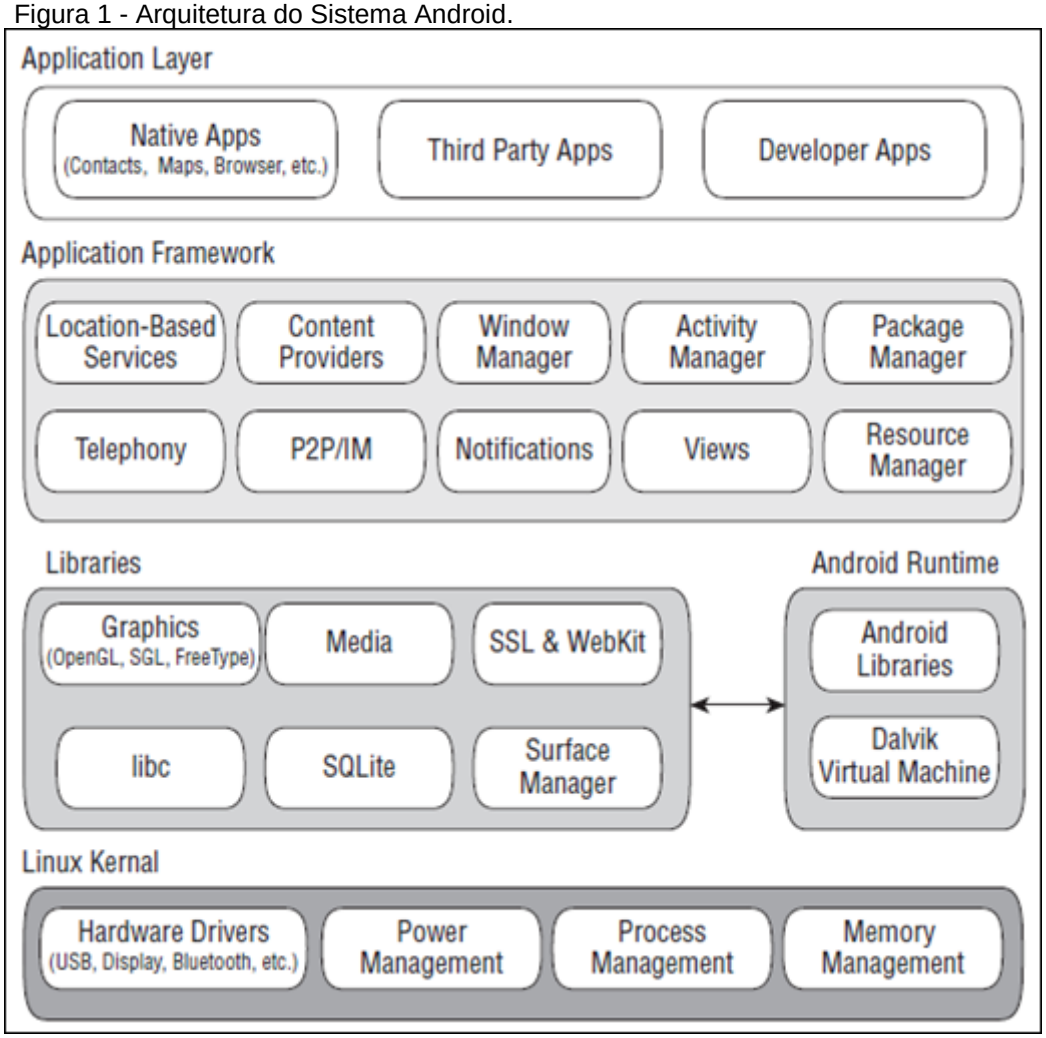
O grande sucesso do Android também pode ser explicado pelas grandes empresas que estão envolvidas no seu desenvolvimento. Ele foi criado pela Open Handset Alliance (OHA), que é uma aliança liderada pela gigante e revolucionária Google e conta com empresas líderes no mercado de telefonia, como a Motorola, LG, Samsung, Sony Ericsson e muitas outras (LECHETA, 2010).

3.1 ARQUITETURA

O OS Android foi baseado no kernel do Linux, e sobre ele estão as bibliotecas que dão suporte ao framework. As aplicações são desenvolvidas em Java e são executadas com a máquina virtual Dalvik, que é otimizada para execução em dispositivos móveis (LECHETA, 2010).

Meier (2010, tradução nossa) apresenta de forma mais estruturada a arquitetura do Android, conforme apresentado na figura 1.

Figura 1 - Arquitetura do Sistema Android.



Fonte: Meier (2010, tradução nossa).

3.1.1 Kernel Linux

Serviços essenciais, incluindo *drivers* de hardware, processos e gerenciamento de memória, segurança, rede e gerenciamento de energia, são tratados por um kernel Linux 2.6. O kernel também fornece uma camada de ligação entre o hardware e o restante da pilha da arquitetura Android (MEIER, 2010, tradução nossa).

A utilização de uma plataforma cheia de recursos, como o kernel do Linux, proporciona enorme poder e capacidades para o Android. Utilizando uma plataforma de código aberto dá a possibilidade de pessoas talentosas e empresas para desenvolvê-la ainda mais (ABLESON; SEM; KING, 2011, tradução nossa).

3.1.2 Bibliotecas

Segundo Burnette (2009, tradução nossa), acima do Kernel há as bibliotecas nativas do Android, que são compiladas para uma arquitetura de hardware específica usada em um aparelho e pré-instalada pelo fabricante. Para o desenvolvimento destas foi utilizado a linguagem C ou C++.

O autor complementa que, algumas das mais importantes bibliotecas são:

a) gerenciador de superfície: utiliza um compositor e gerenciador de janelas. Os comandos de desenho vão para outro *buffer* de imagem, que combinados com outras imagens formam o que o usuário visualiza, ao invés de desenhar diretamente no buffer da tela. Este sistema permite a criação de efeitos, como as transações suaves entre as telas e ver através de janelas;
b) gráficos 2D e 3D: elementos bidimensionais e tridimensionais são combinados em uma única interface de usuário. A biblioteca utiliza aceleração 3D através do hardware se este tiver suporte;
c) codecs de mídias: pode reproduzir e gravar áudio e vídeo em diferentes formatos, incluindo AAC, AVC (H.264), H.263, MP3 e MPEG-4;
d) base de dados SQL: utiliza uma versão do SQLite, mesma base de dados utilizada no navegador Firefox e no iPhone da Apple. É possível utilizar armazenamento persistente nas aplicações;
e) *browser engine*: para a exibição rápida de conteúdo HTML, utiliza WebKit, com licença *open source*. Mesmo utilizado no navegador da

Google (o Chrome), no navegador da Apple (o Safari), no iPhone da Apple e na plataforma S60 da Nokia.

Estas são as bibliotecas nativas mais importantes e são utilizadas por programas que executam em um nível superior da arquitetura Android. Mas a plataforma não é limitada somente a estas bibliotecas, é possível criar e implantar bibliotecas através do Native Development Tookit (NDK).

3.1.3 Android *Runtime*

Também acima do kernel do Linux, está o Android Runtime, que é composto por bibliotecas adicionais do Android e a máquina virtual Dalvik, conforme apresentado abaixo, que formam a base para o *framework* de aplicação (BURNETTE, 2009, tradução nossa; MEIER, 2010, tradução nossa):

a) Bibliotecas do Android: Embora o desenvolvimento para Android seja feito em Java, Dalvik não é uma máquina virtual Java. As bibliotecas do Android fornecem a maioria das funcionalidades das bibliotecas do núcleo Java, como também, bibliotecas específicas para o Android;

b) Máquina Virtual Dalvik: é uma máquina virtual baseada em registradores que foi otimizada para garantir que um dispositivo possa executar várias instâncias eficientemente. Ele tem como base o Kernel do Linux para gerenciamento de *threading* e memória de baixo nível.

3.1.4 *Framework* de Aplicação

Acima da camada de bibliotecas e da *runtime*, está a camada *framework* de aplicação. Esta disponibiliza classes de desenvolvimento de alto nível, que podem ser utilizados por desenvolvedores na criação de aplicativos, e também faz a abstração do acesso ao hardware e gerencia a interface de usuário e recursos de aplicações. Esta camada já vem pré-instalada com o Android, mas pode ser estendida (BURNETTE, 2009, tradução nossa; MEIER, 2010, tradução nossa).

Segundo Burnette (2009), as partes mais importantes desta estrutura são:

a) *Activity Manager:* controla o ciclo de vida das aplicações e também é responsável por manter uma *backstack* (“pilha”) da navegação do usuário;

b) *Content providers:* estes objetos encapsulam dados que precisam ser compartilhados entre aplicativos, como a lista de contatos;
c) *Resource manager:* recurso engloba todos os componentes de uma aplicação que não sejam código;
d) *Location manager:* dispositivos Android possuem recursos que permitem saber a sua localização;
e) *Notification manager:* eventos como mensagens que chegam, compromissos, alertas e muito mais podem ser apresentados de uma forma discreta.

3.1.5 Camada de Aplicação

A camada de aplicação está no topo da arquitetura Android. Nesta camada os aplicativos nativos e de terceiros são construídos utilizando as mesmas bibliotecas de API. Esta roda em conjunto com o Android Runtime e utiliza as classes e serviços disponibilizados pela camada *framework* de aplicação (MEIER, 2010, tradução nossa).

3.2 VERSÕES OFICIAIS

A Google Inc. ([2013?]b, tradução nossa) divulgou uma tabela com o número relativo de dispositivos que executam uma determinada versão da plataforma Android, que está sendo apresentada na tabela 2. Os dados foram coletados durante um período de 14 dias encerrado em 02 de abril de 2013. Versões com menos de 0,1% de distribuição não são apresentadas.

Tabela 2 - Distribuição das versões.

Version	Codename	API	Distribution
1.6	Donut	4	0.1%
2.1	Eclair	7	1.7%
2.2	Froyo	8	4.0%
2.3 - 2.3.2	Gingerbread	9	0.1%
2.3.3 - 2.3.7		10	39.7%
3.2	Honeycomb	13	0.2%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.3%
4.1.x	Jelly Bean	16	23.0%
4.2.x		17	2.0%

Fonte: Google Inc. ([2013?]b).

Como apresentado na tabela 2, a versão 2.3 é a mais distribuída. Segundo Google Inc. ([2013?]b, tradução nossa), esta versão possui vários recursos, sendo que os principais são listados abaixo:

a) Android 2.3 – 2.3.7, API 10 (Gingerbread – Dezembro de 2010):

- API SIP, para desenvolvimento aplicações com uso de telefonia,
- *Near Field Communications* (NFC),
- giroscópio e outros sensores,
- suporte a múltiplas câmeras,
- mixagem de áudio,
- gerenciador de download,
- aplicativo StrictMode (auxilia desenvolvedores a monitorar e melhorar o desempenho de aplicativos),
- suporte para *overscroll*,
- eventos de movimento,
- controles de seleção de texto,
- suporte a telas maiores (*Tablets*),
- suporte a conexões Bluetooth não seguras,
- API's gráficas,

- API de reconhecimento de voz.

A versão mais recente é a 4.2 (Jelly Bean), liberada em Outubro de 2012, mas considerando que a versão 2.3 é a mais distribuída e já possui os recursos necessários para o desenvolvimento deste trabalho, ela será utilizada para implementação do software proposto.

3.3 CARACTERÍSTICAS DO ANDROID

Em uma visão mais ampla, o Android possui algumas características principais que são importantes pontos para diferenciá-lo no mercado, como Multiprocesso, Toque, Gestos, Multitoque (STEELE; TO, 2010, tradução nossa). Será descrito um pouco sobre estas características a seguir.

3.3.1 Multiprocesso

O sistema operacional Android não restringe o processador para executar um aplicativo de cada vez. O sistema gerencia as prioridades de cada aplicação para a utilização do processador. Isto é um grande benefício do Android, já que permite que sejam executadas tarefas em segundo plano mesmo quando o usuário está utilizando o dispositivo (STEELE; TO, 2010, tradução nossa).

3.3.2 Toque, Gestos e Multitoque

As telas sensíveis ao toque (*touchscreen*) são uma forma intuitiva de interação com aparelhos móveis. Sendo uma maneira bem natural de interação, já que as instruções podem ser dadas com o toque do dedo na tela. O multitoque (*Multitouch*) permite instruções com o toque de mais de um dedo na tela ao mesmo tempo. Isto é frequentemente usado para fazer zoom ou girar o que está sendo visualizado (STEELE; TO, 2010, tradução nossa).

O toque feito com o dedo na tela do dispositivo torna a interação menos precisa e muitas vezes é baseado mais no movimento do que no simples contato. Aplicativos nativos do Android fazem uso extensivo de interfaces que recebem o toque do dedo do usuário, incluindo o uso de arrastar, movimentos para percorrer listas ou executar ações (MEIER, 2010, tradução nossa).

Estas funções de toque e gestos permite uma interação amigavel com mapas, que será um dos recursos da aplicação a ser desenvolvida neste trabalho.

O Android conta com recursos que impressionam, como a navegação na internet com a tela *touchscreen*, o uso do Google Maps e GPS em aplicações. Trabalhar com estes recursos se torna fácil para os desenvolvedores devido as diversas API's que já estão disponíveis para o Android que permitem uma fácil integração das aplicações com os diversos recursos (LECHETA, 2010).

A API, ou interface de programação de aplicativo, é o núcleo do SDK do Android. Uma API é uma coleção de funções, métodos, propriedades, classes e bibliotecas que é utilizado para criar programas que trabalham com plataformas específicas. O Android API contém todas as informações específicas necessárias para criar aplicativos que podem trabalhar e interagir com um dispositivo baseado em Android. O Android SDK também contém dois conjuntos complementares de APIs: do Google e opcionais (DIMARZIO; POLO, 2008, tradução nossa). No próximo capítulo serão apresentados alguns recursos e APIs que são importantes para o desenvolvimento deste trabalho.

4 RECURSOS DO ANDROID

Neste capítulo serão apresentados alguns dos recursos do Android que serão utilizados para o desenvolvimento deste trabalho.

4.1 GOOGLE MAPS

O Google Maps é um serviço de mapas que permite visualizar mapas básicos ou personalizados e informações sobre empresas locais, incluindo as respectivas localizações, sendo possível visualizar imagens de satélite, ampliando, reduzindo ou movimentando. Este serviço pode ser utilizado de várias formas, como por exemplo, por meio de uma página da Web com o Google Maps incorporado ou com um dispositivo móvel (GOOGLE INC., [2013?]c).

A figura 2 apresenta um diagrama com alguns recursos disponíveis no Google Maps.

Figura 2 - Recursos do Google Maps.

Fonte: Adaptado de Google Inc.([2013?]d).

Considerando que os recursos expostos na figura 2 são funcionalidades importantes para a aplicação de corretores de imóveis, uma breve descrição sobre elas são apresentadas abaixo (GOOGLE INC., [2013?]d, tradução nossa):

a) pesquisa do Google Maps: busca de lugares, empresas, cruzamentos, endereços, etc;
b) mapa: a área do mapa mostra o local geográfico com os resultados de pesquisa correspondentes e outras informações desse local;
c) Street View: permite visualização e navegação pelas imagens das ruas;
d) janela de informações: ao clicar em uma janela de informações, ela exibirá dados adicionais sobre um determinado lugar;
e) visualizações - alterna entre Mapa, Satélite e Terra.

O Google Maps em dispositivos móveis tem sido extremamente popular, e o Android oferece fácil controle e utilização deste serviço da Google. Com uma interface de programação familiar a do Google Maps é possível desenvolver com facilidade aplicações que apresentam e manipulam mapas (MEIER, 2010, tradução nossa).

A Google mantém APIs para permitir a sincronização de dados dos seus serviços com o aplicativo Android. Entre estas APIs destaca-se a desenvolvida para o Google Maps que permite a visualização e o controle de mapas com facilidade (HASHIMI; KOMATINENI; MACLEAN, 2010).

Com a API do Google Maps pode ser manibulado o mapa de várias maneiras, como inserindo itens para marcação de uma localização, conforme apresentado na figura 3 a localização de um determinado imóvel utilizando uma imagem (LECHETA, 2010).

Figura 3 - Overlay no mapa.

Fonte: Do autor.

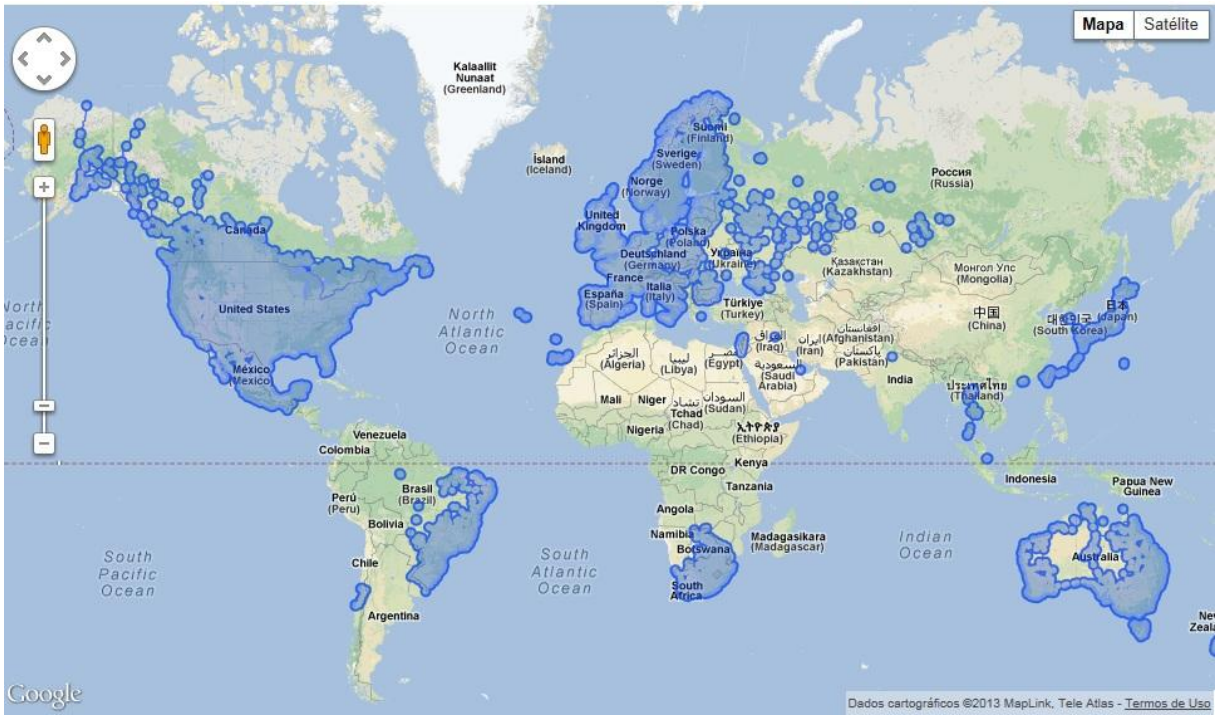
Nos itens do mapa é possivel tratar o evento do clique para que execute uma ação, como apresentar uma notificação com detalhes do local ou até mesmo abrir uma outra tela (LECHETA, 2010).

Estas seriam as principais funcionalidades para trabalhar com o Google Maps em uma aplicação Android. E para trabalhar em conjunto com o Google Maps, a Google criou um serviço capaz de apresentar imagens mais reais de uma determinada localização, o Street View.

4.2 STREET VIEW

O Street View permite explorar lugares por meio de imagens em 360 graus no nível da rua. Foi lançado em maio de 2007 com imagens de cinco cidades dos Estados Unidos e hoje já possui imagens dos sete continentes. A figura 4 apresenta destacado em azul os locais que as imagens do Street View estão disponíveis (GOOGLE INC., [2013?]e).

Figura 4 - Imagens do Street View no mundo.



Fonte: Google Inc. ([2013?]e).

A figura 5 apresenta em mais detalhes as regiões do Brasil com imagens do Street View, considerando que grande parte das metrópoles brasileiras já estão mapeadas.

Figura 5 - Imagens do Street View no Brasil.

Fonte**:** Google Inc. ([2013?]e).

A figura 6 apresenta um exemplo de como é exibido os lugares pelo Street View, sendo uma montagem de fotos da cidade de Londres, na Inglaterra, de forma que apresenta o local em 360 graus.

Figura 6 - Imagem de Londres em 360 graus.

Fonte: Google Inc. ([2013?]e).

O serviço Street View da Google pode ser chamado por uma *intent* em uma aplicação Android (GOOGLE INC., [2013?]f, tradução nossa).

Uma *intent* representa a intenção da aplicação de realizar determinada tarefa. A *intent* é enviada para o sistema operacional, que por sua vez interpreta a mensagem e responde a solicitação (LECHETA, 2010).

Para a chamada de um serviço uma *intent* deve receber como parâmetro a ação e um Identificador Uniforme de Recursos (URI) com o recurso a ser solicitado. Então para chamada do Street View a ação deve ser VIEW e o URI deve ser google.streetview:cbll=lat,lng&cbp=1,yaw,,pitch,zoom&mz=mapZoom. Neste URI o campo cbll é obrigatório e os campos cbp e mz são opcionais. A tabela 3 apresenta o que seria cada um dos parâmetros.

Tabela 3 - Parametros do URI para Street View.

lat	**Latitude**
lng	Longitude
yam	Ângulo de vista em graus no sentido horário a partir do Norte
pitch	Ângulo de visão vertical em graus de -90 (olhar para cima) a 90 (olhar para baixo)
zoom	Zoom da imagem. 1,0 = normal, 2,0 = ampliada em 2x, 3,0 = 4x ampliado, e assim por diante.
mapzoom	Zoom que será repassado ao mapa quando retornar ao modo de visualização no mapa.

Fonte: Adaptado de Google Inc. ([2013?]f, tradução nossa).

4.3 PROVEDORES DE LOCALIZAÇÃO

O Android pode obter a posição geográfica do dispositivo utilizando o GPS e, com menos precisão, utilizando informações de torres de telefonia celular nas proximidades, ou até mesmo uma rede Wi-Fi quando o dispositivo estiver conectado (BURNETTE, 2009, tradução nossa).

Devido à precisão e potência do GPS, normalmente os softwares são desenvolvidos utilizando este recurso. No entanto, ele não pode acessar os dados do satélite em locais cobertos ou alguns modelos de dispositivo não o possuem. Desta forma, outras opções de provedor de localização disponibilizadas pelo Android também são úteis (ABLESON; SEM; KING, 2011, tradução nossa).

O GPS é um sistema de posicionamento global baseado em tecnologia de satélites. As posições dos satélites são previstas e transmitidas junto com o sinal de GPS para dispositivos com esta tecnologia. Através de várias posições conhecidas e a distâncias entre o receptor e os satélites é possível determinar a posição do receptor (XU, 2007, tradução nossa).

Os telefones móveis possuem cada vez mais recursos e o GPS está entre os mais disponíveis nestes aparelhos. E para facilitar o desenvolvimento de aplicações que utilizem este recurso, o Android SDK contém uma API para trabalhar com qualquer hardware de GPS (DIMARZIO; POLO, 2008, tradução nossa; MEIER, 2010, tradução nossa).

O Android possui API de localização de fácil utilização. Com ela pode ser calculado a localização do dispositivo e após exibir em um mapa utilizando o serviço do Google Maps (ABLESON; SEM; KING, 2011, tradução nossa; DIMARZIO; POLO, 2008, tradução nossa). Além disso, uma classe fornece acesso aos serviços do sistema. Esta permite obter na aplicação a ultima localização reconhecida do provedor por meio de um de seus métodos. Neste método deve ser passado como parâmetro o provedor a ser utilizado, que pode ser: provedor por GPS e provedor por rede.

Esta API facilita a localização geográfica do dispositivo por meio de diferentes provedores. Este recurso será utilizado neste trabalho no cadastro de imóveis, por esta ser uma das mais importantes informações para venda de um imóvel.

4.4 CÂMERA

É comum em dispositivos móveis os aplicativos utilizarem a câmera integrada à aplicação. Para utilizar este recurso o Android possui API que permite manipular a câmera do dispositivo através da aplicação (LECHETA, 2010).

Mas no Android também é possível fazer a integração de aplicações. Desta forma, ao invés de utilizar a API e ter o trabalho de manipular a câmera pela aplicação, pode-se chamar a aplicação nativa da câmera para fazer a captura da imagem e retornar um objeto Bitmap, que representa uma imagem no Android (LECHETA, 2010).

Para utilizar a aplicação nativa da câmera, primeiro deve ser criada uma *Intent*, que representa uma ação que a aplicação deseja executar. *Intent* em português é intenção, e esta intenção é enviada como uma mensagem para o sistema operacional, que ao receber executa a ação de acordo com esta mensagem (LECHETA, 2010).

Então para a captura da imagem utilizando a aplicação nativa é feito a solicitação do serviço de câmera, assim a interface da aplicação nativa será apresentada ao usuário para que possa tirar uma foto ou vídeo. Na aplicação deve ser configurado o método *onActivityResult()* que recebe o retorno ao fim da operação do serviço solicitado, assim pode ser recuperado a imagem da aplicação nativa (GOOGLE INC., [2013?]g, tradução nossa).

A figura 7 apresenta um exemplo de código onde é executada a chamada do serviço. A figura 8 apresenta a implementação do método que recebe o retorno da aplicação nativa da câmera.

Figura 7 - Criando Intent para chamada de aplicação da câmera e inicialização.

```
private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;
private Uri fileUri;

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    // create Intent to take a picture and return control to the calling application
    Intent intent = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);

    fileUri = getOutputMediaFileUri(MEDIA_TYPE_IMAGE); // create a file to save the image
    intent.putExtra(MediaStore.EXTRA_OUTPUT, fileUri); // set the image file name

    // start the image capture Intent
    startActivityForResult(intent, CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE);
}
```

Fonte: Google Inc. ([2013?]g).

Figura 8 - Implementação do método onActivityResult.

```
private static final int CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE = 100;

@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
    if (requestCode == CAPTURE_IMAGE_ACTIVITY_REQUEST_CODE) {
        if (resultCode == RESULT_OK) {
            // Image captured and saved to fileUri specified in the Intent
            Toast.makeText(this, "Image saved to:\n" +
                     data.getData(), Toast.LENGTH_LONG).show();
        } else if (resultCode == RESULT_CANCELED) {
            // User cancelled the image capture
        } else {
            // Image capture failed, advise user
        }
    }
}
```

Fonte: Google Inc. ([2013?]g).

Conforme apresentado acima, para obter uma imagem ou vídeo com Android utilizando a aplicação nativa é simples e de rápido desenvolvimento, já que a aplicação apenas solicita e recebe os dados, sem ter o trabalho de manipular a câmera.

4.5 BANCO DE DADOS SQLITE

O armazenamento em arquivos funciona bem quando a quantidade de dados é pequena ou são de um mesmo tipo como áudio e vídeo. Para o armazenamento de dados em grande quantidade a melhor maneira é utilizar um banco de dados relacional (BURNETTE, 2009, tradução nossa).

O Android inclui o banco de dados SQLite, que pertence a lista de bibliotecas nativas. Este é um dos bancos de dados mais implantados em todo o mundo, sendo utilizado pelo Firefox e iPhone da Apple, e a sua popularidade se explica por ser gratuito, pequeno e não necessitar de nenhuma configuração ou administração (BURNETTE, 2009, tradução nossa).

Existem três formas para criação do banco de dados SQLite com Android, sendo uma delas com a própria API do Android. Esta é a forma recomendada, pois a própria aplicação pode criar o banco de dados, o que torna mais prático. Então esta será a forma utilizada neste trabalho (LECHETA, 2010).

O SQLite armazena dados do tipo *null*, *integer*, *real*, *text* e *blob*. Blob é o armazenamento de um objeto binário, então permite o armazenamento de qualquer objeto que é convertido para este formato. Imagem é um exemplo de objeto que pode ser convertido para o formato binário e pode ser armazenado no SQLite. Como neste trabalho serão armazenadas também imagens obtidas através da aplicação nativa da câmera, será possível salvar estas imagens também no banco de dados (SQLITE, [2013?]).

Com o banco de dados SQLite no Android pode ser visto a facilidade de armazenar grande quantidade e diversidade de dados com persistência.

5 TRABALHOS CORRELATOS

Foram encontrados trabalhos acadêmicos que desenvolveram aplicações com a plataforma Android, utilizando alguns recursos e APIs que serão utilizados neste trabalho. Entre estes, encontramos também trabalhos voltados para o mercado imobiliário, como o Mobile Real Estate Agent for Android (GUPTA, 2011).

5.1 MOBILE REAL ESTATE AGENT FOR ANDROID

Dissertação de Mestrado, desenvolvido pelo acadêmico Rohit Kumar Gupta, para o curso de Ciência da Computação pela San Diego State University, no ano de 2011.

Neste trabalho foi desenvolvida uma aplicação móvel utilizando a plataforma Android que permite a busca por informações de imóveis, com funções integradas, como também por corretor de imóveis. Esta é feita em um banco de dados Web Service específico. Pode ser feita com base no código, na localização, no status, entre outras informações. Após a busca, pode ser visto a localização em um mapa, como também visualizar a rota para chegar ao imóvel, onde foi utilizado a API Google Maps. A busca por corretor de imóveis apresenta informações como o telefone, que permite iniciar uma ligação pela própria aplicação (GUPTA, 2011).

5.2 DESENVOLVIMENTO DE APLICATIVO PARA SMARTPHONE COM A PLATAFORMA ANDROID

Trabalho de conclusão de curso, desenvolvido pelo acadêmico Rafael J. Werneck de A. Martins, para o curso de Engenharia de Computação pela Pontifícia Universidade Católica do Rio De Janeiro, no ano de 2009, para obtenção do título de Bacharel, com orientação do prof. Bruno Feijó. Teve como objetivo destacar as principais características do Android, como os componentes fundamentais de suas aplicações, interface com o usuário, recursos, armazenamento de informações, mapas e localização, e demonstrar como elas podem ser utilizadas através da criação de uma aplicação que explore os recursos da plataforma (MARTINS, 2009).

Ao fim do trabalho foi desenvolvida uma aplicação Android que exibe rotas, capturadas através de GPS, em um mapa, utilizando a API do Google Maps (MARTINS, 2009).

5.3 G-SMS: PROTÓTIPO DE APLICAÇÃO DE ENVIO DE SMS GEOREFERENCIADAS

Trabalho de conclusão de curso, desenvolvido pelo acadêmico Carlos Roberto Bender, para o curso de Ciência da Computação pela Universidade Regional de Blumenau – FURB, no ano de 2011, para obtenção do título de Bacharel, com orientação do prof. Dr. Mauro Marcelo Mattos.

O objetivo foi desenvolver uma aplicação Android que faça o envio de SMS georeferenciada de um dispositivo para outro e que o receptor apresente em um mapa a localização recebida. Para o desenvolvimento deste trabalho foi necessário a utilização da API Google Maps, para apresentação da localização no mapa, da API para gerencia e enviou de SMS, e API de localização, para obter a posição geográfica a partir do GPS ou rede móvel (BENDER; MATTOS, 2011).

5.4 INTERFACEAMENTO ENTRE DISPOSITIVOS MÓVEIS E JOGOS ELETRÔNICOS DE COMPUTADORES UTILIZANDO O ACELERÔMETRO BASEADO NO SISTEMA OPERACIONAL ANDROID

Este trabalho de conclusão de curso, desenvolvido pelo acadêmico William Bertan da Silva, para o curso de Ciência da Computação pela Universidade do Extremo Sul Catarinense – UNESC, no ano de 2012, para obtenção do título de Bacharel, com orientação do prof. MSc. Paulo João Martins. Teve como objetivo principal desenvolver uma aplicação utilizando-se do acelerômetro para captura de movimentos do dispositivo móvel para controle de jogo eletrônico em um servidor remoto. Para implementação do trabalho foi necessário estudo do desenvolvimento para dispositivos móveis com Android e das capacidades do acelerômetro em ambiente móvel (SILVA, 2012).

6 JERIMÓVEIS – APLICAÇÃO MÓVEL PARA CORRETORES

A presente pesquisa consiste no desenvolvimento de uma aplicação móvel que utiliza a plataforma Android, voltada para corretores de imóveis, denominada Jerimóveis. A aplicação tem como principais funções o cadastro de imóveis com posicionamento geográfico, imagens e exibição destas informações de forma intuitiva. Utiliza recursos da plataforma como Google Maps, Street View, Câmera e Banco de dados, proporcionando ao corretor de imóveis uma ferramenta eficiente para auxílio em seu trabalho.

6.1 METODOLOGIA

Primeiramente foi realizado levantamento bibliográfico com relação à plataforma Android e o mercado imobiliário.

Após esta fase, com o objetivo de entender a função do corretor de imóveis e suas necessidades, foi realizado um estudo sobre a intermediação imobiliária permitindo que a partir deste estudo fossem encontradas dificuldades que poderiam ser mitigadas com o uso da tecnologia móvel.

Tendo conhecimento das necessidades do corretor para exercer sua função, foi estudada a plataforma Android para obter mais detalhes sobre os recursos e funções disponibilizados. Após este estudo, foram definidos e descritos os recursos a serem utilizados pela aplicação.

A partir destes levantamentos foi possível definir os principais recursos que deveriam ser desenvolvidos na aplicação para o corretor de imóveis.

Além dos testes executados ao longo do desenvolvimento do projeto, foram realizados testes adicionais na aplicação para verificar que o software desenvolvido possui uma qualidade aceitável para suprir as necessidades do corretor. Estes testes foram realizados em diferentes dispositivos reais com versões 2.3.3, 3.2, 4.0.3 e 4.1.2 do Android para assegurar sua compatibilidade com a versão 2.3 e com versões mais recentes.

Para o desenvolvimento foi necessários um ambiente de desenvolvimento Eclipse com a SDK Android.

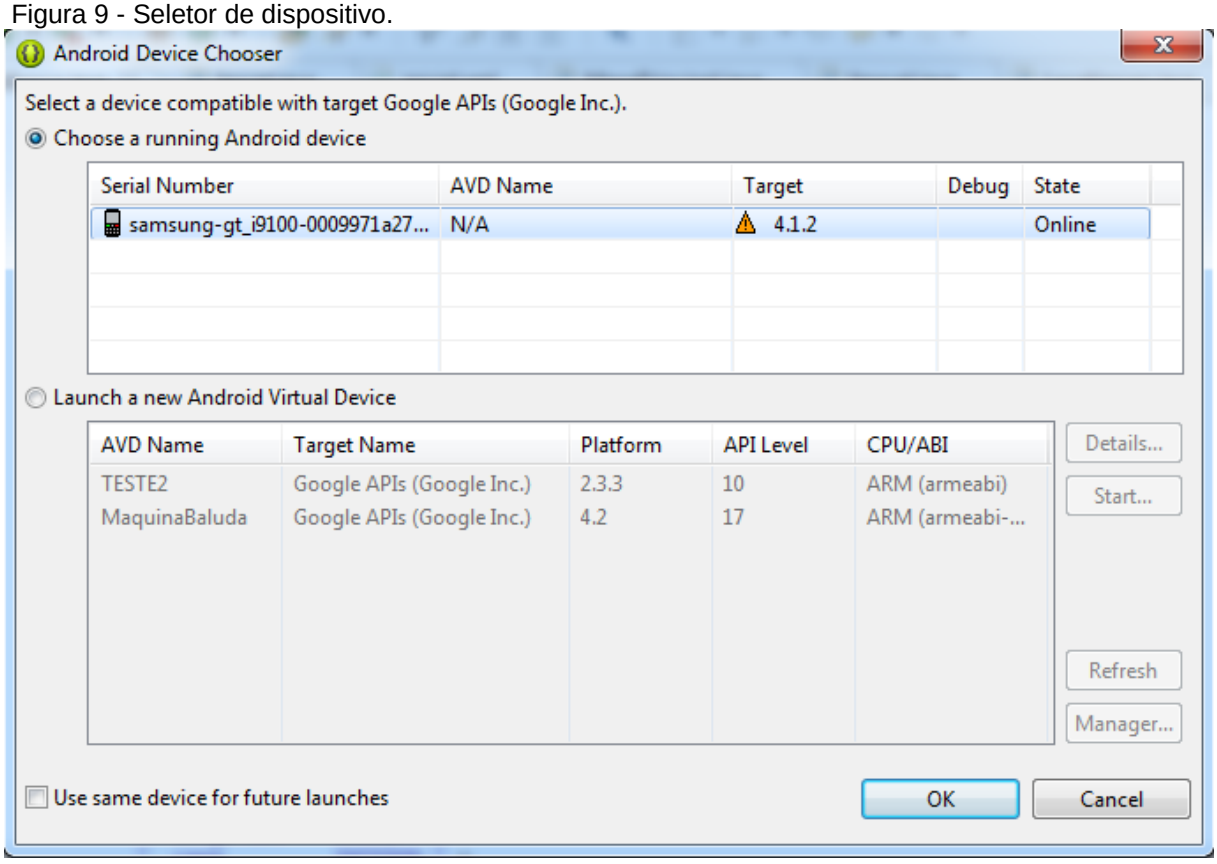
6.1.1 Estudo das ferramentas

Foi utilizado a IDE Eclipse com o *plugin* Android Development Tools (ADT) na versão 21.0.0 para o desenvolvimento da aplicação. Este *plugin* amplia os recursos do Eclipse para o desenvolvimento de projetos para Android, adicionando os pacotes de API Android.

Para facilitar o trabalho com mapas e *StreetView* foi utilizado o pacote de APIs da Google na versão 10. Este pacote é uma extensão do Kit de Desenvolvimento Android e é disponibilizado pela Google com ferramentas para facilitar a integração das aplicações com os seus serviços. Foi utilizada a versão 10, pois é a versão compatível com a versão 2.3.3 do Android, esta era a mais utilizada quando se iniciou esta pesquisa.

Para realizar testes não foi necessário emulador. O ambiente de desenvolvimento que foi instalado permite que a aplicação seja compilada e já instalada diretamente em um dispositivo real através de conexão USB. Quando é compilada a aplicação, o ambiente apresenta o dispositivo que está conectado como opção para executar a aplicação, como demonstra a figura 9.

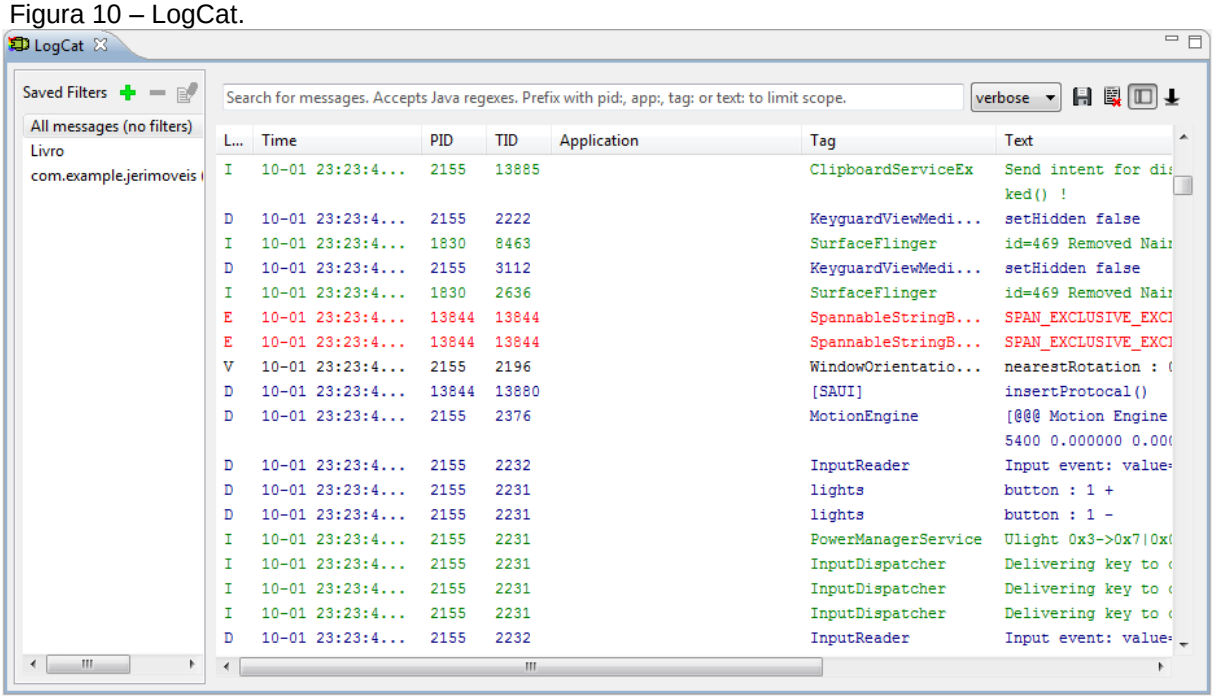
Figura 9 - Seletor de dispositivo.



Fonte: Do autor.

Para poder encontrar erros internos e “debugar” a aplicação foi utilizado a ferramenta LogCat, que tem por finalidade gerenciar todos os logs do sistema operacional e também mensagens de aplicativos. Esta ferramenta é exibida sempre que uma aplicação é compilada no ambiente que foi descrito anteriormente, e mesmo que seja executado em um dispositivo real, se manter conectado a USB, os logs são retornados. A figura 10 apresenta a tela do LogCat.

Figura 10 – LogCat.

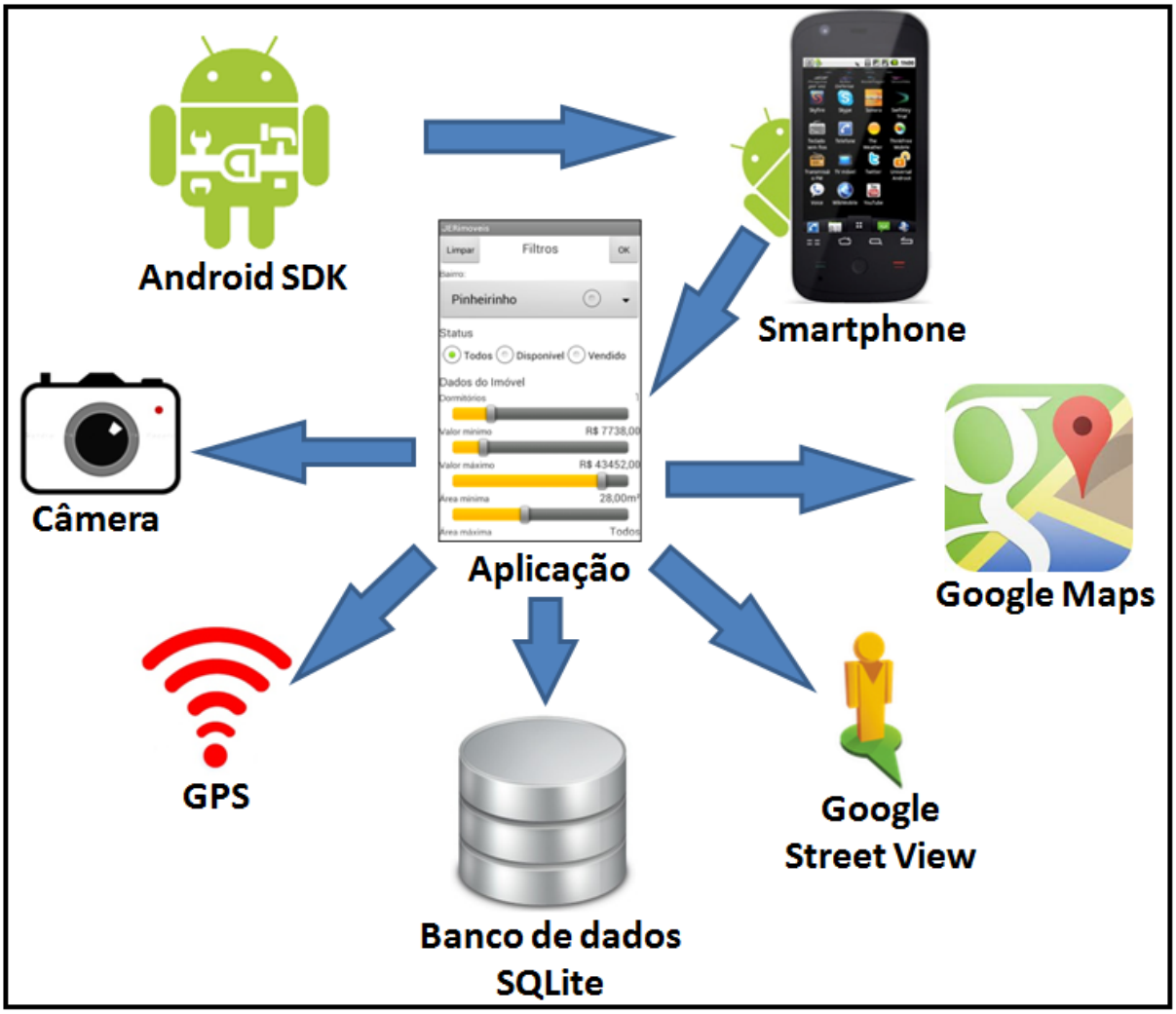


Fonte: Do autor.

6.1.2 Modelo Conceitual

Neste projeto, para desenvolvimento da aplicação móvel, foram utilizados diversas tecnologias e recursos disponibilizados pela plataforma Android. A figura 11 apresenta o fluxo de comunicação entre os recursos, com os recursos envolvidos.

Figura 11 – Fluxo de comunicação entre os recursos.



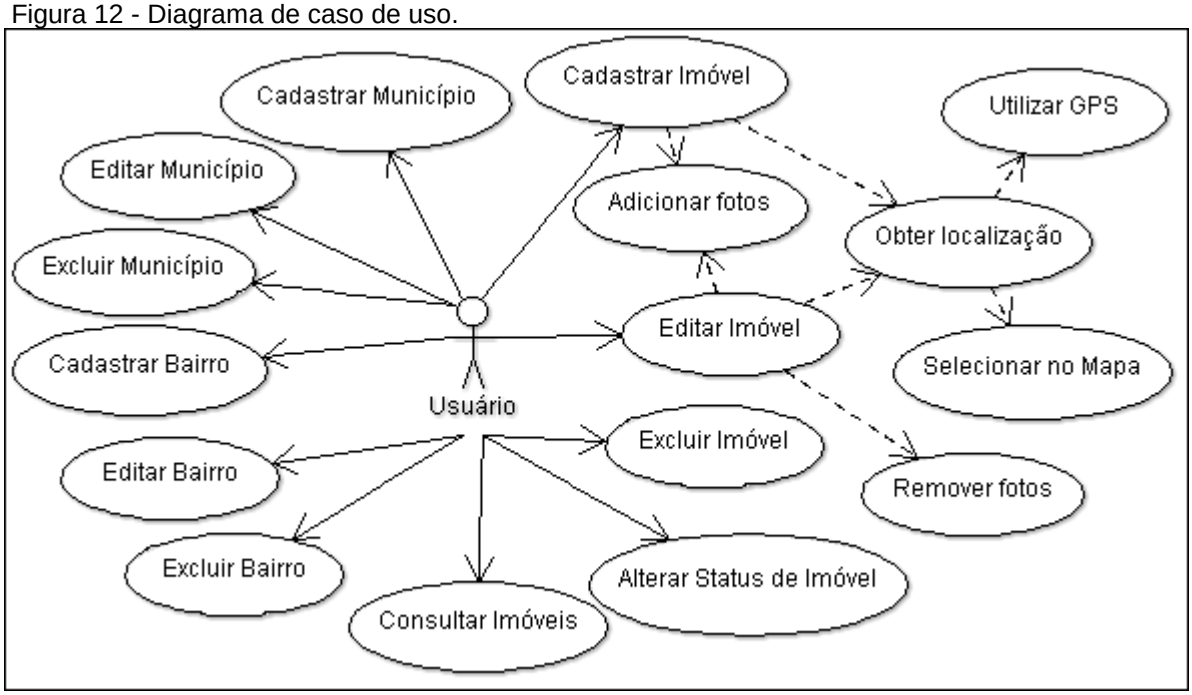
Fonte: Do autor.

Como demonstrado na figura 11, para o desenvolvimento foi necessário um *smartphone* e o SDK da plataforma. O software desenvolvido utiliza os seguintes recursos fornecidos pela plataforma android: Câmera, GPS, Banco de dados SQLite, Google Maps e Google Street View. A câmera e o GPS são utilizados durante o cadastro de imóveis, o banco de dados para o registro dos cadastros de forma relacional e consistente, e o Google Maps e Street View para apresentação detalhada da localização, ambiente, vizinhança, entre outras informações que se referem ao local do imóvel.

Na figura 12 é demonstrado o caso de uso da aplicação, onde podem ser observadas as principais funcionalidades, que são: cadastrar municípios, bairros e imóveis, adicionar fotos e coordenadas geográficas ao cadastro de imóveis e realizar

a consulta dos imóveis cadastrados. O cadastro e consulta de imóvel são as funções que utilizam os recursos de maior destaque na aplicação, que são: GPS, Câmera, Google Maps e Google Street View.

Figura 12 - Diagrama de caso de uso.



Fonte: Do autor.

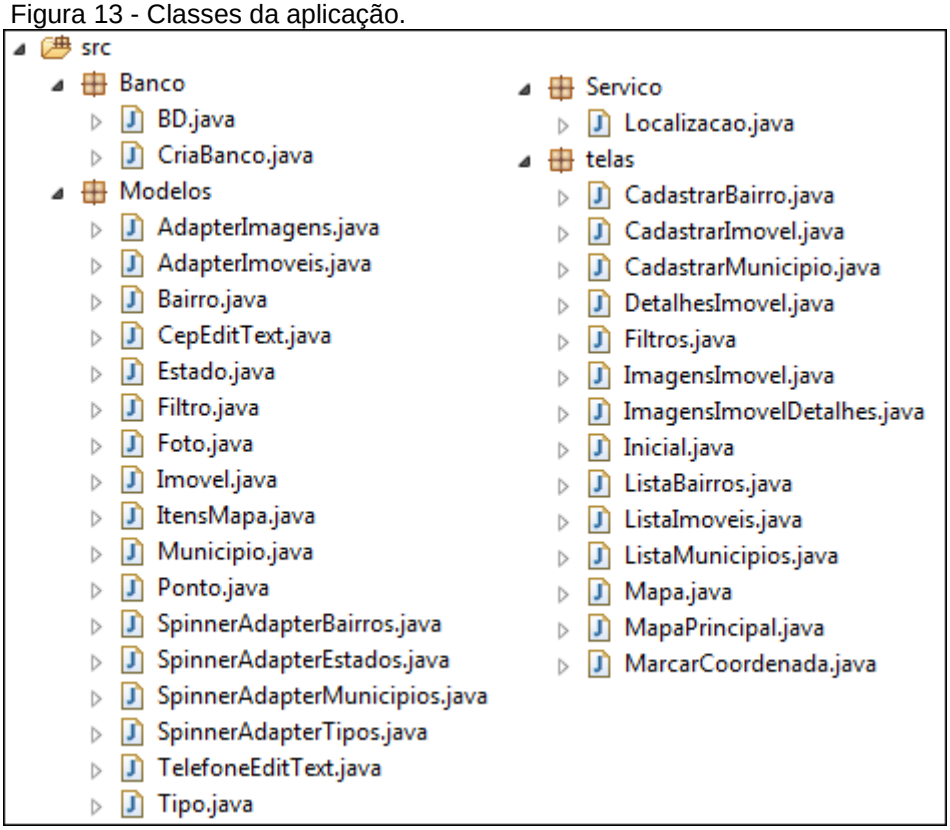
6.2 IMPLEMENTAÇÃO

Após a definição e estudo de todos os recursos e tecnologias a serem utilizados, foi iniciado o desenvolvimento da aplicação.

Para organização das classes foi seguido arquitetura Model-view-controller (MVC[1]), separando em grupos por função: modelo, visão e controle. Classes como `Imovel.java`, `Bairro.java` e `Municipio.java` foram agrupadas como *modelo,* classes que fazem as transações com o banco foram consideradas como *controle* e as classes que estendem `Activity.java`, que são responsáveis por fazer a gerência das telas da aplicação, foram consideradas como *visão.* A figura 13 ilustra as classes criadas e como foram organizadas.

[1] modelo de arquitetura de software com objetivo básico de separar lógica de negócio e apresentação.

Figura 13 - Classes da aplicação.



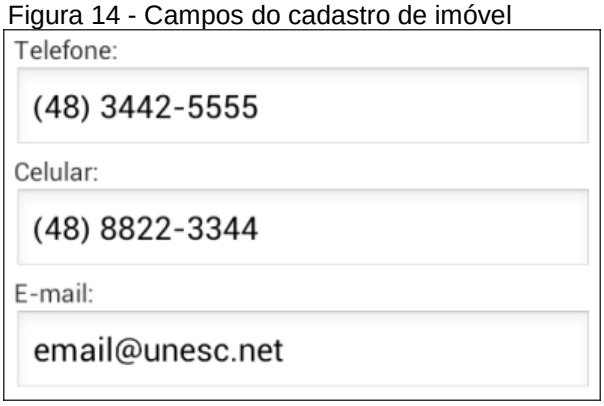
Fonte: Do autor.

O pacote *telas* contém as classes responsáveis por gerenciar cada tela da aplicação, onde a classe `CadastrarImovel.java` gerencia a tela de cadastro de imóvel, sendo uma das funcionalidades principais da aplicação. No cadastro de imóvel são inseridas as seguintes informações: tipo de imóvel, estado, município, bairro, endereço, CEP, número, quantidade de dormitórios, valor, área, telefone, celular, email, responsável, descrição e ainda a coordenada geográfica e fotos do imóvel.

Para as informações como CEP, telefone e celular, foi inserido máscara no campo, para o correto preenchimento pelo usuário. Para isto foi criado classe personalizada estendendo a classe `EditText.java`. Esta classe permite adicionar uma ação para ser executada cada vez que o campo for alterado, assim foi possível tratar a entrada do usuário e deixar os campos personalizados. Já para o campo de email, foi feito apenas um método para validação dentro da própria classe de cadastro. Este método é chamado quando o usuário tenta salvar o imóvel. A

figura 14 demonstra os campos de telefone, celular e email, preenchidos com a máscara que foi configurada.

Figura 14 - Campos do cadastro de imóvel



Fonte: Do autor.

Para obter a coordenada geográfica foram implementadas duas formas: através de provedores de localização (GPS e Internet) e através da seleção de um ponto no mapa pelo usuário. Foi utilizada a opção de seleção da localização no mapa, pois os provedores nem sempre estão disponíveis.

Como na utilização de provedores foi verificada uma demora no cálculo da coordenada, foi criado uma classe como serviço, ou seja, estende a classe `Service.java` do Android, com a intenção de ocultar do usuário o tempo de espera para efetuar o cálculo, desta forma o calculo é feito enquanto o usuário faz outras operações na aplicação.

Como a busca pela coordenada é uma das tarefas do Android que mais consome a bateria dos dispositivos, foi configurado para este serviço ser iniciado apenas quando a tela de cadastro de imóvel for aberta. Enquanto o usuário preenche as demais informações do imóvel, o serviço de localização tem o tempo de obter a coordenada.

Para obter a coordenada dos provedores foi utilizada a classe `LocationManager.java`, que após inicializada representa o serviço de localização do Android e possui métodos que permitem o controle dos provedores. Para definir o provedor a ser utilizado e identificar as localizações foi utilizado o método `requestLocationUpdates(String, int, int, LocationListener)`, onde o primeiro parâmetro é o nome do provedor a ser utilizado, o segundo é a diferença mínima de tempo para considerar uma nova coordenada, o terceiro a

diferença mínima de distância para considerar uma nova coordenada e o quarto permite definir a ação a ser realizada a cada alteração de localização. A figura 15 ilustra a utilização do provedor GPS.

Figura 15 - Obtendo localização com provedor GPS

```
locationManager = (LocationManager) mContext.getSystemService(Context.LOCATION_SERVICE);

locationListenerGps = new LocationListener() {
    public void onLocationChanged(Location location) {
        verificarLocalizacao(location);
    }
    public void onStatusChanged(String provider, int status, Bundle extras) {}
    public void onProviderEnabled(String provider) {}
    public void onProviderDisabled(String provider) {
    }
};

locationManager.requestLocationUpdates(LocationManager.GPS_PROVIDER, 0, 0, locationListenerGps);
```

Fonte: Do autor.

Como demonstrado na figura 15, a classe `LocationListener.java` possui o método `onLocationChanged(Location)`, que é acionado e recebe a localização sempre que o provedor encontra uma nova localização. Neste método foi acionada uma verificação na localização recebida, onde verifica se a precisão da nova é maior que a anterior, se for maior atualiza a coordenada que será utilizada no imóvel.

Para obter a localização por meio da seleção de um ponto no mapa, foi criado a classe `MarcarCoordenada.java`, que apresenta um mapa e tem o método `dispatchTouchEvent(MotionEvent)` sobrescrito para captura do toque sobre o mapa, para obter a localização selecionada, considerando somente quando o mapa for clicado, pois quando o mapa for movido não deve ser considerado a coordenada. Isto foi possível pela ação do evento que é recebido como parâmetro no método. Há três ações para o evento, sendo: 0 quando a tela é pressionada, 1 quando “solta” a tela e 2 quando é movido. A figura 16 apresenta o método `dispatchTouchEvent(MotionEvent)` e como foi tratado estas ações.

Figura 16 - Método para obter coordenada apartir do toque no mapa.

```
public boolean dispatchTouchEvent(MotionEvent ev) {
    int actionType = ev.getAction();
    if(actionType == 0){
        click = true;
    }else if(actionType == 1){
        if(click==true){
            Projection proj = mapView.getProjection();
            GeoPoint loc = proj.fromPixels((int)ev.getX(), (int)ev.getY());
            final Double longitude = ((double)loc.getLongitudeE6())/1000000;
            final Double latitude = ((double)loc.getLatitudeE6())/1000000;
            click = false;
            AlertDialog.Builder builder = new AlertDialog.Builder(this);
            builder.setMessage("Confirma coordenadas?\nLatitude: "+latitude+"\nLongitude: "+longitude)
                   .setCancelable(false)
                   .setPositiveButton("Sim", new DialogInterface.OnClickListener() {...
                   .setNegativeButton("Não", new DialogInterface.OnClickListener() {...
            AlertDialog alert = builder.create();
            alert.show();
        }
    }else if(actionType == 2){
        click = false;
    }
    return super.dispatchTouchEvent(ev);
}
```

Fonte: Do autor.

Mais detalhes sobre a utilização do mapa serão apresentados junto com a tela principal da aplicação.

Outra informação importante que contempla o cadastro de imóvel na aplicação são as fotos. Para captura de imagens com a câmera do dispositivo integrada, foi utilizada a aplicação nativa, assim não foi necessário implementar o controle completo, pois a aplicação nativa faz o trabalho com a câmera e retorna a imagem em formato Bitmap[2].

Para a chamada da aplicação nativa da câmera foi utilizado o código que é apresentado na figura 17, onde foi criada uma *intent* com a ação para abrir a aplicação da câmera e esta *intent* é enviada para o sistema operacional.

Figura 17 - Chamada da aplicação nativa da câmera.

```
Intent it = new Intent(MediaStore.ACTION_IMAGE_CAPTURE);
startActivityForResult(it,0);
```

Fonte: Do autor.

Foi utilizado o método `startActivityForResult()` no envio da ação para o sistema operacional, para poder receber o retorno da imagem capturada. Utilizando este método foi possível obter a imagem pelo método

[2] Mapa de bits em inglês. São imagens que contêm a descrição de cada pixel.

`onActivityResult()` que recebe o retorno de todas as ações que são chamadas pelo método `startActivityForResult()`. A figura 18 apresenta como o método foi implementado, onde a imagem que é retornada em formato Bitmap é convertida para um *array de byte* para poder ser persistida no banco de dados.

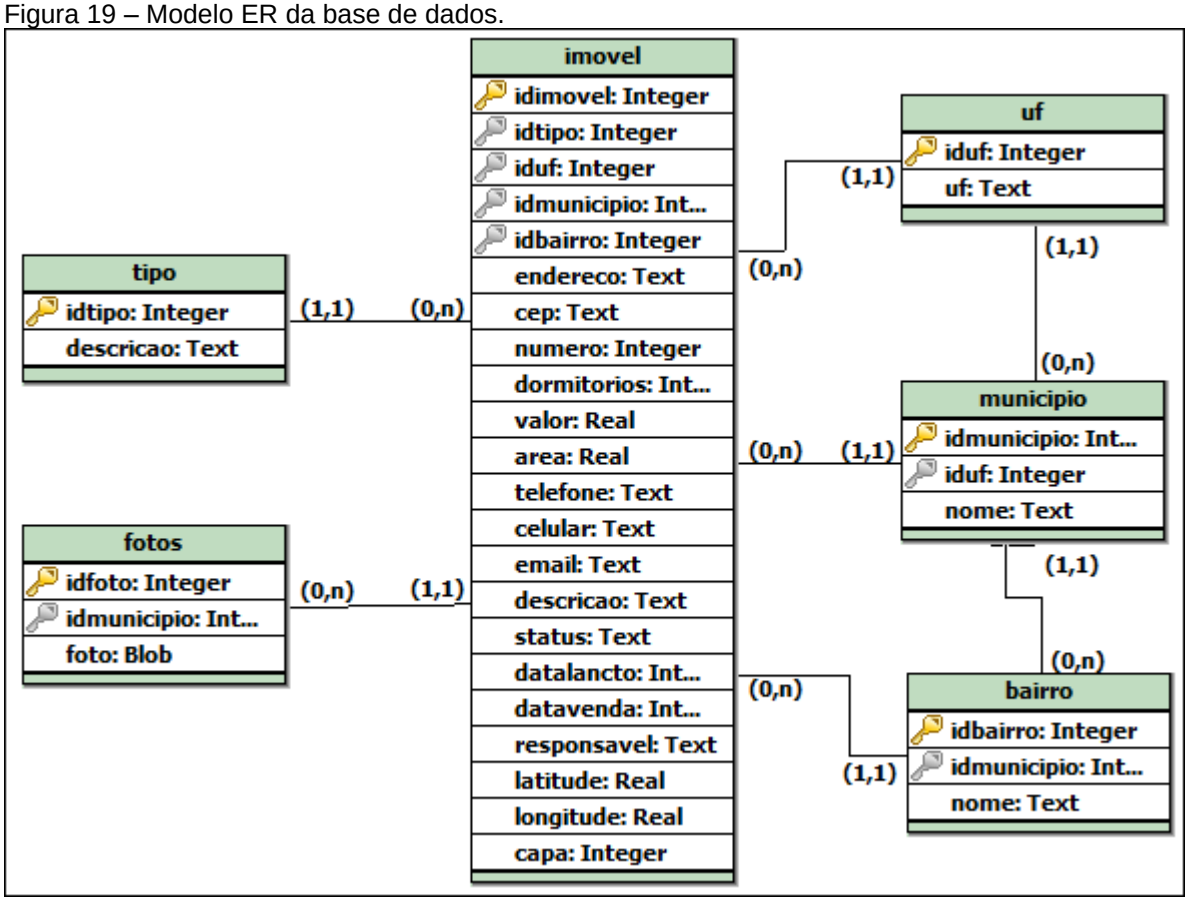
Figura 18 - Método para receber o retorno da câmera.

```
protected void onActivityResult(int requestCode, int resultCode, Intent data){
    super.onActivityResult(requestCode, resultCode, data);
    if(requestCode == 0){
        if(resultCode == -1){

            Bundle param = data.getExtras();
            Bitmap bitmap = (Bitmap) param.get("data");

            ByteArrayOutputStream stream = new ByteArrayOutputStream();
            bitmap.compress(Bitmap.CompressFormat.PNG, 100, stream);
            byte[] byteArray = stream.toByteArray();

            fotos.add(new Foto(0, byteArray, 0));
            listarFotos();
        }
    }
}
```

Fonte: Do autor.

Para a persistência destes dados de imóvel e também cadastro de tipos de imóvel, estados, municípios e bairros, foi utilizado o banco de dados SQLite, por ser nativo do Android. A figura 19 demonstra o modelo ER da base de dados criada.

Figura 19 – Modelo ER da base de dados.



Fonte: Do autor.

Para criar o banco de dados foi utilizado a API do Android, que permite a criação do banco de dados durante a execução da aplicação e, após criar o banco, pode ser executado comando SQL para criar as tabelas e preencher os dados. Foi criado a classe `CriaBanco.java` que estende a classe `SQLiteOpenHelper.java` da API, esta contém toda a lógica de criação de um banco de dados.

O construtor da classe da API recebe como parâmetro o objeto Context, o nome do banco, o objeto cursorFactory e a versão do banco. Esta classe contêm métodos que são chamados quando o banco precisa ser criado ou atualizado, no caso de nova versão. O método chamado para criar o banco foi sobrescrito para criar as tabelas e inserir os cadastros de tipos de imóveis e de estados, que já foram pré-definidos. A figura 20 ilustra este método. O método de atualização foi sobrescrito somente para excluir as tabelas e chamar o método de criação do banco para recriá-las.

Figura 20 - Métodos da classe CriaBanco.

```
public CriaBanco(Context context) {
    super(context, "JER", null, 1);
}
@Override
public void onCreate(SQLiteDatabase arg0) {
    arg0.execSQL(tableFotos);
    arg0.execSQL(tableTipo);
    arg0.execSQL(tableUf);
    arg0.execSQL(tableMunicipio);
    arg0.execSQL(tableBairro);
    arg0.execSQL(tableImovel);
    for(String estado : estados){
        ContentValues values = new ContentValues();
        values.put("uf", estado);
        arg0.insert("uf", null, values);
    }
    for(String tipo : tipos){
        ContentValues values = new ContentValues();
        values.put("descricao", tipo);
        arg0.insert("tipo", null, values);
    }
}
```

Fonte: Do autor.

A classe `BD.java` contém os métodos que fazem as inserções, alterações, exclusões e consultas no banco que foi criado. Para alterar a base de dados é necessário obter uma instância que é representada pela classe `SQLiteDatabase.java`. Esta classe contém os métodos para manipulação de base de dados. A figura 21 apresenta como foi obtido esta instância da base na classe `BD.java`.

Figura 21 - Instância da base de dados.

```
private SQLiteDatabase database;
private CriaBanco db;

public BD(Context context) {
    db = new CriaBanco(context);
    database = db.getWritableDatabase();
}
```

Fonte: Do autor.

Após todos os cadastros, os imóveis podem ser apresentados em lista ou no mapa. A classe responsável por apresentar os imóveis no mapa é a `MapaPrincipal.java`.

Para trabalhar com mapa no Android foi verificado que as classes específicas de mapas não pertencem à biblioteca de classes do Android SDK, estas pertencem à outra biblioteca que é restrita da Google. Então foi alterado nas propriedades do projeto para compilar utilizando o Google APIs, e não o SDK

padrão. O Google APIs contém todas as classes do SDK e mais as classes específicas de mapas.

Para exibir o mapa foi criado um arquivo de layout, que na plataforma Android é um XML, com a *tag* `<com.google.android.maps.MapView/>`. A classe desta tag se comunica, internamente, com os serviços do Google Maps, então foi necessário declarar permissão para acesso à internet no arquivo *AndroidManifest.xml*.

A utilização dos serviços do Google Maps exige uma chave de autenticação, esta foi inserida como atributo a classe `MapView.java` no arquivo de layout. Esta chave de autenticação foi obtida seguindo instruções contidas na obra de Ricardo Lecheta (LECHETA, 2010). A figura 22 demonstra o arquivo de layout com a chave inserida.

Figura 22 - Arquivo de layout do Mapa.

```
<?xml version="1.0" encoding="utf-8"?>
    <com.google.android.maps.MapView
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:id="@+id/mapaPrincipal"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:clickable="true"
        android:apiKey="0P4t-vSSvbGBlxndqpPD_FTP9-Xomq1N-Udiylg"
    />
```

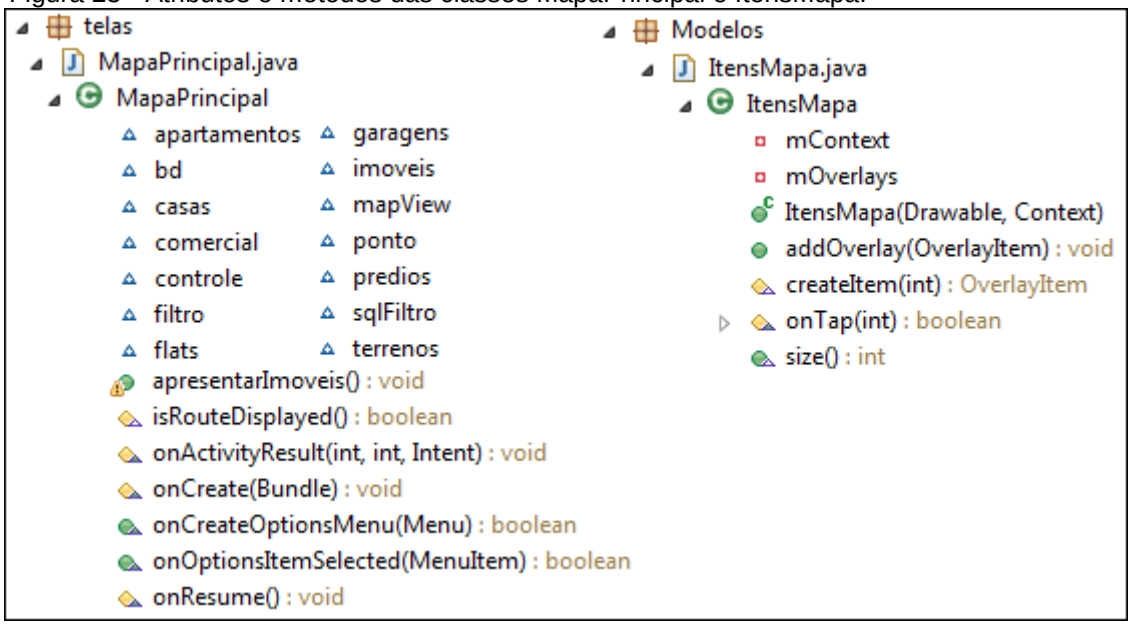
Fonte: Do autor.

Como o pacote de mapas não é padrão da plataforma Android, também foi necessário declará-lo no arquivo *AndroidManifest.xml* da aplicação. Por fim, foi declarado o arquivo de layout na classe `MapaPrincipal.java`, que estende `MapActivity.java`, pois esta contém também métodos específicos para gerenciar o mapa.

Para atribuir marcações que se referem aos imóveis no mapa, foi utilizado a classe `OverlayItem.java`, mas esta corresponde apenas a uma marcação no mapa. Então para adicionar mais de uma marcação, foi criado a classe `ItensMapa.java`, estendendo a classe `ItemizedOverlay.java` que contém uma lista de itens. Foi sobrescrito método `onTap()`, que é acionado sempre que um item é selecionado, assim foi possível executar a ação desejada. Então, por fim, foi adicionada a classe `ItensMapa.java` como atributo da classe

`MapaPrincipal.java`. A figura 23 lista os atributos e métodos das classes `MapaPrincipal.java` e `ItensMapa.java`.

Figura 23 - Atributos e métodos das classes MapaPrincipal e ItensMapa.



Fonte: Do autor.

Com o clique sobre um item do mapa, é possível recuperar o item que foi selecionado no método `onTap()`. Neste método foi configurada uma tela de diálogo com o usuário, que dá a opção de visualizar a imagem da rua do imóvel utilizando o serviço do Google StreetView. Isto é possível, pois cada item possui sua coordenada geográfica, e isto é o necessário para passar por parâmetro para o serviço do StreetView. A figura 24 demonstra o código utilizado no método `onTap()` quando o usuário optar por exibir a rua do imóvel, onde primeiro é criado uma *Intent* e após iniciado a mesma.

Figura 24 - Utilização do serviço Google StreetView

```
Intent streetView = new Intent(android.content.Intent.ACTION_VIEW,Uri.parse(
"google.streetview:cbll="+
(item.getPoint().getLatitudeE6()/1E6) +","+
(item.getPoint().getLongitudeE6()/1E6) +
"&cbp=1,99.56,,1,-5.27&mz=21"));

mContext.startActivity(streetView);
```

Fonte: Do autor.

Estes foram os pontos “chave” da implementação, que fizeram a ligação entre a aplicação e os recursos disponibilizados pelo Android.

6.2.1 Funcionamento do software

A aplicação foi desenvolvida com intuito de facilitar a apresentação de imóveis pelo corretor. Então a tela principal da aplicação é o mapa com marcações que referenciam os imóveis cadastrados, de acordo com a localização, facilitando a visualização dos imóveis cadastros. Como foram definidos tipos de imóveis, as marcações no mapa são diferenciadas para cada tipo. A figura 25 apresenta o mapa com as marcações. Ao clicar sobre uma marcação, é exibida uma tela de diálogo com informações básicas do imóvel, dando opção ao usuário de visualizar detalhes ou abrir o StreetView na localização do imóvel. A figura 26 demonstra esta tela de diálogo.

Figura 25 - Mapa principal com as marcações.

Fonte: Do autor.

Figura 26 - Tela de diálogo.

Fonte: Do autor.

Na primeira utilização da aplicação não há nenhuma marcação, pois não há imóveis cadastrados. Assim a primeira ação a ser feita quando instalado a aplicação, são os cadastros. Os tipos de imóveis e os estados foram pré-definidos e são inseridos na criação do banco de dados. Os tipos de imóvel são: apartamento, casa, terreno, comercial, prédio, flats e vaga de garagem. E os estados são todos os brasileiros. Então o cadastro deve ser realizado na seguinte sequência: município, bairro e imóvel. A tela principal contém o *menu* padrão do Android que permite o acesso aos cadastro. Conforme demonstra a figura 27, o *menu* tem quatro opções: Municípios, Bairros, Imóveis e Filtrar.

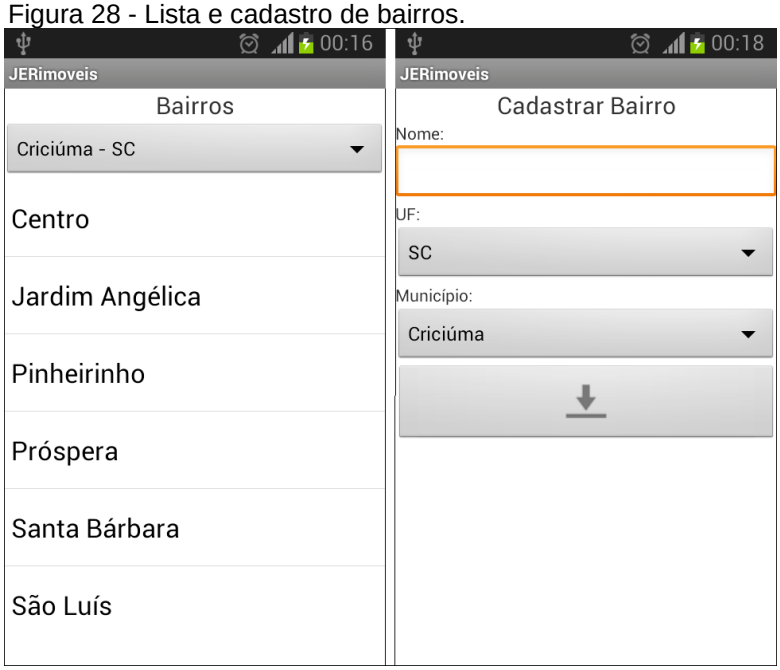
Figura 27 - Menu da tela principal.

Fonte: Do autor.

O *menu* Municípios dá acesso a lista de municípios. Nesta tela, além da visualização, é possível editar e remover, clicando na lista, e cadastrar novo município, através do *menu*. Na tela de cadastro pode ser informado o nome do município e selecionar o estado ao qual pertence.

O tratamento de bairros é semelhante. O *menu* Bairros dá acesso a lista de bairros. Nesta tela, além da visualização, é possível editar e remover, clicando na lista, e cadastrar novo bairro, através do *menu*. Na tela de cadastro pode ser informado o nome do bairro, selecionar o estado e município ao qual pertence. A figura 28 apresenta a tela que lista os bairros a esquerda e a tela de cadastro de bairros a direita, que são semelhantes às telas referentes a municípios.

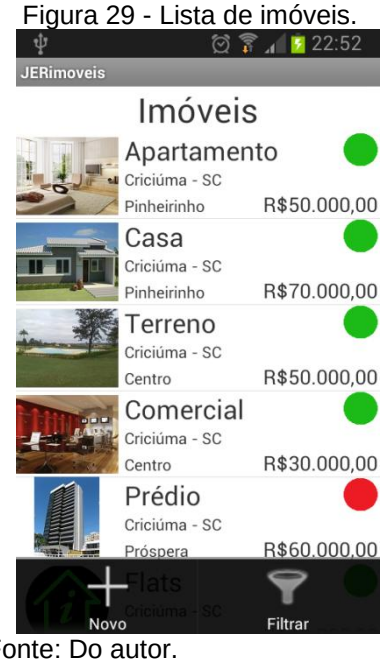
Figura 28 - Lista e cadastro de bairros.



Fonte: Do autor.

O *menu* Imóveis dá acesso a lista de imóveis, onde cada item da lista contêm a foto de capa e informações básicas do imóvel. Esta tela de lista tem mais funções que as listas de municípios e bairros. Clicando sobre um imóvel na lista, o usuário tem quatro opções: visualizar detalhes, apresentar imóvel no mapa, editar e excluir. Através do menu pode ser cadastrado novo imóvel e também realizar filtro na lista, que é o mesmo filtro acessado pela tela principal. A figura 29 demonstra esta tela com os imóveis e o menu.

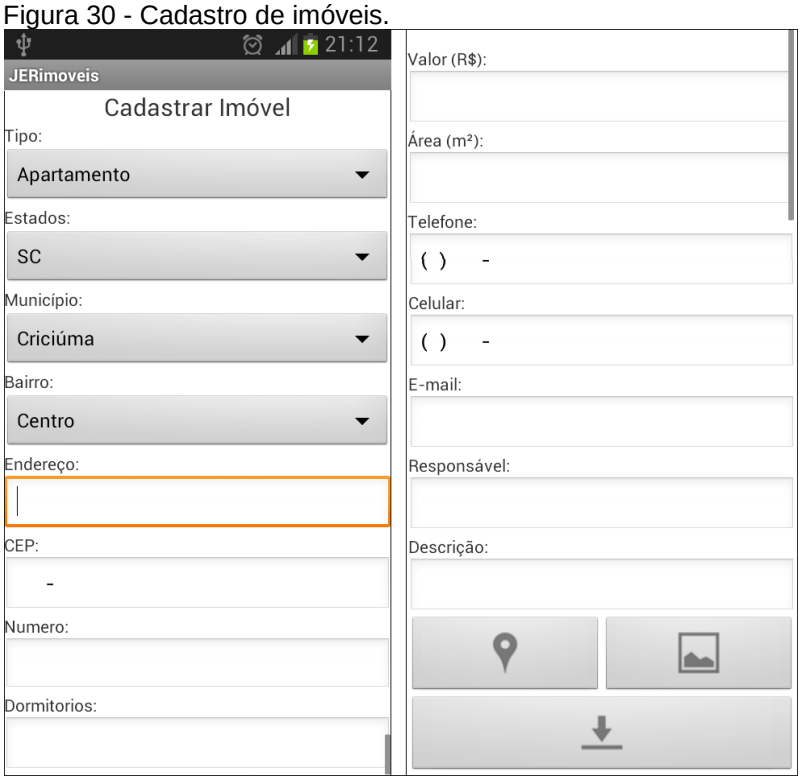
Figura 29 - Lista de imóveis.



Fonte: Do autor.

Na tela de cadastro de imóveis é informado o tipo do imóvel, estado, município bairro, endereço, CEP, número na rua, quantidade de dormitórios, valor, área, telefone, celular, email, responsável e descrição, e ainda pode ser informada a localização geográfica e inserido fotos. Estes foram considerados dados importantes para a venda de um imóvel de acordo com os requisitos levantados na fase inicial do projeto, principalmente a localização geográfica e fotos, que permitem uma visão real do estado do imóvel. A figura 30 exibe a tela de cadastro em duas partes.

Figura 30 - Cadastro de imóveis.



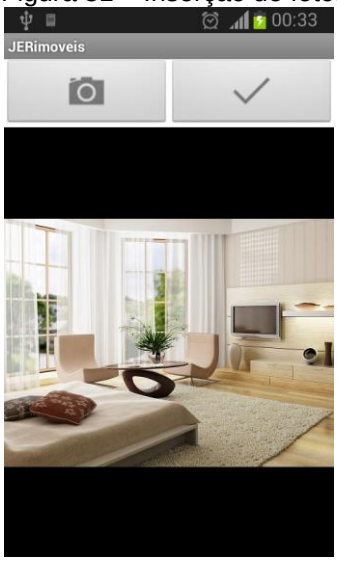
Fonte: Do autor.

Para informar a localização do imóvel é apresentado duas opções através de tela de diálogo, sendo: localização atual obtida pelos provedores ou seleção no mapa. Como há um serviço, para obter a localização dos provedores, sendo executado durante o cadastro do imóvel, este sinaliza na tela de cadastro de imóvel quando há localização fornecida pelos provedores, para induzir o usuário a utilizar a localização dos provedores, já que é mais prático e pode ser até mais preciso que a seleção no mapa.

A tela de cadastro também dá acesso a tela de cadastro e gerencia das fotos do imóvel. Esta tela exibe a lista de fotos e permite a interação com a aplicação

nativa da câmera para captura da foto. Nesta tela ainda é possível excluir ou definir uma foto como capa. A figura 31 apresenta esta tela de inserção de fotos.

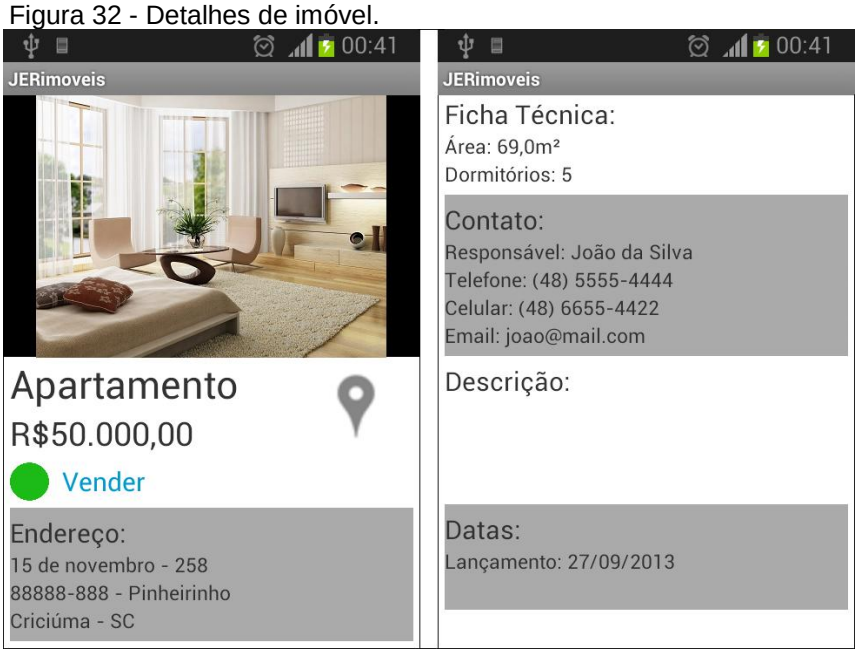
Figura 31 – Inserção de fotos.



Fonte: Do autor.

Todos os detalhes inseridos no cadastro do imóvel podem ser visualizados na tela de detalhes. Esta pode ser acessada, como mencionado anteriormente, através das marcações do mapa principal ou da lista de imóveis. Nesta tela, além de visualizar os detalhes do imóvel, pode ser alterado o status entre *vendido* e *disponível*. A figura 32 apresenta a tela de detalhes em duas partes.

Figura 32 - Detalhes de imóvel.



Fonte: Do autor.

Como demonstra a figura 32, a parte superior da tela de detalhes apresenta as fotos, o tipo de imóvel, o valor e dá acesso a visualização do imóvel no mapa, considerando estes dados como informações relevantes para o corretor, dando uma visibilidade geral sobre o imóvel.

Por fim, o quarto menu da tela principal dá acesso a tela de filtros. Esta permite localizar os imóveis, utilizando como filtro os seguintes dados: tipo, estado, município, bairro, status, quantidade de dormitórios, valor mínimo, valor máximo, área mínima e área máxima. Este mesmo filtro pode ser acessado pelo menu da tela com a lista de imóveis. A figura 33 apresenta a parte inferior da tela de filtros.

Figura 33 - Filtros.

Fonte: Do autor.

Estas são as funcionalidades da aplicação desenvolvida, que engloba a utilização de todos os recursos informados na proposta do trabalho.

6.3 RESULTADOS OBTIDOS

A pesquisa feita neste trabalho resultou em uma aplicação robusta utilizando a plataforma Android, com diversas tecnologias e recursos envolvidos.

A aplicação é destinada ao gerenciamento de imóveis por profissionais da área, com telas intuitivas e práticas, utilizando recursos variados da plataforma Android, como mapas, GPS, banco de dados e câmera, com objetivo de dar mais praticidade à aplicação. Com os recursos utilizados é possível trabalhar com informações indispensáveis para a venda de um imóvel, como localização e fotos.

A facilidade de apresentação de imóveis com o mapa é uma das funções mais importantes, mas a aplicação se destaca das demais existentes no mercado, como Moving Imóveis e ZAP Imóveis, pela possibilidade de cadastro, com localização e fotos, e a utilização do Google Street View (MOVING IMÓVEIS, [2013?]; ZAP IMÓVEIS, [2013?]).

Como conclusão desta etapa, todos os objetivos propostos foram atingidos, deixando uma aplicação de código livre e robusta, disponível aos acadêmicos interessados tanto em sua utilização, como em melhorias e também como base para novos projetos.

7 CONCLUSÃO

Foram efetuados estudos que proporcionaram o entendimento sobre a plataforma Android, características e aspectos funcionais. Foi possível identificar a facilidade no desenvolvimento com esta plataforma, isto devido a grande quantidade de documentação e APIs que estão disponíveis.

A grande quantidade de APIs também permite o desenvolvimento de aplicações robustas, que utilizem recursos de alto nível como apresentado no desenvolvimento deste trabalho. Esta plataforma incentiva novos desenvolvedores, por sua facilidade de desenvolvimento e integração com diversos recursos.

Além destas vantagens, ainda é de código aberto e possui grande quantidade de dispositivos compatíveis de diferentes fabricantes, pois a plataforma Android foi desenvolvida por uma aliança de empresas, sendo a maioria de telefonia, como Samsung e LG.

Utilizando esta plataforma, o objetivo geral e os objetivos específicos deste trabalho foram atingidos, resultando em uma aplicação robusta com diversas tecnologias e recursos envolvidos, voltada para o cadastro e apresentação de imóveis. A aplicação faz a utilização de recursos que a plataforma Android permite a integração com o uso de APIs, como mapas, GPS, banco de dados e câmera.

Foram encontradas algumas dificuldades, mas os materiais utilizados para estudo do desenvolvimento Android proporcionaram um entendimento prático e funcional das ferramentas e recursos utilizados. As APIs permitiram a utilização dos recursos com facilidade, como obter a localização com o GPS. A possibilidade de integração com as aplicações nativas da plataforma facilitou a captura de imagens, pois foi utilizada a aplicação nativa da câmera, ficando para a aplicação desenvolvida apenas o trabalho de receber a imagem.

A partir dos resultados obtidos nesta pesquisa, bem como as tecnologias abordadas, ficam como sugestões para trabalhos futuros:

a) Integração com aplicação de gestão de imobiliária;

b) tornar a aplicação multiplataforma;

c) aprofundar a aplicação de acordo com as necessidades do corretor;

d) utilização da API versão 3 do Google Maps;

e) implementar o cadastro automático do endereço do imóvel de acordo com a coordenada informada;

f) sugerir a próxima visita em imóvel próximo;

g) melhorar o layout e compatibilidade para diferentes dimensões de telas.

REFERÊNCIAS

ABLESON, W. Frank; SEN, Robi; KING, Chris. **Android in Action.** Second Edition [S. l.]: Manning Publications, 2011.

BENDER, Carlos Roberto. **G-SMS: protótipo de aplicação de envio de sms georeferenciadas.** 2011. 53 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Universidade Regional de Blumenau, Blumenau, 2011. Disponível em: <http://campeche.inf.furb.br/tccs/2011-I/TCC2011-1-10-VF-CarlosRBender.pdf>. Acesso em: 11 mai. 2013.

BRASIL. Conselho Federal de Corretores de Imóveis. Lei nº 6.530 de 12 de maio de 1978.

BURNETTE, Ed. **Hello, Android:** Introducing Google's Mobile Development Platform. 2nd edition [S. l.]: Pragmatic Bookshelf, 2009.

CINTRA, Marcos. Consumo, inflação e contas externas. **CRECISP,** São Paulo, ano 04, n. 07, p. 46-47, 2011. Disponível em: <http://www.crecisp.gov.br/revista/#>. Acesso em: 14 abr. 2013.

COMIN, Alisson Tomaz. **Promessa de compra e venda de imóvel e a resolução pela cláusula de arrependimento.** 2003. 101 f. Trabalho de Conclusão de Curso (Pós-Graduação em Administração de Empresas) – Universidade do Extremo Sul Catarinense, Criciúma, 2003.

CRECISP. Imóveis só com corretor. **CRECISP**, São Paulo, ano 04, n. 08, p. 37-39, 2011. Disponível em: <http://www.crecisp.gov.br/revista/#>. Acesso em: 14 abr. 2013.

_____. Jovens ingressam na carreira de corretores de imóveis. **CRECISP**, São Paulo, ano 05, n. 09, p. 20-22, 2012. Disponível em: <http://www.crecisp.gov.br/revista/#>. Acesso em: 14 abr. 2013.

_____. **Perfil do Corretor.** [2013?]. Disponível em: <http://www.crecisp.gov.br/perfil_do_corretor.asp>. Acesso em: 01 abr. 2013.

DIMARZIO, J. F.; POLO, G. L. **Android, A Programmer's Guide.** [S. l.]: McGraw-Hill Osborne Media, 2008.

GOOGLE INC.. **Google Play.** [2013?]a. Disponível em: < https://play.google.com/> Acesso em: 01 abr. 2013.

_____. **Dashboards**. [2013?]b. Disponível em: <http://developer.android.com/about/dashboards/index.html >. Acesso em: 15 abr. 2013.

_____. **Bem-vindo ao Google Maps.** [2013?]c. Disponível em: <http://support.google.com/maps/bin/answer.py?hl=pt-BR&topic=1687350&answer=144352> Acesso em: 15 abr. 2013.

_____. **Por dentro do Google Maps.** [2013?]d. Disponível em: <http://support.google.com/maps/bin/answer.py?hl=pt-BR&answer=144349&topic=1687350&ctx=topic> Acesso em: 16 abr. 2013.

_____. **Onde o Street View está disponível?** [2013?]e. Disponível em: <http://maps.google.com.br/intl/pt-BR/help/maps/streetview> Acesso em: 16 abr. 2013.

_____. **Intents List: Invoking Google Applications on Android Devices.** [2013?]f. Disponível em: <http://developer.android.com/guide/appendix/g-app-intents.html> Acesso em: 17 abr. 2013.

_____. **Camera.** [2013?]g. Disponível em: <http://developer.android.com/guide/topics/media/camera.html> Acesso em: 16 abr. 2013.

GUPTA, Rohit Kumar. **Mobile Real Estate Agent for Android.** 2011. 67 p. Dissertação (Mestrado em Ciência da Computação) - San Diego State University, San Diego, 2011. Disponível em: <http://sdsu-dspace.calstate.edu/bitstream/handle/10211.10/1712/Gupta_Rohit.pdf>. Acesso em: 11 mai. 2013.

HASHIMI, Sayed; KOMATINENI, Satya; MACLEAN, Dave. **Pro Android 2.** [S. l.]: Apress, 2010.

LECHETA, Ricardo R. **Google Android:** aprenda a criar aplicações para dispositivos móveis com o Android SDK. 2 ed. São Paulo: Novatec, 2010.

LINDENBERG FILHO, Sylvio. **Guia prático do corretor de imóveis:** fundamentos e técnicas. São Paulo: Atlas, 2006. 179 p.

MALLICK, Martyn. **Mobile and Wireless Design Essentials.** New Jersey: John Wiley & Sons, 2003.

SILVA, William Bertan da. **Interfaceamento entre dispositivos móveis e jogos eletrônicos de computadores utilizando o acelerômetro baseado no Sistema Operacional Android.** 2012. 96 f. Trabalho de Conclusão de Curso (Graduação em

Ciência da Computação) – Universidade do extremo sul catarinense, Criciúma, 2012.

MARTINS, Rafael. **Desenvolvimento de Aplicativo para Smartphone com a Plataforma Android.** 2009. 50 f. Trabalho de Conclusão de Curso (Graduação em Ciência da Computação) - Pontifícia Universidade Católica do Rio de Janeiro, Rio de Janeiro, 2009. Disponível em: <http://www.icad.puc-rio.br/~projetos/android/files/monografia.pdf>. Acesso em: 9 mai. 2013.

MEIER, Reto. **Professional Android™ 2 Application Development.** New York: Wiley Publishing, 2010.

MOVING IMÒVEIS. **Moving Imóveis.** [2013?]. Disponível em: < https://play.google.com/store/apps/details?id=br.com.moving&hl=pt_BR > Acesso em: 18 out. 2013.

PAES, Milton. **Número de corretores de imóveis aumenta.** DCI, São Paulo, 02 jun. 2011. Disponível em: <http://www.dci.com.br/cidades/numero-de-corretores-de-imoveis-aumenta-id257076.html>. Acesso em: 01 abr. 2013.

RAPOSO, Alexandre Tinel; HEINE, Cláudio B. **Manual jurídico do corretor de imóveis.** 7. ed. Rio de Janeiro: Ímã Publicidade, 2004. 846 p.

SQLITE Org. **Datatypes In SQLite Version 3.** [2013?]. Disponível em: <http://www.sqlite.org/datatype3.html> Acesso em: 18 abr. 2013.

STEELE, James; TO, Nelson. **The Android Developer's Cookbook:** Building Applications with the Android SDK. New York: Addison-Wesley Professional, 2010.

TELECO. **Sistemas Operacionais.** [2012?]. Disponível em: < http://www.teleco.com.br/sist_operacional.asp > Acesso em: 08 out. 2012.

XU, Guochang. **GPS:** Theory, Algorithms and Applications. 2. ed. Berlin: Springer, 2007.

ZAP IMÓVEIS. **ZAP Imóveis Mobile.** [2013?]. Disponível em: < http://www.zap.com.br/imoveis/mobile > Acesso em: 18 out. 2013.

APÊNDICE A – ARTIGO CIENTÍFICO

Software Android para Corretores de Imóveis

Rafael Silva de Bitencourt[1], Gilberto Vieira da Silva[1]

[1]Curso de Ciência da Computação – Universidade do Extremo Sul Catarinense (UNESC)
Criciúma – SC – Brasil

rafael_silbit@hotmail.com,gilbertovieirasilva@hotmail.com

Abstract. *This paper main objetive is to develop an mobile aplication, using android platform to help realtors to manage properties and to introduce them to customers. Providing a brief understanding about the emerging Android platform, its main features and also as the integration of new aplications with available resources in the platform, as: camera, location providers, SQLite database, Google Maps and Street View services. Through this study it is expected that innovations by social and scientific communities might be created and widespreaded with present and future Technologies.*

Resumo. *Este artigo tem como objetivo desenvolver uma aplicação móvel, utilizando a plataforma Android, para auxiliar corretores de imóveis na gerência de imóveis e na apresentação à clientes. Proporcionando um breve entendimento sobre a utilização da plataforma emergente Android, suas principais características como também a integração de novas aplicações com recursos disponíveis na plataforma, como: câmera, provedores de localização, banco de dados SQLite, serviço Google Maps e Google StreetView. Através deste estudo espera-se que possam ser criadas e difundidas inovações pelas comunidades social e científica, com as tecnologias presentes e futuras.*

1. Introdução

O mercado imobiliário está apresentando um crescimento no Brasil e fatores que têm forte influência são o aumento do crédito para pessoas físicas e das faixas salariais, que dão às famílias um maior poder aquisitivo [CINTRA, 2011; LINDENBERG FILHO, 2006].

Este crescimento desperta o interesse de investidores e também pela profissão de corretor de imóveis. Até mesmo profissionais de outras áreas estão se especializando para trabalharem como corretor de imóveis, devido às boas perspectivas de lucros que este mercado vem oferecendo [CRECISP, 2012; PAES, 2011].

O aumento de profissionais na área aumenta também a concorrência. Esta competitividade leva o cliente a escolher o corretor melhor preparado. Então, corretores menos experientes devem buscar métodos que sejam um diferencial para poderem entrar no mercado. Hoje a utilização de tecnologia na apresentação dos imóveis pode ser um grande diferencial [LINDENBERG FILHO, 2006].

Conforme buscas em sites como Google Play [GOOGLE INC., (2013?)a], hoje há aplicativos móveis de empresas que permitem que clientes possam realizar consultas de imóveis em seu banco de dados, porém nenhum destes aplicativos é para uso específico do corretor, impossibilitando que ele gerencie seu cadastro de imóveis e apresente a seus clientes.

Aplicações móveis englobam novas tecnologias e conceitos que algumas empresas ainda não conhecem, sendo que uma aplicação móvel pode trazer diversos benefícios para

qualquer negócio, como: diferenciar-se de empresa concorrente, eficiência na produtividade e mobilidade no negócio [MALLICK, 2003].

O Android conta com diversos recursos e aplicações nativas, como o Google Maps. Além disso, programadores de todo o mundo podem contribuir com o desenvolvimento desta tecnologia criando novas funcionalidades.

Outro ponto positivo é a possibilidade de desenvolver aplicações utilizando a linguagem Java e seus recursos, sendo Java uma linguagem de programação bem difundida no mercado [LECHETA, 2010].

2. Mercado Imobiliário

A expansão do mercado imobiliário brasileiro atrai investidores que querem aproveitar a valorização, que vem sendo constante, neste segmento. Profissionais de outras áreas também estão se especializando para trabalharem como corretor de imóveis, a fim de desfrutar dos benefícios desta evolução [PAES, 2011].

O aumento no número de corretores de imóveis é percebido pelo número de inscritos no CRECI. Em 2011 o CRECISP já possuía mais 114 mil inscritos. Em 2009 foram 9152 novos profissionais e em 2010 foram 9835 [CRECISP, 2011].

Como a aquisição de um imóvel é um investimento financeiramente alto, os clientes que desejam comprar procuram o maior número de informações possíveis, com objetivo de obter segurança no negócio. Estas informações podem ser encontradas nos diferentes meios de comunicação social ou internet, apresentando diversas alternativas. Desta forma, o cliente acaba recorrendo ao corretor de imóveis para intermediar a aquisição devido ao número excessivo de informações que pode deixá-lo confuso [LINDENBERG FILHO, 2006].

Segundo Brasil (1978, p. 1), a Lei n° 6.530 que disciplina o exercício da profissão diz que, "compete ao corretor exercer a intermediação na compra, venda, permuta e locação de imóveis, podendo, ainda, opinar quanto à comercialização imobiliária".

Deve-se ter atenção no contato entre o corretor e o cliente, pois o marketing imobiliário consiste basicamente no relacionamento com os clientes. Então, se bem trabalhado este contato, será um diferencial que gerará mais clientes fiéis e satisfeitos à empresa ou profissional [COMIN, 2003].

Segundo o presidente do CRECISP, José Augusto Viana Neto, hoje os clientes estão mais exigentes, cobrando informações detalhadas do imóvel que desejam adquirir, conforme apresentado na tabela 1. Assim o corretor de imóveis precisa estar muito mais preparado, tanto educacionalmente como tecnologicamente, para poder atender com agilidade a questionamentos de seus clientes [CRECISP, 2012].

Tabela 4 - Critérios de avaliação.

Critérios de avaliação mais usados no processo decisório	
Área do terreno	Distribuição interna
Ano de construção	Acabamento
Localização	lnfra-estrutura
Preço	Despesas de condomínio
Metragens	Serviços disponíveis
Número de dormitórios	Número de unidades
Orientação solar	Número de garagens

Destes critérios, o que tem maior influência é a localização do imóvel [LINDENBERG FILHO, 2006].

Com a competitividade nota-se a necessidade de os corretores se diferenciarem para poderem se destacar na profissão. Com o ingresso de profissionais cada vez mais jovens nesta área e com maior conhecimento tecnológico, a aplicação móvel pode ser utilizada como meio para apresentar seu produto de forma diferenciada.

Assim este artigo apresenta o Sistema Operacional Android como alternativa para encapsular um aplicativo de alto nível que pode auxiliar os corretores na efetivação das vendas dos imóveis.

3. Sistema Operacional Android

O Android é uma plataforma que está revolucionando o mercado global de telefonia celular. É a primeira plataforma open source para aplicação móvel que influenciou os principais mercados de telefonia móvel de todo o mundo [ABLESON; SEM; KING, 2011, tradução nossa].

A chegada do Sistema Operacional Android encantou os consumidores, mas a grande diferença ocorreu para os desenvolvedores, que antes não tinham muitas alternativas para o desenvolvimento móvel, pois os sistemas operacionais existentes eram proprietários e restringiam aplicativos de terceiros. Com o Android o desenvolvedor tem liberdade para implementar suas aplicações que podem utilizar recursos cada vez mais poderosos do hardware móvel [MEIER, 2010, tradução nossa].

3.1. Arquitetura

O OS Android foi baseado no kernel do Linux, e sobre ele estão as bibliotecas que dão suporte ao framework. As aplicações são desenvolvidas em Java e são executadas com a máquina virtual Dalvik, que é otimizada para execução em dispositivos móveis [LECHETA, 2010].

Meier [2010, tradução nossa] apresenta a arquitetura do Android, conforme figura 1.

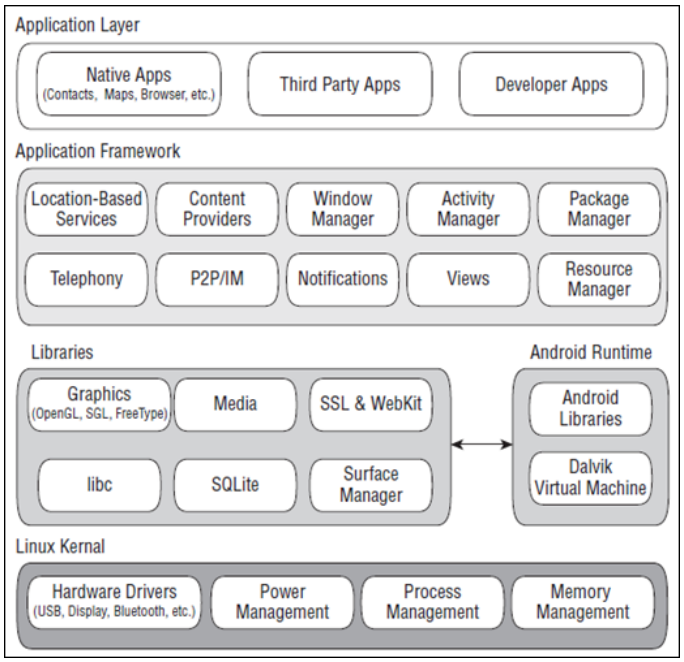


Figura 1 - Arquitetura do Sistema Android.

4. Recursos do Android

Alguns recursos e API da plataforma Android foram importantes para o desenvolvimento do trabalho, como: câmera, provedores de localização, banco de dados SQLite, serviço Google Maps e Google StreetView.

4.1. Google Maps

O Google Maps é um serviço de mapas que permite visualizar mapas básicos ou personalizados e informações sobre empresas locais, incluindo as respectivas localizações, sendo possível visualizar imagens de satélite, ampliando, reduzindo ou movimentando. Este serviço pode ser utilizado de várias formas, como por exemplo, por meio de uma página da Web com o Google Maps incorporado ou com um dispositivo móvel [GOOGLE INC., (2013?)b].

A Google mantém APIs para permitir a sincronização de dados dos seus serviços com o aplicativo Android. Entre estas APIs destaca-se a desenvolvida para o Google Maps que permite a visualização e o controle de mapas com facilidade [HASHIMI; KOMATINENI; MACLEAN, 2010].

4.2. Street View

O Street View permite explorar lugares por meio de imagens em 360 graus no nível da rua. Foi lançado em maio de 2007 com imagens de cinco cidades dos Estados Unidos e hoje já possui imagens dos sete continentes. A figura 2 apresenta destacado em azul os locais que as imagens do Street View estão disponíveis [GOOGLE INC., (2013?)c].

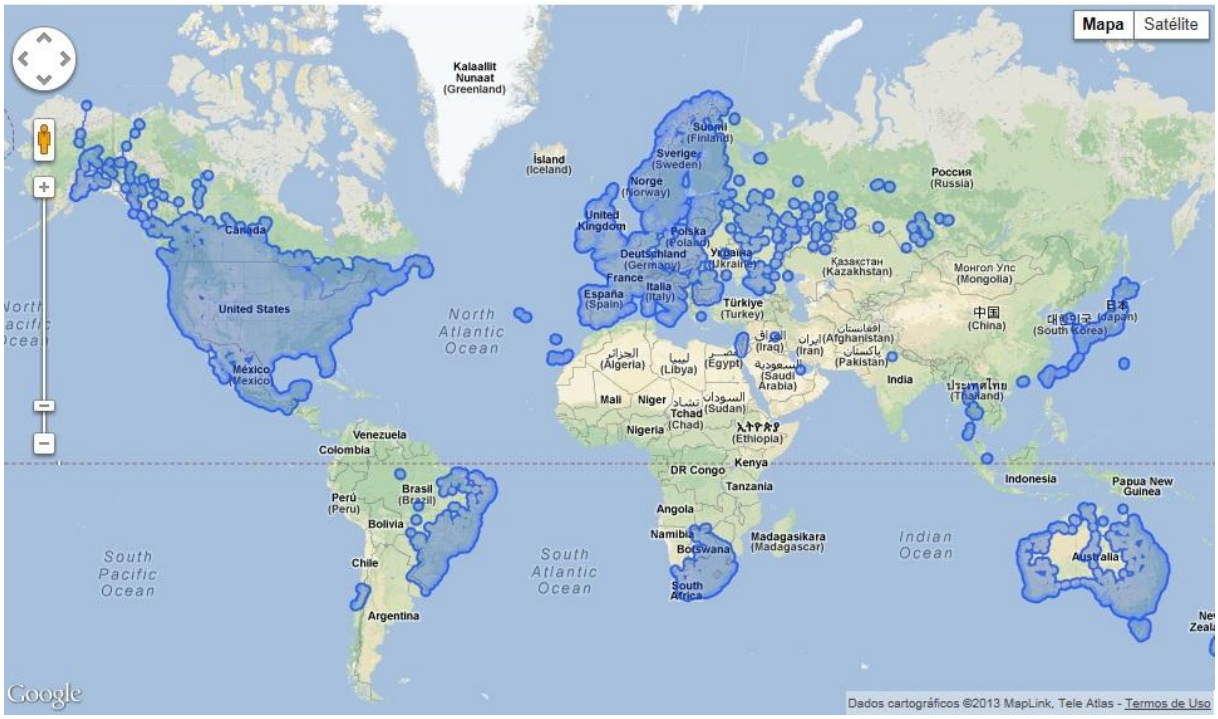


Figura 2. Imagens do Street View no mundo.

4.3. Provedores de localização

O Android possui API de localização de fácil utilização. Com ela pode ser calculado a localização do dispositivo e após exibir em um mapa utilizando o serviço do Google Maps [ABLESON; SEM; KING, 2011, tradução nossa; DIMARZIO; POLO, 2008, tradução nossa].

Além disso, uma classe fornece acesso aos serviços do sistema. Esta permite obter na aplicação a ultima localização reconhecida do provedor por meio de um de seus métodos. Neste método deve ser passado como parâmetro o provedor a ser utilizado, que pode ser: provedor por GPS e provedor por rede.

Esta API facilita a localização geográfica do dispositivo por meio de diferentes provedores. Este recurso será utilizado neste trabalho no cadastro de imóveis, por esta ser uma das mais importantes informações para venda de um imóvel.

4.4. Câmera

Para utilizar a câmera de um dispositivo o Android possui API que permite manipular através da aplicação, mas também é possível fazer a integração de aplicações. Desta forma, ao invés de utilizar a API e ter o trabalho de manipular a câmera pela aplicação, pode-se chamar a aplicação nativa da câmera para fazer a captura da imagem e retornar um objeto Bitmap, que representa uma imagem no Android [LECHETA, 2010].

4.5. Banco de dados SQLite

O Android inclui o banco de dados SQLite, que pertence a lista de bibliotecas nativas. Este é um dos bancos de dados mais implantados em todo o mundo, sendo utilizado pelo Firefox e iPhone da Apple, e a sua popularidade se explica por ser gratuito, pequeno e não necessitar de nenhuma configuração ou administração [BURNETTE, 2009, tradução nossa].

O SQLite armazena dados do tipo null, integer, real, text e blob. Blob é o armazenamento de um objeto binário, então permite o armazenamento de qualquer objeto que é convertido para este formato. Imagem é um exemplo de objeto que pode ser convertido para o formato binário e pode ser armazenado no SQLite. Como neste trabalho serão armazenadas também imagens obtidas através da aplicação nativa da câmera, será possível salvar estas imagens também no banco de dados [SQLITE, [2013?]].

Existem três formas para criação do banco de dados SQLite com Android, sendo uma delas com a própria API do Android. Esta é a forma recomendada, pois a própria aplicação pode criar o banco de dados, o que torna mais prático. Então esta será a forma utilizada neste trabalho [LECHETA, 2010].

5. Metodologia

Após ter realizado o estudo do mercado imobiliário, a função do corretor de imóveis e as tecnologias envolvidas, foram definidas e desenvolvidas as funcionalidades na aplicação para o corretor de imóveis.

Para o desenvolvimento foi utilizado a IDE Eclipse com o plugin Android Development Tools (ADT) na versão 21.0.0 para o desenvolvimento da aplicação. Este plugin amplia os recursos do Eclipse para o desenvolvimento de projetos para Android, adicionando os pacotes de API Android com o SDK. Utilizado também o pacote de APIs da Google para o trabalho com seus serviços.

Para realizar testes não foi necessário emulador. O ambiente de desenvolvimento que foi instalado permite que a aplicação seja compilada e já instalada diretamente em um dispositivo real através de conexão USB. Quando é compilada a aplicação, o ambiente apresenta o dispositivo que está conectado como opção para executar a aplicação. Então foi utilizado também o dispositivo móvel com sistema operacional Android na versão 4.1.2.

5.1. Modelo Conceitual

O projeto envolve em sua estrutura tecnologias e softwares diversos. A figura 3 apresenta o fluxo de comunicação entre os recursos, com os recursos envolvidos.

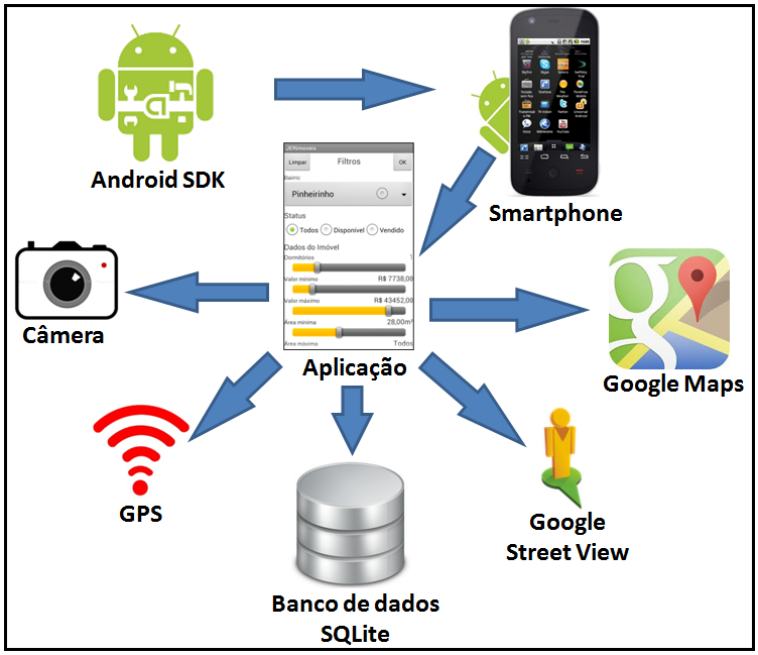


Figura 3 – Fluxo de comunicação entre os recursos.

Como demonstrado na figura 3, para o desenvolvimento foi necessário um smartphone e o SDK da plataforma. O software desenvolvido utiliza os seguintes recursos fornecidos pela plataforma android: Câmera, GPS, Banco de dados SQLite, Google Maps e Google Street View. A câmera e o GPS são utilizados durante o cadastro de imóveis, o banco de dados para o registro dos cadastros de forma relacional e consistente, e o Google Maps e Street View para apresentação detalhada da localização, ambiente, vizinhança, entre outras informações que se referem ao local do imóvel.

Na figura 4 é demonstrado o caso de uso da aplicação, onde podem ser observadas as principais funcionalidades, que são: cadastrar municípios, bairros e imóveis, adicionar fotos e coordenadas geográficas ao cadastro de imóveis e realizar a consulta dos imóveis cadastrados. O cadastro e consulta de imóvel são as funções que utilizam os recursos de maior destaque na aplicação, que são: GPS, Câmera, Google Maps e Google Street View.

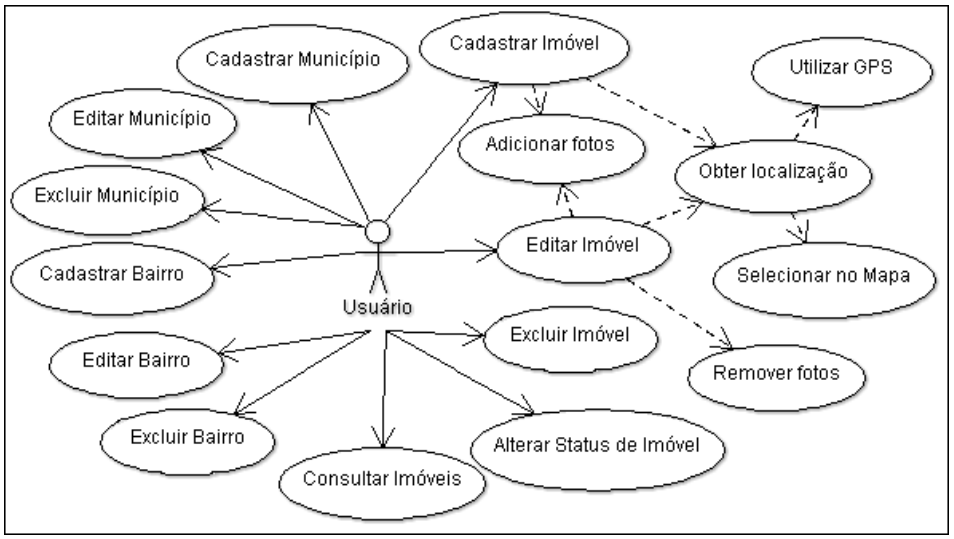


Figura 4 - Diagrama de caso de uso.

6. Implementação

Após a definição e estudo de todos os recursos e tecnologias a serem utilizados, foi iniciado o desenvolvimento da aplicação.

Para organização das classes foi seguido arquitetura Model-view-controller (MVC), separando em grupos por função: modelo, visão e controle. Classes como Imovel.java, Bairro.java e Municipio.java foram agrupadas como modelo, classes que fazem as transações com o banco foram consideradas como controle e as classes que estendem Activity.java, que são responsáveis por fazer a gerência das telas da aplicação, foram consideradas como visão. A figura 5 ilustra as classes criadas e como foram organizadas. A figura 6 ilustra a tela principal da aplicação, sendo o mapa com as marcações de imóveis.

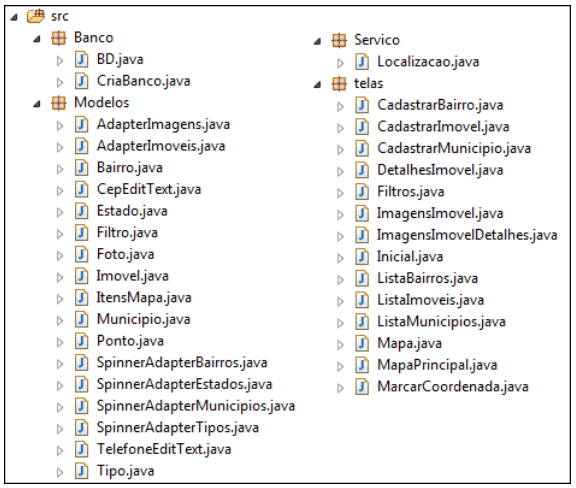


Figura 5 - Classes da aplicação.

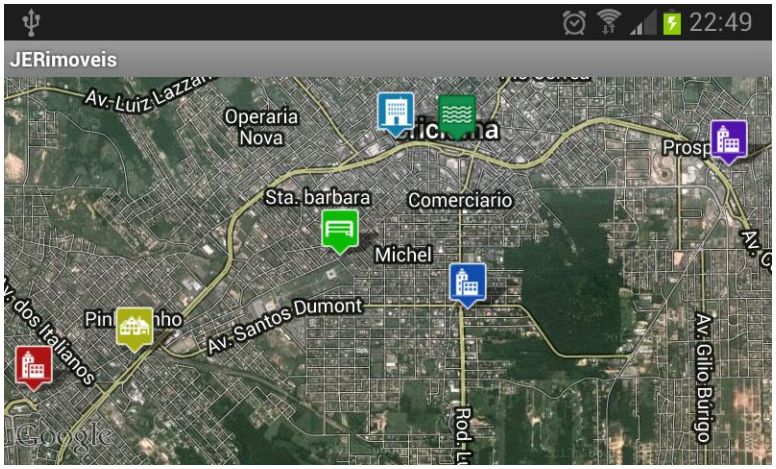


Figura 6 - Classes da aplicação.

6.1 Funcionamento do software

A aplicação foi desenvolvida com intuito de facilitar a apresentação de imóveis pelo corretor. Então a tela principal da aplicação é o mapa com marcações que referenciam os imóveis cadastrados, de acordo com a localização, facilitando a visualização dos imóveis cadastros.

O menu dá acesso as lista e cadastro de municípios, bairros e imóveis, e também a tela de filtros, que permite filtrar os imóveis que devem ser apresentados no mapa de acordo com suas características cadastrais.

Na tela de cadastro de imóveis é informado o tipo do imóvel, estado, município bairro, endereço, CEP, número na rua, quantidade de dormitórios, valor, área, telefone, celular, email, responsável e descrição, e ainda pode ser informada a localização geográfica e inserido fotos. Estes foram considerados dados importantes para a venda de um imóvel de acordo com os requisitos levantados na fase inicial do projeto, principalmente a localização geográfica e fotos, que permitem uma visão real do estado do imóvel.

A lista de imóveis apresenta dados básicos do imóvel, como: tipo, endereço, valor, status e foto definida como capa, tendo a opção de apresentar os detalhes, editar, excluir, alternar o status entre vendido e disponível e apresentar no mapa.

Além da apresentação dos detalhes do imóvel, da apresentação no mapa, a aplicação permite a demonstração da rua do imóvel com a utilização do serviço Google Street View.

7. Resultados Obtidos

A pesquisa feita neste trabalho resultou em uma aplicação robusta utilizando a plataforma Android, com diversas tecnologias e recursos envolvidos.

A aplicação é destinada ao gerenciamento de imóveis por profissionais da área, com telas intuitivas e práticas, utilizando recursos variados da plataforma Android, como mapas, GPS, banco de dados e câmera, com objetivo de dar mais praticidade à aplicação. Com os recursos utilizados é possível trabalhar com informações indispensáveis para a venda de um imóvel, como localização e fotos.

A facilidade de apresentação de imóveis com o mapa é uma das funções mais importantes, mas a aplicação se destaca das demais existentes no mercado, como Moving Imóveis e ZAP Imóveis, pela possibilidade de cadastro, com localização e fotos, e a utilização do Google Street View [MOVING IMÓVEIS, (2013?); ZAP IMÓVEIS, (2013?)].

8. Conclusão

Foram efetuados estudos que proporcionaram o entendimento sobre a plataforma Android, características e aspectos funcionais. Foi possível identificar a facilidade no desenvolvimento com esta plataforma, isto devido a grande quantidade de documentação e APIs que estão disponíveis.

A grande quantidade de APIs também permite o desenvolvimento de aplicações robustas, que utilizem recursos de alto nível como apresentado no desenvolvimento deste trabalho. Esta plataforma incentiva novos desenvolvedores, por sua facilidade de desenvolvimento e integração com diversos recursos.

Além destas vantagens, ainda é de código aberto e possui grande quantidade de dispositivos compatíveis de diferentes fabricantes, pois a plataforma Android foi desenvolvida por uma aliança de empresas, sendo a maioria de telefonia, como Samsung e LG.

Utilizando esta plataforma, o objetivo geral e os objetivos específicos deste trabalho foram atingidos, resultando em uma aplicação robusta com diversas tecnologias e recursos envolvidos, voltada para o cadastro e apresentação de imóveis. A aplicação faz a utilização de recursos que a plataforma Android permite a integração com o uso de APIs, como mapas, GPS, banco de dados e câmera.

Foram encontradas algumas dificuldades, mas os materiais utilizados para estudo do desenvolvimento Android proporcionaram um entendimento prático e funcional das ferramentas e recursos utilizados. As APIs permitiram a utilização dos recursos com facilidade, como obter a localização com o GPS. A possibilidade de integração com as

aplicações nativas da plataforma facilitou a captura de imagens, pois foi utilizada a aplicação nativa da câmera, ficando para a aplicação desenvolvida apenas o trabalho de receber a imagem.

Referências

ABLESON, W. Frank; SEN, Robi; KING, Chris. Android in Action. Second Edition [S. l.]: Manning Publications, 2011.

BRASIL. Conselho Federal de Corretores de Imóveis. Lei nº 6.530 de 12 de maio de 1978.

BURNETTE, Ed. Hello, Android: Introducing Google's Mobile Development Platform. 2nd edition [S. l.]: Pragmatic Bookshelf, 2009.

CINTRA, Marcos. Consumo, inflação e contas externas. CRECISP, São Paulo, ano 04, n. 07, p. 46-47, 2011. Disponível em: <http://www.crecisp.gov.br/revista/#>. Acesso em: 14 abr. 2013.

COMIN, Alisson Tomaz. Promessa de compra e venda de imóvel e a resolução pela cláusula de arrependimento. 2003. 101 f. Trabalho de Conclusão de Curso (Pós-Graduação em Administração de Empresas) – Universidade do Extremo Sul Catarinense, Criciúma, 2003.

CRECISP. Imóveis só com corretor. CRECISP, São Paulo, ano 04, n. 08, p. 37-39, 2011. Disponível em: <http://www.crecisp.gov.br/revista/#>. Acesso em: 14 abr. 2013.

_____. Jovens ingressam na carreira de corretores de imóveis. CRECISP, São Paulo, ano 05, n. 09, p. 20-22, 2012. Disponível em: <http://www.crecisp.gov.br/revista/#>. Acesso em: 14 abr. 2013.

DIMARZIO, J. F.; POLO, G. L. Android, A Programmer's Guide. [S. l.]: McGraw-Hill Osborne Media, 2008.

GOOGLE INC.. Google Play. [2013?]a. Disponível em: < https://play.google.com/> Acesso em: 01 abr. 2013.

_____. Bem-vindo ao Google Maps. [2013?]b. Disponível em: < http://support.google.com/maps/bin/answer.py?hl=pt-BR&topic=1687350&answer=144352> Acesso em: 15 abr. 2013.

_____. Onde o Street View está disponível? [2013?]c. Disponível em: <http://maps.google.com.br/intl/pt-BR/help/maps/streetview> Acesso em: 16 abr. 2013.

HASHIMI, Sayed; KOMATINENI, Satya; MACLEAN, Dave. Pro Android 2. [S. l.]: Apress, 2010.

LECHETA, Ricardo R. Google Android: aprenda a criar aplicações para dispositivos móveis com o Android SDK. 2 ed. São Paulo: Novatec, 2010.

LINDENBERG FILHO, Sylvio. Guia prático do corretor de imóveis: fundamentos e técnicas. São Paulo: Atlas, 2006. 179 p.

MALLICK, Martyn. Mobile and Wireless Design Essentials. New Jersey: John Wiley & Sons, 2003.

MEIER, Reto. Professional Android™ 2 Application Development. New York: Wiley Publishing, 2010.

MOVING IMÒVEIS. Moving Imóveis. [2013?]. Disponível em: < https://play.google.com/store/apps/details?id=br.com.moving&hl=pt_BR > Acesso em: 18 out. 2013.

PAES, Milton. Número de corretores de imóveis aumenta. DCI, São Paulo, 02 jun. 2011. Disponível em: <http://www.dci.com.br/cidades/numero-de-corretores-de-imoveis-aumenta-id257076.html>. Acesso em: 01 abr. 2013.

SQLITE Org. Datatypes In SQLite Version 3. [2013?]. Disponível em: <http://www.sqlite.org/datatype3.html> Acesso em: 18 abr. 2013.

ZAP IMÓVEIS. ZAP Imóveis Mobile. [2013?]. Disponível em: < http://www.zap.com.br/imoveis/mobile > Acesso em: 18 out. 2013.