UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ

COORDENAÇÃO DO CURSO DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CURSO SUPERIOR DE TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

CEDULIO CEZAR E SILVA

DAVID AUGUSTO MARCELINO VEIGA

GERENCIADOR WEB-MÓVEL IMOBILIÁRIO UTILIZANDO ANDROID E QR CODE

TRABALHO DE DIPLOMAÇÃO

PONTA GROSSA

2012

CEDULIO CEZAR E SILVA

DAVID AUGUSTO MARCELINO VEIGA

GERENCIADOR WEB-MÓVEL IMOBILIÁRIO UTILIZANDO ANDROID E QR CODE

Trabalho de conclusão de curso de graduação, apresentado à disciplina Trabalho de Diplomação, do curso Superior de Tecnologia em Análise e Desenvolvimento de Sistemas da Coordenação do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas – COADS – como requisito parcial à obtenção do título Tecnólogo na Universidade Tecnológica Federal do Paraná – UTFPR, Câmpus Ponta Grossa.

Orientador: Prof. Danillo Belmonte.

PONTA GROSSA

2012

TERMO DE APROVAÇÃO

GERENCIADOR WEB-MÓVEL IMOBILIÁRIO UTILIZANDO ANDROID E QR CODE

por

CEDULIO CEZAR E SILVA
DAVID AUGUSTO MARCELINO VEIGA

Este Trabalho de Conclusão de Curso foi apresentado em cinco de junho de 2012 como requisito parcial para a obtenção do título de Tecnólogo em Tecnologia em Análise e Desenvolvimento de Sistemas. Os candidatos foram arguidos pela Banca Examinadora composta pelos professores abaixo assinados. Após deliberação, a Banca Examinadora considerou o trabalho aprovado.

_____
Danillo Leal Belmonte
Prof. Orientador

_____
Simone de Almeida
Membro titular

_____
Thalita Scharr Rodrigues
Membro titular

AGRADECIMENTOS

Gostaríamos de agradecer primeiramente ao nosso Orientador Danillo Belmonte, que com paciência nos orientou com grande competência durante o período de desenvolvimento deste trabalho.

Agradecemos a nossas famílias e amigos pelo apoio nos momentos difíceis e compreensão durante os momentos em que ficamos ausentes.

RESUMO

SILVA, Cedulio C. e VEIGA, David Augusto M. **GERENCIADOR WEB-MÓVEL IMOBILIÁRIO UTILIZANDO ANDROID E QR CODE**. 2012. 59 páginas. Trabalho de Conclusão de Curso - Universidade Tecnológica Federal do Paraná. Ponta Grossa, 2012.

Este trabalho mostra como tema principal o desenvolvimento de duas aplicações utilizando a metodologia *eXtreming Programming* para o ramo imobiliário, sendo uma voltada para dispositivos móveis que utilizam o Sistema Operacional Android. A função da aplicação móvel é ler QR Codes e apresentar dados referente ao imóvel solicitado para os usuários de forma simples e rápida. A outra aplicação é voltada para a *web* e tem por finalidade o cadastro de imóveis, de usuários do sistema, além de alimentar a aplicação móvel através de arquivos XML. Para desenvolver tais aplicações foram utilizadas as seguintes tecnologias: Android, API Google Maps, biblioteca ZXing, QR Code, JavaEE, XML e MVC. A regra de negócio utilizada nesse sistema mostra-se promissora em cidades como, por exemplo, aquelas com um número elevado de universitários, que o tempo de permanência dos locatários nos imóveis é menor, causando assim uma grande movimentação na procura de imóveis para aluguel.

Palavras-chave: Android. QR Code. XML. *Google Maps*. JavaEE.

ABSTRACT

SILVA, Cedulio C. e VEIGA, David Augusto M. **GERENCIADOR WEB-MÓVEL IMOBILIÁRIO UTILIZANDO ANDROID E QR CODE**. 2012. 59 pages. Final Paper- Federal Technology University - Parana. Ponta Grossa, 2012

This work shows how the main theme the development of two applications using the methodology eXtreming Programming for the real estate, one focused on mobile devices using the Android operating system. The function of the mobile application is to read QR Codes and present data relating to real estate required for users to quickly and easily. The other application is directed to the web and aims to register the properties, users of the system, besides feeding the mobile application via XML files. To develop such applications were used the following technologies: Android, Google Maps API, library ZXing, QR Code, JavaEE, XML and MVC. The business rule used in this system shows promise in cities such as, for example, those with a high number of students, the residence time of the tenants in buildings is smaller, thus causing a significant movement in search of properties for rent.

Keywords: Android. QR Code. XML. *Google Maps*. JavaEE.

LISTA DE FIGURAS

LISTA DE GRÁFICOS

LISTA DE QUADROS

SUMÁRIO

1 INTRODUÇÃO

Recentemente houve um crescimento exponencial de um novo Sistema Operacional (SO) *mobile* que fornece uma gama de possibilidades ao ramo empresarial. Tal SO é o Android. O sucesso do Android mostra-se devido ao baixo custo dos aparelhos que o utilizam e o tipo da licença do sistema, que é livre e de código aberto. Assim cada fabricante de *smartphones* pode modificar o sistema de acordo com as necessidades e capacidades de seu *hardware*. Outros fatores que ocasionaram o sucesso deste SO são a variedade de aplicativos e a quantidade de aplicativos gratuitos disponíveis para *download* no Android *Market*.

Visando promover o ramo imobiliário que também cresce a largos passos, uma estratégia é oferecer serviços adicionais que utilizam tecnologias móveis e *web*, criando assim uma vantagem sobre empresas que não se atualizam e que estarão perdendo oportunidades de negócios. Afinal os consumidores procuram cada vez mais facilidades, a fim de descobrir se o produto realmente atende a sua necessidade e é viável financeiramente.

O método comum de anúncio de imóveis é através de jornais ou placas no próprio imóvel. Logo, o cliente ficava à mercê de estar interessado por algo inviável ou a uma visita frustrada ao ver uma casa, apartamento que não correspondesse à sua expectativa. Com o surgimento da internet, foi possível além de ver fotos e características, pesquisar de acordo com a necessidade e vontade. Porém, mesmo com todas as melhorias que a internet trouxe o usuário ainda está atrelado a uma regra de negócio que o impossibilita de visualizar as informações sobre o imóvel que ele deseja de uma forma rápida e prática, pois para visualizar as informações desejadas precisaria de um computador a sua disposição com acesso a internet.

1.1 JUSTIFICATIVA

Com o avanço da tecnologia *mobile* é possível colocar toda a facilidade que a internet trouxe no bolso do usuário e unir mais vantagens e facilidades.

Uma delas é a utilização do Sistema de Posicionamento Global (GPS), presente em muitos *smartphones*, onde o usuário poderá visualizar a localização correta dos imóveis. E para agilizar o processo de pesquisa o usuário poderá utilizar-se dos QR Codes que a imobiliária estará disponibilizando no local do imóvel. Depois, com a ajuda do *smartphone* que possui o sistema Android, o usuário poderá interpretar o QR Code trazendo todas as informações disponíveis.

Desta forma, ao invés do usuário precisar chegar em casa ou ir até a imobiliária, estará obtendo as informações no local. Porém, para que tudo isso seja possível, um sistema *web* é necessário e deve desempenhar as seguintes funções: processamento das requisições vindas da aplicação móvel, armazenamento de dados como, por exemplo, preço e fotos e gerar o QR Code específico de cada imóvel.

1.2 OBJETIVO GERAL

Aperfeiçoar o processo de busca de informações sobre imóveis disponíveis para aluguel ou venda, fornecendo informações disponíveis para o usuário instantaneamente no local do imóvel, diminuindo assim o tempo que seria gasto para ir até a imobiliária responsável ou então a necessidade de ligar e aguardar até ser atendido para receber as informações desejadas.

1.3 OBJETIVOS ESPECÍFICOS

Para alcançar o objetivo geral se faz necessário o desenvolvimento de duas aplicações, uma para *web* e outra voltada para dispositivos móveis.

1.3.1 Aplicação *Web*

O sistema *web* basicamente supre a base de dados, gera QR Codes e atende a requisições de imóveis vindas da aplicação móvel. No entanto, se um usuário desejar visualizar as informações de algum imóvel através de um

navegador também será possível a URL gerada e armazenada nos QR Codes direcionam para páginas de apresentação dos imóveis.

Para atender a esses requisitos, os seguintes objetivos específicos (abaixo relacionados) são necessários, e tais módulos foram identificados:

- Desenvolver sistema controlador;
- Desenvolver sistema de autenticação;
- Desenvolver sistema gerenciador de usuários;
- Desenvolver sistema gerenciador de operações;
- Desenvolver sistema gerenciador de tipos;
- Desenvolver sistema gerenciador de características;
- Desenvolver sistema gerenciador de imóveis;
- Desenvolver sistema gerador de QR Codes;
- Desenvolver sistema gerador de arquivos no formato XML;
- Desenvolver sistema de visualização para *web*.

1.3.2 Aplicação Móvel

A aplicação móvel interpreta o QR Code fornecido pelo sistema *web* que contém um endereço. Com a URL ele efetua uma requisição à aplicação móvel via HTTP, processa o XML que é retornado e mostra as informações como, por exemplo, fotos e valor do imóvel. Também é possível visualizar o endereço do imóvel no Google Maps e adicionar imóveis visualizados como favoritos para visualização posterior.

Os seguintes objetivos específicos a seguir relacionados são necessários para concluir a operação (módulos identificados):

- Desenvolver sistema de leitura de QR Codes;
- Desenvolver sistema de requisição HTTP;
- Desenvolver sistema de interpretação de arquivos XML;
- Desenvolver sistema de apresentação de Imóvel;
- Desenvolver sistema de gerenciamento de favoritos;
- Desenvolver sistema de geolocalização.
- Desenvolver sistema de endereços não visitados.

1.4 ORGANIZAÇÃO DO TRABALHO

Este trabalho está dividido em cinco capítulos. O segundo capítulo aprofunda as ferramentas e conceitos utilizados no desenvolvimento.

No capítulo seguinte a abordagem é sobre a metodologia utilizada neste trabalho.

O quarto capítulo apresenta o desenvolvimento das aplicações, com diagramas e protótipos, bem como explicações sobre seu funcionamento.

O último capítulo apresenta o encerramento do trabalho com uma conclusão sobre as dificuldades encontradas durante o desenvolvimento e mostra oportunidades de trabalhos futuros.

2 REFERENCIAL TEÓRICO

Neste Capítulo serão apresentados as ferramentas e tecnologias utilizadas para desenvolver o trabalho.

2.1 TECNOLOGIAS

2.1.1 Orientação a Objetos

O conceito de orientação a objetos remete aos anos 60, criada como uma alternativa ao tradicional paradigma Estruturado, que foi mais utilizado durante a história da informática. As primeiras linguagens a utilizarem conceitos de orientação a objetos foram a Simula 67 e Smalltalk(MARTIN e ODELL, 1995).

O pilar principal do paradigma são os objetos, que representam a abstração de um objeto do mundo real, trazido para o mundo computacional e possui mais fundamentos como Martin e Odell (1995) discorrem que a orientação a objetos é uma abstração dos elementos do mundo real para o mundo computacional.

Com a orientação a objetos são definidas classes, as quais são um molde de um objeto do mundo real, e são definidos seus atributos e comportamentos, também assim possibilitando o reuso dos mesmos.

2.1.2 UML

A UML (*Unified Modelling Language*) surgiu após a necessidade de unir os mais importantes métodos de análise orientada a objetos, os principais métodos unificados foram: Booch, Rumbaugh e Jacobson, porém a UML se trata somente de uma linguagem de modelagem, e não todo o processo de análise, como no caso é o *Rational Unified Process*(RUP) também se utilizando dos principais processos de Booch, Rumbaugh e Jacobson.

A linguagem traz todo um conjunto de diagramas necessários para a modelagem de *software* que auxiliam no desenvolvimento e também na

manutenção do código. A versão corrente em 2011 da UML é a 2.3, estando a versão 2.4 em fase beta. A UML possui algumas categorias de diagrama, um exemplo são os diagramas de caso de uso que demonstram a interação entre usuário e sistema, proposto por Jacobson, existem os Diagramas de Caso de uso(Use Case) e diagramas de caso de uso de negócios(Business Use Case), que são geralmente os primeiros diagramas a serem feitos por demonstrarem o cenário que o sistema estará presente, como demonstrado na Figura 1.

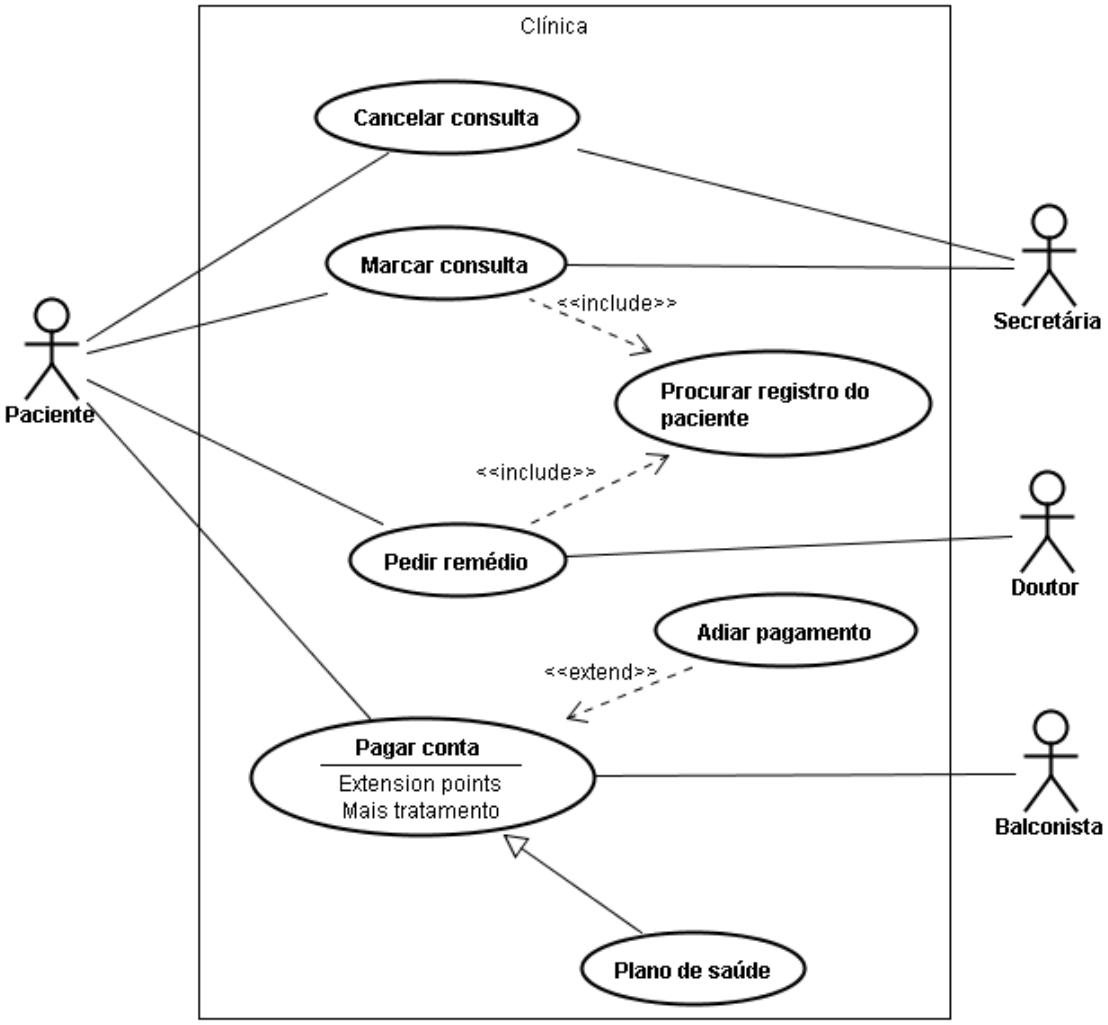

Figura 1 - Diagrama de Caso de Uso para uma Clínica
Fonte: (UCFG , 2012)

2.1.3 QR Code

Para falar sobre QR Codes primeiramente precisa-se falar sobre os códigos de barras, que segundo Rocha (2011) tornaram-se famosos pela sua velocidade de leitura e acurácia, automatizando vários processos comerciais. Os códigos de barras consistem basicamente da representação de números, impressos em forma de barras. O preto retém a luz e o branco reflete, possibilitando um leitor de interpretá-lo. Um leitor precisa estar configurado para ler determinado tipo de código de barras, pois existem vários padrões internacionais, que não serão citados pois não são eles o foco dessa seção.

Entretanto mesmo com todas as vantagens do código de barras, o mercado exigia mais informação retida de dentro de um código, o que levou pesquisadores a criarem códigos de barras 2D como pode-se ver alguns exemplos na Figuras 2 e Figura 3 . Isso gerou problemas como aumentar o tamanho da área do código, operações complicadas de leitura e encarecimento do custo de impressão.

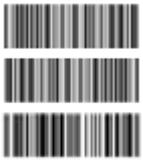

Figura 2 - Código de barras múltiplo
Fonte: (DENSO, 2010)

Figura 3 - Código de barras 2D com pilha
Fonte: (DENSO, 2010)

Para resolver esses problemas Denso em 1994 criou um padrão que que foram nomeados de *QR Code* que se baseia nos códigos de barras 2D que armazena informações na vertical e na horizontal, mostrado na Figura 4.

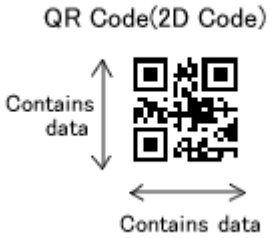

Figura 4 - Demonstração QR Code e código de barras simples.
Fonte: Adaptado (DENSO, 2010)

Essa tecnologia é tão poderosa que um único QR Code pode armazenar até 7089 caracteres numéricos ou 4296 caracteres alfanuméricos segundo a Figura 5 disponibilizada pela Denso.

		QR Code	PDF417	DataMatrix	Maxi Code
					
Developer(country)		DENSO(Japan)	Symbol Technologies (USA)	RVSI Acuity CiMatrix (USA)	UPS (USA)
Type		Matrix	Stacked Bar Code	Matrix	Matrix
Data capacity	Numeric	7,089	2,710	3,116	138
	Alphanumeric	4,296	1,850	2,355	93
	Binary	2,953	1,018	1,556	
	Kanji	1,817	554	778	

Figura 5 - Comparação entre os códigos de barras 2D.
Fonte: (DENSO, 2010)

Além de suas vantagens o QR Code ainda é resistente a possíveis danos ao código. O QR Code possui a capacidade de correção de erros, os dados contidos no código podem ser recuperados mesmo com o código se o código for parcialmente danificado (Denso, 2010), exemplos de códigos danificados que ainda podem ser utilizados, são ilustrados na Figura 6.

Figura 6 - Exemplo de possíveis danos causados ao QR Code.
Fonte: (DENSO, 2010)

2.1.4 XML

Segundo Marchal (2000) a *eXtensible Markup Language,* também conhecida como XML, é uma linguagem de marcação desenvolvida pelo W3C com o intuito principal de resolver algumas limitações da HTML.

Comenta Heitlinger (2001) que o XML teve o sucesso garantido pelo fato de grandes empresas como a Microsoft, Oracle, SAP, Software AG, IBM prestarem suporte. Outro fator que contribuiu para o rápido avanço do XML é a sua "natureza", que em pouco tempo tornou-se um padrão para troca de dados entre aplicações, como podemos ver na Figura 7.

Figura 7 - Troca de dados através de XML.
Fonte: (HEITLINGER, 2010)

A XML trouxe muitas melhorias e a principal é a opção de criar *tags* totalmente customizáveis, o que definiu a escolha da mesma para a utilização no *software*.

2.1.5 Java

Lançada em 1995 pela Sun, Java é uma linguagem orientada à objetos (DEITEL e DEITEL, 2005). Inicialmente, a linguagem Java foi largamente utilizada integradas às páginas da *web* para dar maior interatividade com a utilização de *applets*, porém hoje a linguagem permite desde aplicações *desktop* a aplicações *web* através de *servlets* e JSP.

É uma linguagem interpretada, logo possui grande portabilidade. Baseada em C++, seu código é compilado gerando um *bytecode*, que é interpretado em uma máquina virtual (DEITEL e DEITEL, 2005). Esta máquina virtual possui implementação em diversas plataformas e arquiteturas, permitindo que o *software* seja executado sem necessidade de adaptação do código para outra plataforma. Estas plataformas atendem desde celulares até mesmo aplicações para servidores, na qual o Java garante robustez com suporte a *multithreading* como Furgeri (2002) cita:

> "Threads (linhas de execução) são o meio pelo qual se consegue fazer com que mais de um evento aconteça simultaneamente em um programa. Assim, é possível criar servidores de rede multiusuários, em que cada thread, por exemplo, cuida de uma conexão de um usuário ao servidor, isto é, um mesmo programa pode ser executado várias vezes ao mesmo tempo e cada execução estar processando uma instrução em um ponto diferente do mesmo programa." (FURGERI, 2002, p. 49)

A escolha de Java se deve pela popularidade da linguagem como pode ser visto na Figura 8. Outro fator relevante foi a facilidade de encontrar material para auxílio no desenvolvimento, tanto como material de estudo.

Position Apr 2012	Position Apr 2011	Delta in Position	Programming Language	Ratings Apr 2012	Delta Apr 2011	Status
1	2	↑	C	17.555%	+1.39%	A
2	1	↓	Java	17.026%	-2.02%	A
3	3	=	C++	8.896%	-0.33%	A
4	8	↑↑↑↑	Objective-C	8.236%	+3.85%	A
5	4	↓	C#	7.348%	+0.16%	A
6	5	↓	PHP	5.288%	-1.30%	A
7	7	=	(Visual) Basic	4.962%	+0.28%	A
8	6	↓↓	Python	3.665%	-1.27%	A
9	10	↑	JavaScript	2.879%	+1.37%	A
10	9	↓	Perl	2.387%	+0.40%	A
11	11	=	Ruby	1.510%	+0.03%	A
12	24	↑↑↑↑↑↑↑↑↑↑	PL/SQL	1.373%	+0.92%	A
13	13	=	Delphi/Object Pascal	1.370%	+0.34%	A
14	35	↑↑↑↑↑↑↑↑↑↑	Visual Basic .NET	0.978%	+0.64%	A
15	15	=	Lisp	0.951%	+0.02%	A
16	17	↑	Pascal	0.812%	+0.10%	A
17	16	↓	Ada	0.783%	+0.01%	A--
18	18	=	Transact-SQL	0.760%	+0.18%	A
19	22	↑↑↑	Logo	0.652%	+0.12%	B
20	52	↑↑↑↑↑↑↑↑↑↑	NXT-G	0.578%	+0.35%	B

Figura 8 - As linguagens de programação mais populares.
Fonte: (Tiobe, 2012)

2.2 FERRAMENTAS

2.2.1 ZXing

Para interpretar os QR Codes optou-se por utilizar um projeto *open source* que se encarrega da leitura e retorno das informações contidas nos códigos sem qualquer operação adicional.

Segundo o próprio grupo, ZXing é projeto *open source* que suporta múltiplos formatos de código de barras, implementado em Java e migrado para

outra linguagens, com foco em utilizar câmeras de celulares para a decodificação de QR Code (ZXing, 2011), a Figura 9 apresenta a logomarca do grupo.

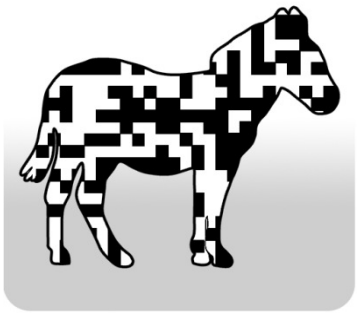

Figura 9 - Logomarca do grupo ZXing.
Fonte: (ZXING, 2011)

2.2.2 Eclipse

Gonçalves (2006, p. 21) comenta que a ferramenta Eclipse antes de se tornar um projeto *de código aberto* pertencia a IBM, que gastou cerca de 40 milhões de dólares no desenvolvimento do projeto. Após essa fase inicial o projeto foi transformado em código aberto para o consórcio chamado *Eclipse.org*, que incluiu grandes empresas como *Borland*, *Rational Software* e *Red Hat*.

> "O Eclipse em si fornece apenas o ambiente integrado para a execução dos *plug-ins* básicos, como editor de textos ASCII, sistema de ajuda e integração ao CVS. Para iniciar o desenvolvimento, em qualquer linguagem que seja, devem ser instalados *plug-ins* adicionais" (GONÇALVES, 2006, p. 22).

A ferramenta Eclipse foi escolhida por possuir código aberto e ser gratuita, que reduz os custos de desenvolvimento do projeto. Outro fator que influenciou a escolha é configuração inicial da ferramenta que contém poucos *plug-ins*, deixando-a leve, pois possui somente *plug-ins* básicos e muito maleáveis.

2.2.3 Android SDK

Segundo Hashimi (2010) antes de o sistema Android surgir e se tornar uma referência havia vários SOs para o ambiente dos *smartphones*, como por exemplo o Symbian SO, *Mobile* Linux, Iphone SO, Moblin (Intel) e entre outros sistemas proprietários. Entretanto, nenhum se tornou um padrão, pois suas API eram muito restritas, todas ficavam para trás se comparadas com o padrão *desktop* e foi nesse momento que a Google entrou em ação com premissas de fornecer acessibilidade e código aberto. A Google iniciou o projeto adquirindo a empresa Android Inc. em 2005 e lançou a primeira versão em 2008 como ilustra a Figura 10. Em 2008 o Android também se tornou código aberto.

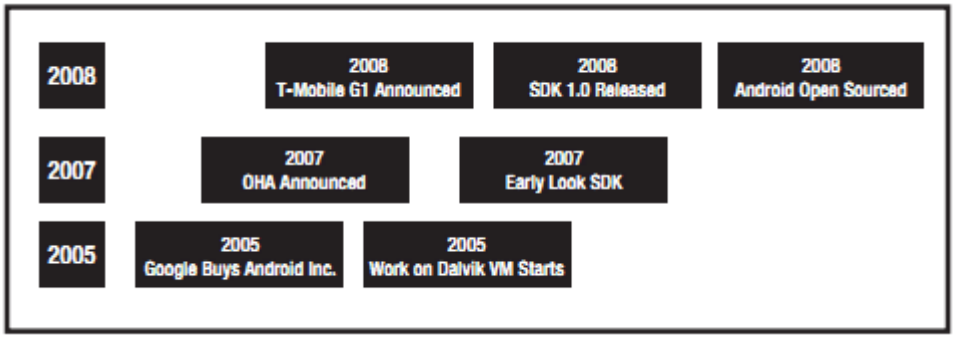

Figura 10 - Linha do tempo do Android até o primeiro lançamento
Fonte: (Hashimi, 2010)

Atualmente existem três linhas do Android uma voltada para *tablets* que é chamada 3.0, outra voltada para *smartphones* chamada 2.0 e a versão 4.0 que é direcionada tanto para *tablets* quanto para *smartphones* mais atuais (GOOGLE, 2012).

Segundo Hashimi (2010) os problemas que levaram a Google a revisão da máquina virtual Java e consequentemente a criação da Dalvik foram as limitações de memória de acesso direto (RAM) e a capacidade de armazenamento disponíveis em dispositivos móveis.

Para o melhor entendimento do funcionamento do Android discuti-se sobre a sua pilha de software que está demonstrada na Figura 11.

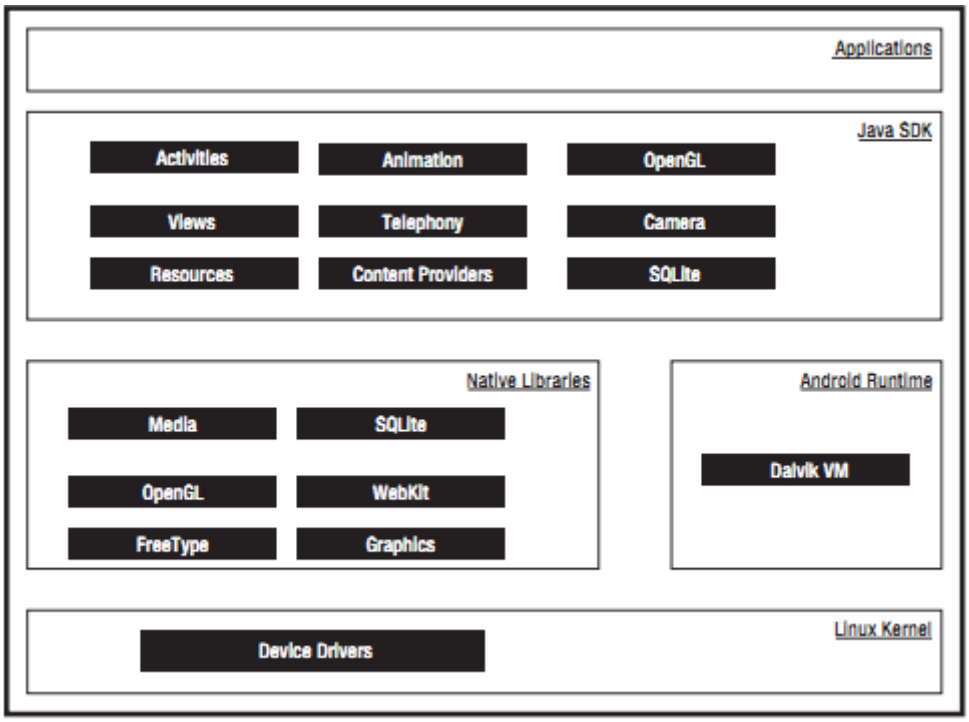

Figura 11 - Pilha de software do Android
Fonte: (Hashimi, 2010)

No núcleo do Android está o Linux na sua versão 2.6.29 que é responsável pelos *drivers*, acesso aos recursos. Como pode-se ver na base da figura.

No nível acima do Linux temos algumas bibliotecas em C/C++ como exemplo a *OpenGL* e a máquina virtual Dalvik. Acima tem-se o ambiente o qual as aplicações utilizam linguagem Java para se comunicar com o Android e seus recursos.

A escolha do Android foi devida o mesmo possuir uma fatia considerável do mercado como aponta a pesquisa realizada pela organização Gartner levantando a porcentagem de mercado em número de dispositivos vendidos para usuários finais no último trimestre de 2011, apresentado na Gráfico 1.

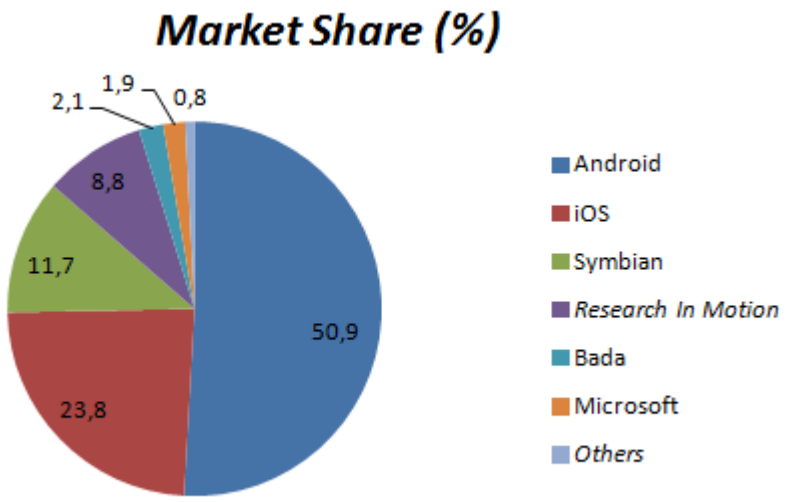

Gráfico 1 - Vendas de smartphones por sistema operacional
Fonte: Adaptado (Gartner, 2012)

2.2.4 PostgreSQL

PostgreSQL é um sistema gerenciador de banco de dados (SGBD) objeto-relacional, criado a partir de um projeto chamado Postgres, desenvolvido na universidade de Berkley em 1986, este projeto consistia em criar um novo sistema de armazenamento de dados com apoio do *Army Research Office* (ARO) e da *National Science Foundation* (NSF)) (MILANI,2008). O Postgres teve sua primeira versão lançada em 1988, e finalizado na sua versão 4.2 em 1993, devido aos altos custos de manutenção relacionado à grande utilização.

Em 1994, Andrew Yu e Jolly Chen criaram o Postgre95, derivado do projeto Postgres, com um interpretador de *SQL* e de código aberto. Com a grande utilização e vários colaboradores, o Postgre95 se tornou PostgreSQL para refletir as mudanças entre o Postgres original e a capacidade de interpretação da linguagem SQL.

O PostgreSQL utiliza a licença BSD, possui características como chave estrangeira, gatilhos, visões e capacidades comparadas à SGBDs pagos como Microsoft SQL Server e Oracle. Uma dessas características é a utilização de transações ACID, além vários outros conceitos de banco de dados como chave estrangeira , uso de Índices, triggers(gatilhos) e uso de linguagem

procedural(PL), Desse modo, ele permite criar funções dentro do banco de dados, permitindo deixar regras do sistema fora da aplicação, garantindo maior leveza ao *software*.

"A utilização de índices pelo PostgreSQL permite realizar consultas extremamente rápidas, graças aos vários tipos, cada um com seu algoritmo, como B-Tree(Árvore B), R-Tree(Árvore R) , hash e GiST" (*The PostgreSQL Global Development Group* , 2005, p.224).

O PostgreSQL foi escolhido por ser um SGBDOR com licença BSD (THE POSTGRESQL GLOBAL DEVELOPMENT GROUP, 2005) garantindo assim o *software* livre do pagamento de receitas e possibilitando a venda ou utilização como serviço sem a necessidade de abertura do código.

2.2.5 Apache Tomcat

Tomcat é um *container* para aplicações Java na web, o *container* é um objeto que controla e possui outros objetos, no caso *servlets*. Toda requisição *web* para um *servlet* é tratada pelo *container*, ilustrado pela Figura 12.

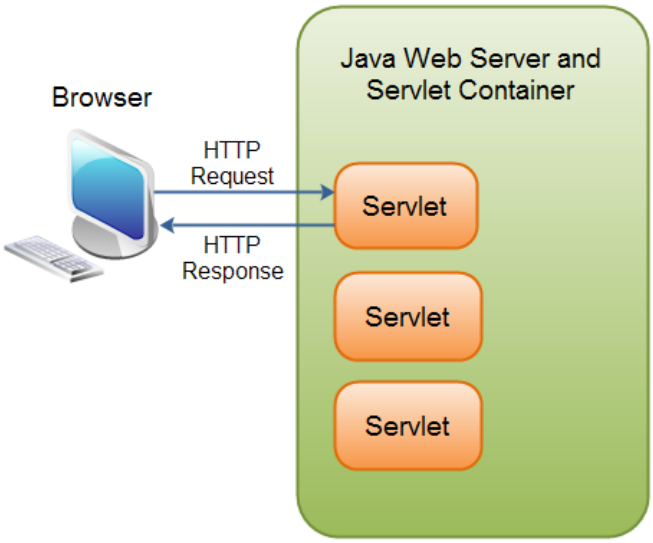

Figura 12 - Utilização de Containers
Fonte: (Jenkov, 2012)

O Tomcat devido à sua robustez, também age como servidor HTTP, porém pode ser integrado à outros Servidores HTTP (LUCOW e MELO, 2010). É mantido pela Apache Foudation.

3 METODOLOGIA

Neste capítulo será apresentado a metodologia utilizada para o desenvolvimentod o projeto.

3.1 CONTEXTO DO PROJETO

Para demonstrar a metodologia empregada no desenvolvimento deste projeto foi criado um diagrama, conforme ilustra a Figura 12 e 13.

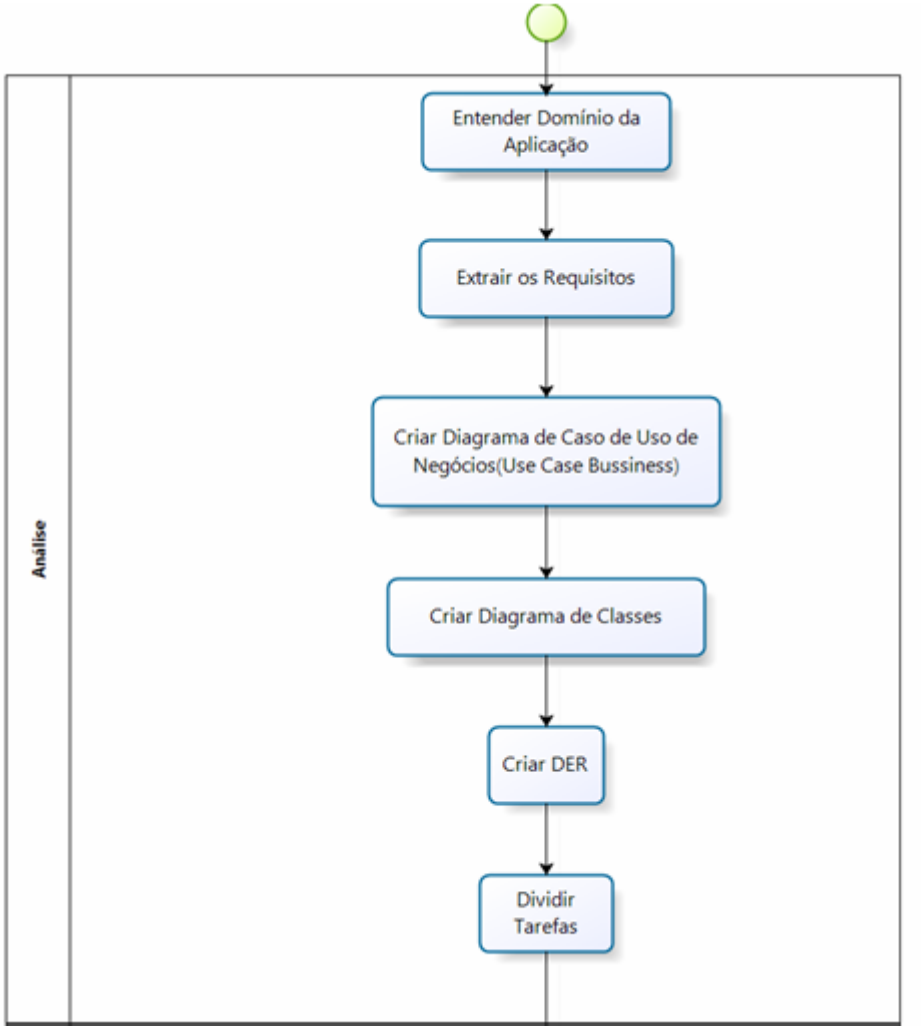

Figura 12 - Diagrama da Fase de Análise.

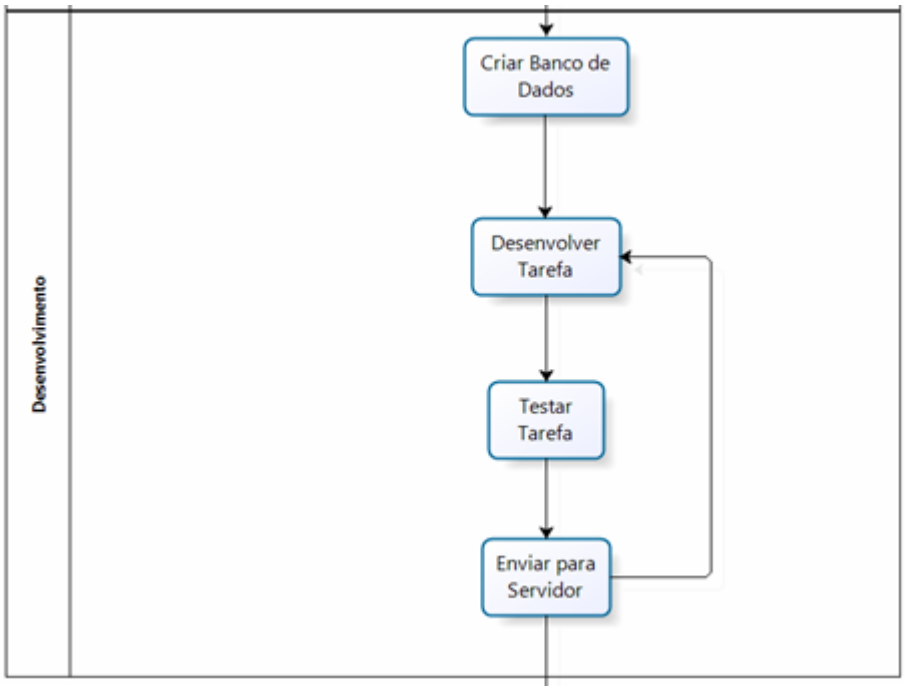

Figura 13 - Diagrama da Metodologia Utilizada

3.2 EXECUÇÃO DO PROJETO

Para o desenvolvimento do trabalho foi utilizado o fluxograma ilustrado na Figura 13, o qual apresentou a metodologia aplicada neste projeto. Essa metodologia trata-se de cada fase necessária à execução do projeto.

Foram também utilizadas práticas consideradas úteis da metodologia XP, em conjunto com elementos da metodologia RUP. Um desses fatores foi o desenvolvimento iterativo do XP, com a divisão em tarefas e uma redução da documentação (TELES, 2004).

A metodologia foi dividida em três etapas: Análise, Desenvolvimento e Manutenção. Durante o período inicial do projeto, foi realizada toda a fase de análise, a fim de descobrir os requisitos mínimos necessários para a liberação de uma versão funcional do sistema. É importante ressaltar que a utilização da simplicidade especificada pela metodologia XP, deve ser empregada, para evitar funcionalidades que não serão utilizadas e não causar perda de tempo durante o desenvolvimento do projeto.

3.3 ANÁLISE

Nesta etapa foram utilizadas alguns padrões da metodologia RUP, como a documentação através de Análise de Requisitos e Digramas como os de Casos de Uso de Negócio (*Business Use Case*) e Diagrama de classes. O objetivo dessa fase é documentar o projeto e facilitar o desenvolvimento.

A metodologia XP entra nessa fase com uma melhora com relação ao tradicional processo RUP, visando documentar somente o que é necessário. Ao começo do projeto, será analisado o conjunto básico de funcionalidades do sistema antes do desenvolvimento.

3.3.1 Entender domínio da aplicação

Toda a fundamentação do projeto foi feita nesta etapa em que a equipe é reunida e devem-se discutir quais os limites que a aplicação deverá ter, e quais medidas serão tomadas durante a fase de análise da aplicação.

A grande vantagem de procurar entender o domínio da aplicação apresentará resultados durante a extração de requisitos, pois a aplicação terá um escopo de onde irá atuar, ou seja entendendo o domínio da aplicação a extração de requisítos flui de uma melhor forma.

3.3.2 Extrair requisitos

Nesta fase da metodologia utilizada para o projeto, foram extraídos primeiramente os requisitos do Conjunto Básico da Aplicação. O conjunto básico deve ser tratado como o núcleo do sistema de modo que todas as requisições feitas serão analisadas e desenvolvidas a partir deste.

Um exemplo dessa aplicação é a parte de imóveis, cujo conjunto básico da aplicação gira em torno do cadastro de imóveis e a sua disponibilização para visualização, outras funcionalidades estarão dependentes deste conjunto, como por exemplo, um sistema de reserva de imóvel, os requisitos para esta funcionalidade serão extraídos após o desenvolvimento do núcleo do sistema.

O processo básico da análise de requisitos é a criação de um documento de requisitos, que demonstre o que deverá ser feito para a funcionalidade e para que ela servirá e qual o resultado esperado.

Após o núcleo do sistema estar desenvolvido e no mercado, novas requisições de clientes e melhoramentos provavelmente surgirão. A cada requisição será necessário fazer uma nova análise e estar sempre em contato com o cliente, a fim de evitar os tradicionais erros de comunicação.

Uma solução é a utilização de duas práticas da metodologia XP, estórias e cartões, em que o cliente escreve o que deseja e qual o resultado esperado, e também a prática do cliente presente, em que o cliente acompanha o início e o final de cada etapa (TELES, 2004).

3.3.3 Diagramas

Diagramas são essenciais para a divisão de tarefas e para o desenvolvimento e numa futura manutenção do código. A utilização de diagramas da UML garante a padronização e legibilidade aos eventuais processos de manutenção. Foram escolhidos dois diagramas, para fazerem parte da aplicação, *Business Use Case*(Figura 14), Diagrama de Classes(Figura 15) e o Diagrama Entidade e Relacionamento(Figura 16).

O *Business Use Case* é utilizado para descrever o comportamento que o sistema terá diante de clientes, parceiros e utilizadores. A escolha deste diagrama deve-se a necessdade de delimitar melhor o escopo da aplicação.

Para ajudar na divisão de tarefas, o diagrama de classes foi utilizado, para definir quais classes serão criadas e qual o relacionamento entre estas. O diagrama de Classes é utilizado em conjunto no desenvolvimento do Diagrama de Entidade Relacionamento (DER).

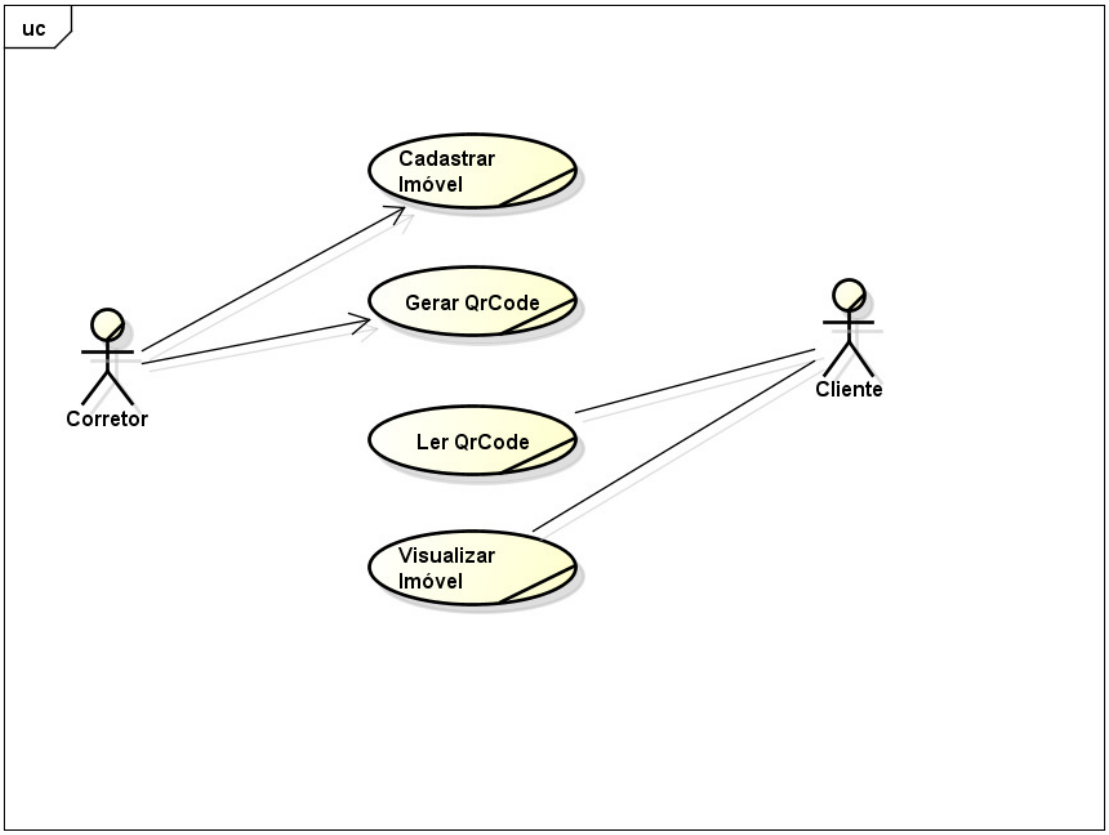

Figura 14- Business Use Case

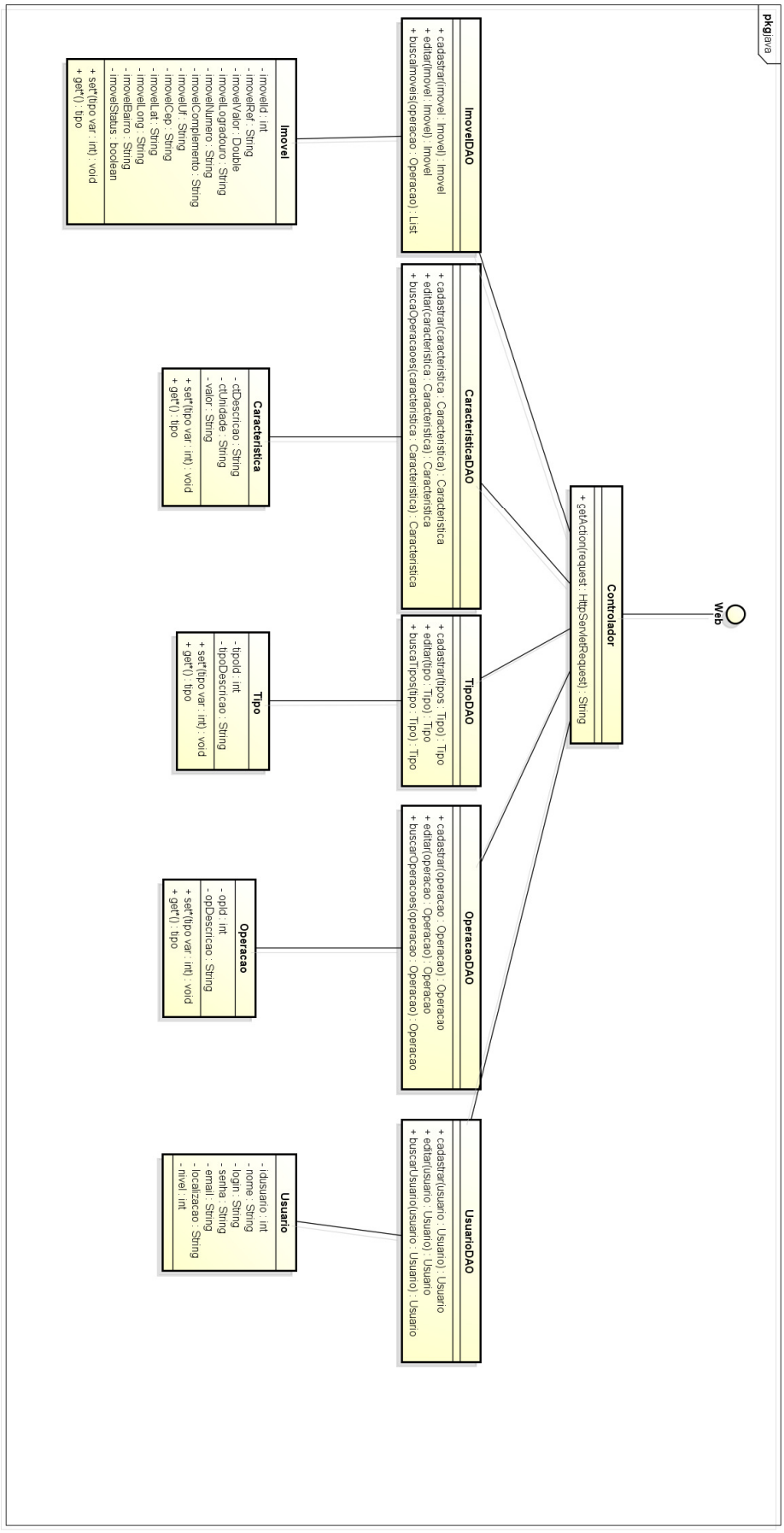

Figura 15- Diagrama de Classes

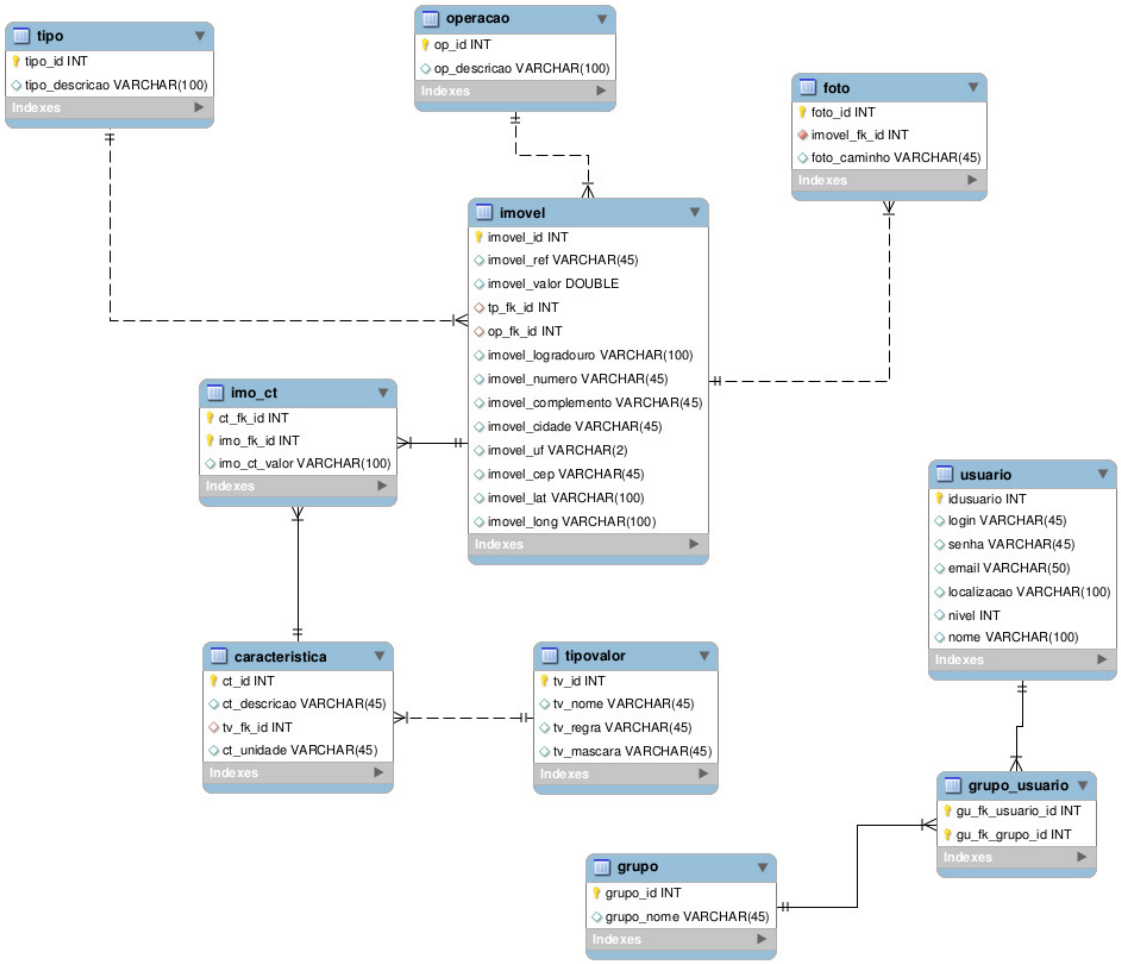

Figura 16 - Digrama Entidade-Relacionamento

3.3.4 Dividir tarefas

Nesta fase o levantamento dos requisitos, foi dividida em tarefas para os desenvolvedores, cada tarefa é voltada para atingir um objetivo especifico do projeto. Desse modo, estas tarefas devem ser delegadas, a fim de realizar todo o trabalho em “passos de bebê”, como prega a metodologia XP.

A utilização desta tática permite melhor desenvolvimento do código, pois garante o desenvolvedor se concentrar em pequenas tarefas, e não se deparar com várias mensagens de erro, que são um resultado natural do desenvolvimento de algo muito complexo de uma só vez.

Outra vantagem da divisão de tarefas é que a metodologia XP é baseada no conceito de desenvolvimento iterativo (TELES, 2004), de modo que cada tarefa é desenvolvida, validada e integrada ao código final.

3.4 DESENVOLVIMENTO

Nesta fase foi aplicado o desenvolvimento iterativo, com a utilização das tarefas definidas na fase anterior, e cada iteração utiliza os três passos descritos a seguir. Nesta fase ocorre a criação do banco de dados com base no DER desenvolvido. Deve-se observar que três artefatos serão gerados após a fase do desenvolvimento: a Aplicação, Código de Testes e Manuais.

Esta fase e os seus passos são executados em dois momentos distintos, sendo que o primeiro momento é visa liberar o conjunto básico da aplicação, ou seja, um sistema funcional, onde o usuário poderá fazer as tarefas especificadas no documento de requisitos inicial do projeto.

O segundo momento é fase em que requisitos adicionais e que não foram especificados para o núcleo do sistemas são pedidos e incrementam a aplicação, como relatórios e outros tipos de controle que sejam condizentes com o conjunto básico do sistema.

3.4.1 Desenvolver tarefa

Esta etapa compreende o desenvolvimento de código da tarefa. O código deve suprir o objetivo final da tarefa. É importante os desenvolvedores manterem um padrão de código definido antes do início. Isto garante a facilidade de leitura.

Durante o desenvolvimento, ressalta-se a utilização de comentários no código e até mesmo uma codificação com base no princípio da simplicidade, em que o código deve ser enxuto ao máximo, utilizando o mínimo de linhas possível e o mais legível possível que se torna fácil atingir este objetivo graças ao desenvolvimento iterativo.

Este princípio afirma que toda a complexidade deve ser descartada e manter a codificação somente dentro do escopo da tarefa, facilitando o entendimento futuro do código e até mesmo a sua otimização.

O desenvolvimento da tarefa possui um tempo estimado pelo próprio desenvolvedor, fazendo uma análise de quanto tempo ele levará para codifica determinada funcionalidade. Assim que uma tarefa é terminada deverá ser focado no próximo passo, garantindo assim as iterações propostas pela metodologia. Outro fator do desenvolvimento desse modo, é que a partir do momento que a tarefa está em fase de testes, o desenvolvedor fica livre para iniciar outra tarefa, não ficando ocioso durante os testes.

3.4.2 Testar tarefa

A etapa de testes visa realizar vários testes desde a interface (testes de caixa-preta) até o impacto em outras funcionalidades do sistema (testes de caixa-branca). Desse modo, é importante testar durante o desenvolvimento e também manter uma equipe somente de testes.

Os testes de caixa-preta verificam o comportamento externo do sistema, a interface, ou seja, como o sistema irá se comportar na visão do usuário. Os testes de caixa-branca verificam o funcionamento interno do sistema e qual o seu impacto.

Uma prática para testes de caixa-preta que pode ser utilizada por desenvolvedores é o TDD (*Test Driven-Development*) que visa escrever os testes antes do desenvolvimento da tarefa. O *Test Driven-Development* é uma técnica de testes, em que se visa escrever o teste antes mesmo do código (BECK, 2002). Segundo o Autor, este tipo de teste faz com que o desenvolvedor se encoraje a escrever o código para que somente passe pelo teste, deste modo, o código se mantém simples e compacto.

3.4.3 Enviar para o servidor

Nesta etapa ocorre a integração do código da tarefa, com toda a aplicação, Sendo que o usuário poderá utilizar a funcionalidade desenvolvida. Mantendo dois ambientes, um de produção e outro de desenvolvimento, garantese que não haja interferência nas tarefas básicas do usuário, inclusive

no desempenho da aplicação, tendo em vista que muitas vezes o código utilizado antes de testado, pode ser lento.

O servidor de desenvolvimento é um servidor de rede interna, logo a velocidade de comunicação é rápida, enquanto o servidor de produção é mantido em um local com infra-estrutura adequada com adequada velocidade de conexão, e também um servidor mais estável e com processamento adequado para dar suporte à aplicação.

4 DESENVOLVIMENTO

Neste capítulo sera apresentado o desenvolvimento das aplicações, com imagens e códigos referentes.

4.1 TAREFAS

Para o desenvolvimento, cada requisito foi dividido em tarefas. Para esta divisão foi utilizado o *software* DotProject. Com este *software* é possível gerenciar e acompanhar cada tarefa, conforme ilustra a Figura 17.

Figura 17 - Tarefas do Módulo WEB

É possível associar um desenvolvedor ou mais para cada tarefa e também permitir que os desenvolvedores alterem o progresso da tarefa.

4.2 DESENVOLVIMENTO WEB

4.2.1 Visão Geral do Sistema

O Sistema é composto de um módulo para a administração dos imóveis cadastrados conforme ilustra a Figura 18 e outro módulo para a visualização por clientes, apresentado pela Figura 19. Com o sistema de administração é possível fazer o controle dos imóveis dinamicamente, assim apresentando para possíveis clientes , os imóveis disponibilizados pela imobiliária, Desse modo não há dependência da parte móvel, criando assim um outro canal de comunicação e divulgação dos imóveis.

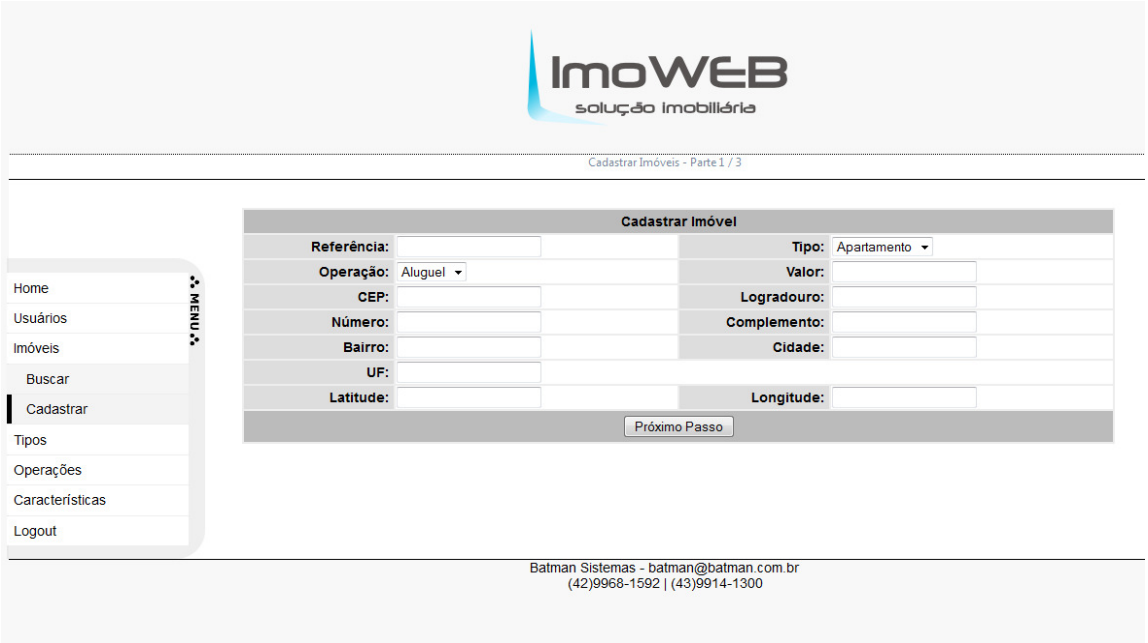

Figura 18 - Módulo de Administração de Imóveis

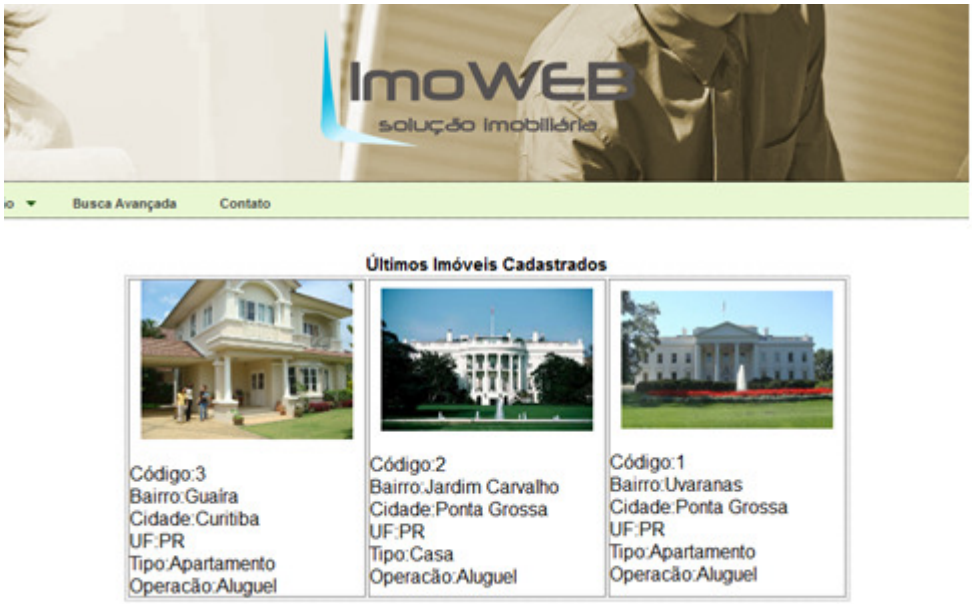

Figura 19 - Módulo de Visualização de Imóveis

O módulo administrativo permite ao utilizador o gerenciamento de imóveis. A utilização do módulo administrativo pode ser dividida em dois tipos de usuários: administrador e gerente. Um administrador pode controlar usuários também. No módulo de visualização dos imóveis, o cliente pode filtrar imóveis e visualizar mais detalhes sobre estes.

O sistema utiliza-se de um controlador, não permitindo que o utilizador saiba qual o endereço exato da página que está sendo visualizada e também um controle de requisições, ou seja, o que é permitido para o usuário. As URIS formadas pelo controlador seguem o padrão: */index?pagina=principal&atributos*
Sendo:

- Index: o controlador;
- Pagina: página que o usuário estará acessando;
- Atributos: diversos atributos que podem ser utilizados.

Esse encapsulamento da URI real da página ou Servlet utilizado, garante maior segurança à aplicação, pois evita que o caminho real para um arquivo da aplicação seja mostrado, sendo assim uma maneira de evitar a descoberta de vulnerabilidades de código.

4.2.2 Geração de QR Code

Para utilização com o sistema móvel, ao cadastrar um imóvel é possível a geração do QR Code específico para este. O QR Code gerado irá conter uma URL, direcionada para o imóvel.

A URL formada possui o seguinte padrão:

http://www.urldosite.com.br/ImoWeb/SvImovel?pagina=imovel&id=3

O QR Code pode ser gerado várias vezes, e também ter um tamanho de até 6000 pixels, definido de acordo com a necessidade do gerente ou administrador, para que possa ser fixado em imóveis sem perdas de qualidade. Nas figuras 20 e 21 é possível visualizar o processo de geração do QR Code através do módulo de Administração.

Figura 20 - Tela de Geração do QR Code

Figura 21 - QR Code Gerado

A geração do QR Code é feita por meio da biblioteca ZXing , em um *servlet* chamado GerarQR. Este servlet recebe como parâmetro o id do imóvel(“ImoFkId”) e o tamanho a ser gerado(“tam”), como é mostrado no Quadro 1.

```
protected void processRequest(HttpServletRequest request,
HttpServletResponse response)
          throws ServletException, IOException {
       response.setContentType("image/png;charset=UTF-8");
        OutputStream os = response.getOutputStream();

       try {
       String imoId = request.getParameter("imoFkId");
       int tam;
       if(request.getParameter("tam").isEmpty())
       tam = 1024;
       else
       tam = Integer.parseInt(request.getParameter("tam"));
       Charset charset = Charset.forName("ISO-8859-1");
       CharsetEncoder encoder = charset.newEncoder();
       byte[] b = null;
       try {
          ByteBuffer bbuf =
encoder.encode(CharBuffer.wrap("http://www.dvdes.com.br/Im
obWeb/index?pagina=imovel&id="+imoId));
          b = bbuf.array();
       } catch (CharacterCodingException e) {
          System.out.println(e.getMessage());
       }
       String data = new String(b, "ISO-8859-1");
       try {
          data = new String(b, "ISO-8859-1");
       } catch (UnsupportedEncodingException e) {
          System.out.println(e.getMessage());
       }
       int h = tam;
       int w = tam;
       com.google.zxing.Writer writer = new QRCodeWriter();
       try {
          matrix = writer.encode(data,
com.google.zxing.BarcodeFormat.QR_CODE, w, h);
       } catch (com.google.zxing.WriterException e) {
          System.out.println(e.getMessage());
       }
        MatrixToImageWriter.writeToStream(matrix, "PNG", os);
        os.flush();
       } finally {
        os.close();
       }
```

Quadro 1 - Servlet responsável por gerar QR Code

O *servlet* altera o *content-type* da página para “image/png”, possibilitado a utilização do *OutputStream* para escrever na página o QR Code gerado. A geração deste é realizado pela Classe com.google.zxing.Writer. Esta classe possui o método *encode*, que transforma uma cadeia de *bytes* em código binário e escreve no dispositivo de *output* pelo método *writeToStream*.

4.3 ANDROID

Basicamente as funcionalidades que a aplicação Android traz são: Tela de apresentação e menu inicial, interpretação do QR Code, manipulação do conteúdo do QR Code, requisição HTTP, recebimento do XML, manipulação do conteúdo do XML, apresentação do imóvel, gerenciamento dos favoritos e apresentação dos favoritos.

4.3.1 Funcionamento básico do sistema

Ao iniciar a aplicação móvel uma tela de apresentação será mostrada que contem o logo tipo e um controle de versão apresentada na Figura 22. Essa tela aguarda por três segundos ou um toque do usuário na tela para assim acionar a próxima.

Figura 22 - Tela de apresentação

Após o evento de toque na tela ou três segundos é apresentado ao usuário uma tela de menu. Com três opções que são: Iniciar o *scanner* de QR *Code,* abrir a tela de favoritos e exibir a tela de endereços não visitados como mostra a Figura 23. Ainda na tela de menu inicial o usuário tem a possibilidade de acessar uma aba contendo informações gerais sobre o sistema como pode ser visto ver na Figura 24.

Figura 23 - Tela de menu inicial.

Figura 24 - Tela de menu inicial com aba Sobre selecionada

Ao selecionar a opção do *scanner* o usuário será redirecionado para tela da biblioteca ZXing, na qual o ele deve centralizar o QR *Code* na tela e então efetuar a leitura. Após a leitura caso o usuário não estiver conectado na *Internet* uma mensagem de aviso será apresentada, perguntando se ele deseja salvar o endereço para visitar quando houver acesso à *Internet*, como ilustrado na Figura 25.

Figura 25- Tela de menu com aviso de aparelho não conectado

Selecionando a opção voltar o sistema não realiza operações e somente deixa o usuário na tela de menu inicial, para que o mesmo possa selecionar alguma outra opção desejada. A opção adicionar salva o endereço no banco de dados do Android e exibe uma mensagem de *feedback* informando que a operação foi realizada com sucesso, Figura 26.

Com o usuário conectado à *Internet* o encerramento da leitura ocorre de uma forma diferente da apresentada anteriormente, que recebe uma mensagem de aviso que o sistema está carregando o imóvel. Ao término do carregamento o usuário é direcionado para a tela de apresentação do imóvel.

Figura 26 - Mensagem de *feedback*

Visando uma melhor exibição das informações foram criadas três abas as quais são: Informações, Endereço e Fotos.

A aba Informações contêm informações gerais sobre o imóvel e uma opção para adicionar o imóvel aos favoritos, apresentado na Figura 24. A localização no mapa e o endereço do imóvel são apresentados na segunda aba da tela, exibido na Figura 27. Para facilitar a navegação do usuário entre as fotos do imóvel foi criado uma lista com a miniatura das imagens do imóvel podendo o usuário selecionar facilmente uma foto e em baixo manipular a mesma, a Figura 28 mostra a primeira imagem da lista selecionada e abaixo a foto em tamanho maior com *zoom*.

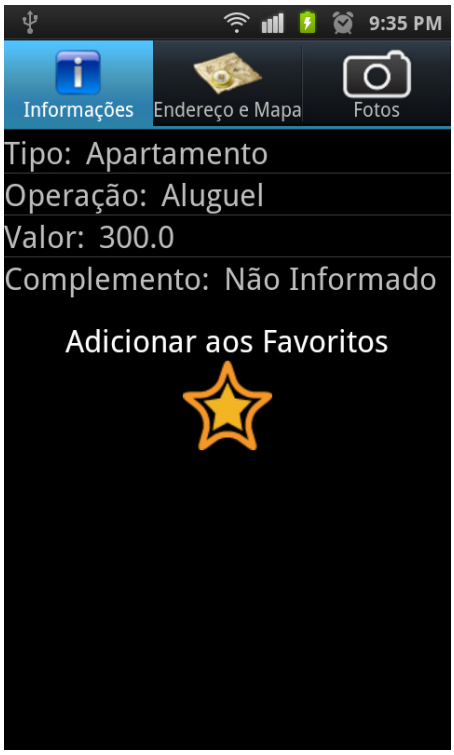

Figura 27 - Tela do imóvel com aba Informações selecionada

Figura 28 - Tela do imóvel com aba Endereço selecionada

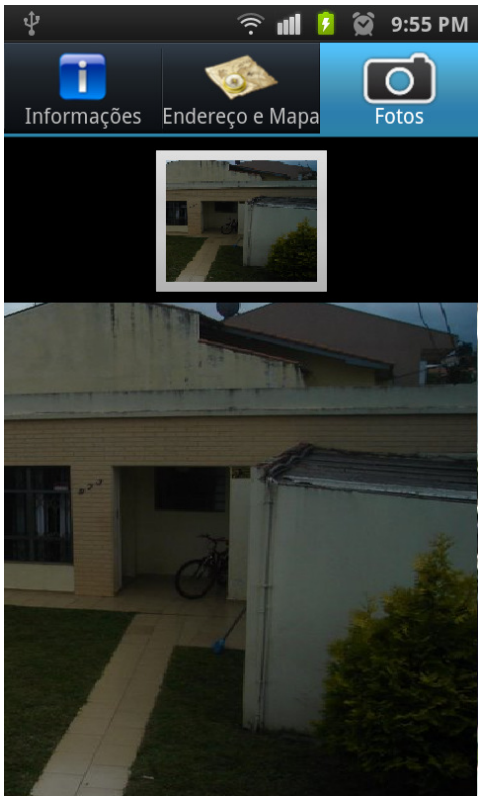

Figura 29 - Tela do imóvel com aba fotos selecionada

A opção “Adicionar aos Favoritos” presente na Figura 30, insere o imóvel no banco de dados do Android para que o usuário tenha a possibilidade de acessar as informações do mesmo sem a necessidade de efetuar a leitura de um QR *Code*. O imóvel pode ser acessado pela opção “Favoritos” ilustrado pela Figura 23, que ao selecionar essa opção será automaticamente direcionado para a tela de favoritos onde é apresentado uma lista com os imóveis armazenados como favoritos, exibido na Figura 31. Ao realizar um clique longo sobre um dos imóveis listados uma mensagem será exibida ao usuário, que o possibilita de visualizar o imóvel ou excluir o mesmo da listagem, como nos mostra a Figura 28.

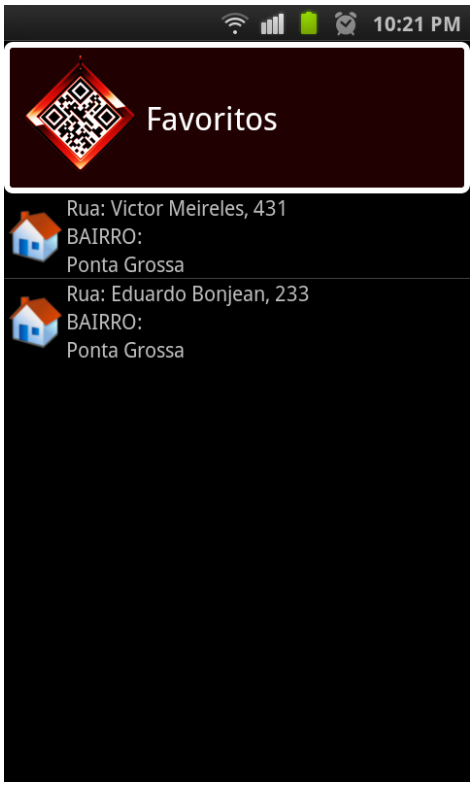

Figura 30 - Tela de favoritos

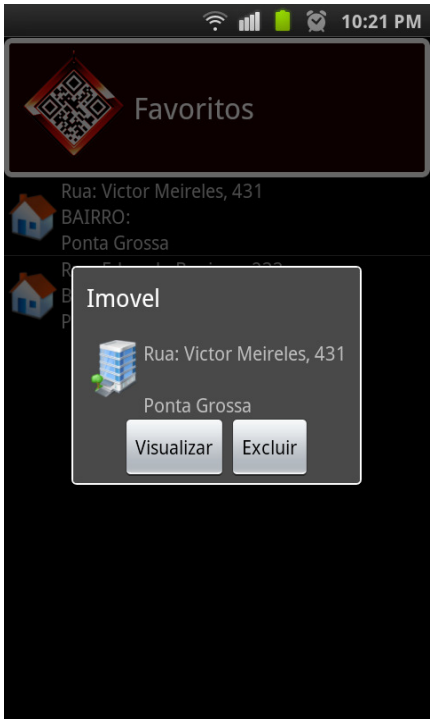

Figura 31 - Tela de favoritos com as opções para o imóvel selecionado

Na tela de menu inicial o usuário tem a possibilidade de acessar uma listagem de endereços não visitados como mostra a Figura 23, que ao acessar essa opção será direcionado para tela de endereços não visitados, apresentado na Figura 32. Seguindo o mesmo padrão da tela de favoritos, ao realizar um clique longo sobre um item da lista será apresentada uma mensagem possibilitando o usuário de visualizar o endereço ou o excluí-lo da listagem, Figura 33.

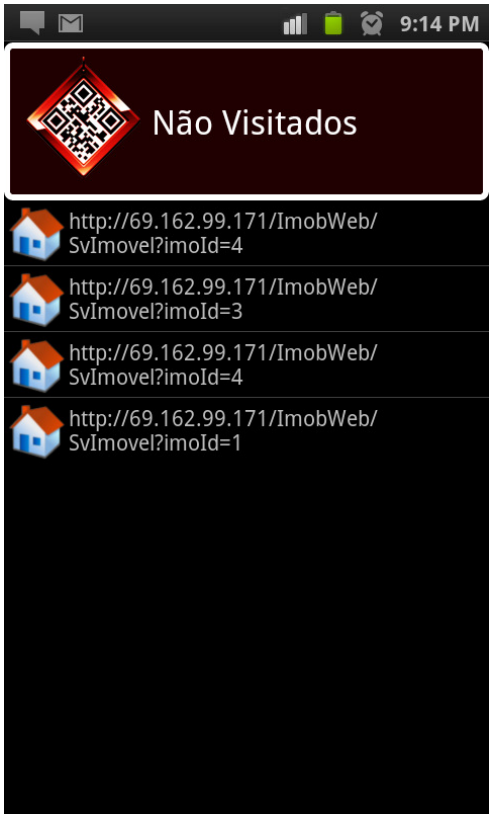

Figura 32 - Tela de endereços não visitados

Figura 33 - Item selecionado na tela de endereços não visitados

4.3.2 Exemplo de funcionamento do Android

No sistema Android as telas são construídas através de arquivos XML. Neste trabalho não serão apresentados todos os arquivos, mas como exemplo pode-se utilizar a Quadro 2, que nos mostra o arquivo referente a tela de apresentação do sistema, o arquivo possui *tags* para exibição de texto (*TextView*), *layouts* (*LinearLayout, RelativeLayout*) e de imagens (*ImageView)*.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
  android:layout_width="fill_parent"
  android:layout_height="fill_parent"
  android:id="@+id/TheSplashLayout"
  android:layout_gravity="center"
  android:background="@color/background"
  android:weightSum="1">

 <RelativeLayout
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:id="@+id/relativeLayout1">

    <TextView
     android:layout_alignParentBottom="true"
     android:textAppearance="?android:attr/textAppearanceSmall"
     android:layout_height="wrap_content"
     android:layout_width="wrap_content"
     android:layout_alignParentRight="true"
     android:text="@string/Versao"
     android:id="@+id/textView1"
     android:textColor="@color/versao"/>

   <ImageView
    android:layout_width="wrap_content"
    android:id="@+id/imageView1"
    android:src="@drawable/imoweb"
    android:layout_height="wrap_content"
    android:layout_centerVertical="true"
    android:layout_centerHorizontal="true"/>

 </RelativeLayout>
</LinearLayout>
```

Quadro 2 - Arquivo XML Responsável pela tela de apresentação

Tal arquivo é carregado por uma classe que estende a *Activity Class* do sistema Android, que possui métodos específicos para o carregamento e apresentação da tela, afim de demonstrar essa operação apresenta-se o Quadro 3 que carrega o arquivo XML da tela de apresentação como mostra a Figura 20.

```
import imoweb.Main.R;
import imoweb.Telas.TelaMenuInicial;
import android.app.Activity;
import android.content.Intent;
import android.content.pm.ActivityInfo;
import android.os.Bundle;
import android.view.MotionEvent;

public class SplashScreen extends Activity {
	private Thread mSplashThread;
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.tela_splash);
		setRequestedOrientation(
		 ActivityInfo.SCREEN_ORIENTATION_PORTRAIT);
		final SplashScreen sPlashScreen = this;
		mSplashThread = new Thread() {
			@Override
			public void run() {
			   try {
			      synchronized (this) {
			      wait(3000);
				}
			       } catch (InterruptedException ex) {
					ex.getMessage();
			  }
			finish();

			Intent intent = new Intent();
			intent.setClass(sPlashScreen,
TelaMenuInicial.class);
			startActivity(intent);
			stop();
			}
		};
		mSplashThread.start();
	}

	@Override
	public boolean onTouchEvent(MotionEvent evt) {
		 if (evt.getAction() == MotionEvent.ACTION_DOWN) {
			synchronized (mSplashThread) {
			mSplashThread.notifyAll();
		 }
		}
		return true;
}
```

Quadro 3 - Classe Java responsável por carregar a tela de apresentação

Após realizar a operação de carregamento da tela de apresentação, uma *thread* aguarda por três segundos para finalizar ou um toque na tela para então carregar a tela de menu inicial.

4.3.3 Interpretar QR Code

Afim de interpretar a imagem que possui um QR *Code* utiliza-se a biblioteca para Android chamada ZXing que foi incorporada no projeto abstraindo assim os processos necessários para extrair o conteúdo do mesmo.

A partir da tela de menu, um botão dispara uma requisição para a biblioteca Quadro 4.

```
public void onClick(View v) {
        try{
                Toast.makeText(getApplicationContext(),
                "Carregando o leitor de QR Codes!",
                Toast.LENGTH_SHORT).show();

                Intent intent = new
                Intent("com.google.zxing.client.android.SCAN");
                intent.putExtra("SCAN_MODE",
                "QR_CODE_MODE");

                startActivityForResult(intent, 0);


        }catch(Exception e){
                Toast.makeText(getApplicationContext(),
                "Falha ao iniciar o scanner.",
                Toast.LENGTH_SHORT).show();
        }
}
```

Quadro 4 - Evento responsável por acionar a biblioteca ZXing

Os parâmetros passados para a biblioteca determinam que será realizada uma leitura de QR Code, definir que será um QR Code é muito importante pois como foi dito antes a biblioteca suporta vários padrões de

códigos. Após o processamento o sistema recebe o resultado através de método pré-determinado, demonstrado no Quadro 5, que recebe o resultado e efetua a requisição do XML referente ao imóvel.

```
public void onActivityResult(int requestCode, int resultCode, Intent
intent){
     if (requestCode == 0) {

         if (resultCode == RESULT_OK) {

              String contents = intent.getStringExtra("SCAN_RESULT");
              String format =
                   intent.getStringExtra("SCAN_RESULT_FORMAT");

             if(Utils.isOnline(this)){

                  try{
                       parser = new XmlParser(contents);
                       Imovel im = parser.parse();

                       Intent myIntent = new Intent(TelaMenuInicial.this,
                                TelaImovel.class);

                       Bundle bd = new Bundle();
                       bd.putSerializable("imovel", im);

                       myIntent.putExtras(bd);
                       startActivity(myIntent);

                  }catch(Exception e){

                       new AlertDialog.Builder(this) .setTitle("Aviso!").
                       setMessage("Falha ao carregar o imóvel. Tente" +
                       "novamente!")
                       .setPositiveButton("Ok", new
                      DialogInterface.OnClickListener() {
                          public void onClick(DialogInterface dialog,
                          int whichButton) {  }
                       }).show();
                  }
         }else{
               new DialogNaoVisitados(this, contents).show();
         }
      }
   }
}
```

Quadro 5 - Método responsável por receber o resultado

4.3.4 Requisição do XML

O documento XML referente ao imóvel é feito através de uma requisição internet, seja ela via *wireless* ou 3G a requisição é feita da mesma forma, o Android deixa isso transparente exemplificado a seguir pela Quadro 6.

```
private Document requisitarXml() throws Exception{

        URL url = new URL(getEndereco());

        DocumentBuilderFactory dbf =
                DocumentBuilderFactory.newInstance();

        DocumentBuilder db = dbf.newDocumentBuilder();

        Document doc =
                db.parse(new InputSource(url.openStream()));

        doc.getDocumentElement().normalize();

        return doc;

}
```

Quadro 6 - Requisição e criação do documento XML do imóvel

4.3.5 Exibição do mapa

Para exibir o mapa foi necessário gerar uma chave no endereço https://developers.google.com/maps/, sem a qual não é possível realiza. A chave utilizada neste projeto pode ser visualizada na Figura 38. Sem a chave o mapa não é exibido na tela do celular, somente uma tela branca.

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

        <com.google.android.maps.MapView
                android:id="@+id/mapView"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:enabled="true"
                android:clickable="true"
                android:apiKey=
                "02FlQOv6Fx_pdyFYsLyhr-EESxOpOaQqbaOPiCA" />


        <LinearLayout
                android:id="@+id/zoom"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:layout_alignParentBottom="true"
                android:layout_centerHorizontal="true" />

</RelativeLayout>
```

Quadro 7 - Arquivo XML da tela que apresenta o mapa

5 CONCLUSÃO E TRABALHOS FUTUROS

5.1 CONCLUSÃO

O desenvolvimento deste trabalho proporcionou estudos sobre um dos principais sistemas operacionais móveis, desenvolvimento *web* com a linguagem Java e tecnologias necessárias para a integração de dois sistemas.

O sistema gerenciador de imóveis voltado para *internet* disponibiliza informações ao sistema móvel através de funcionalidades desenvolvidas ao longo deste projeto entre as quais pode-se citar gerência de imóveis, usuários, integração e visualização. A aplicação móvel por sua vez possui módulos de integração, gerência de favoritos, interpretação de *QR Code* e visualização.

Após a conclusão do desenvolvimento notou-se que busca de imóveis foi aperfeiçoada, proporcionando ao usuário uma experiência mais rica e prática, atingindo assim o principal objetivo deste trabalho.

Houveram algumas dificuldades durante o desenvolvimento, o primeiro problema foi referente a infra-estrutura. Foi imprescindível adquirir um servidor do tipo *cloud*, que possibilitou a instalação dos aplicativos servidores necessários como Apache Tomcat e PostgreSQL, uma vez que hospedagens com estes *softwares* são difíceis de encontrar e com muitas dificuldades para manutenção. Outra dificuldade foi a falta de material para o desenvolvimento com QR Code, tanto na leitura, como na geração do mesmo.

5.2 TRABALHOS FUTUROS

Visando aprimorar alguns aspectos deste trabalho algumas sugestões de melhoria foram identificadas, como por exemplo portar todas as funcionalidades do sistema móvel para outras plataformas populares como Windows e Apple.

Ainda, com a proposta de aumentar o público alvo, tem-se a sugestão de adaptar o sistema *web* de uma forma que mesmo seja capaz de incorporar as imobiliárias de uma cidade, disponibilizando assim um serviço mais completo ao usuário.

Existe também planos para a utilização de novos conceitos e tecnologias com *smartphones*, um deles é o NCF (*Near Field Communication*), onde é possível transformar um *smartphone* em um dispositivo de identificação, utilizado para realizar pagamentos e até mesmo como cartão ponto em empresas.

REFERÊNCIAS

BECK, Kent. **Test-Driven Development By Examples.** 1. ed. Three Rivers Institute, 2002.

DEITEL, Harvey M.; DEITEL, Paul J. **Java: Como Programar.** 6. Ed. Editora

DENSO WAVE. **About 2D Code.** Disponível em: <http://www.denso-wave.com/qrcode/aboutqr-e.html>. Acesso em: 1 jan. 2012.

FURGERI, Sérgio. et al. **Java 2 Ensino Didático.** 5. ed. Editora Érica, 2002, 49p.

GARTNER, Inc. **Worldwide Smartphone Sales to End Users by Operating System in 4Q11 (Thousands of Units).** Disponível em: <http://www.gartner.com/it/page.jsp?id=1924314>. Acesso em: 30 fev. 2012.

GONÇALVES, Edson. et al. **Dominando Eclipse.** 1 ed., Ciência Moderna, 2006, 22p.

HASHIMI, Sayed Y. **Pro Android.** 2. ed. New York: Apress, 2010.

HEUSER, Carlos A. **Projeto de Banco de Dados.** 6. Ed. Editora Bookman, 2008.

JENKOV, Jacob. **Java Servlets**. Disponível em: <http://tutorials.jenkov.com/java-servlets/index.html>. Acesso em: 10 abr. 2012.

LUCKOW, Décio H.; MELO, Alexandre A. **Programação Java Para Web.** 1. Ed. Novatec Editora, 2010.

MACORATTI. **UML - Diagrama de Classes e objetos.** Disponível em: <http://www.macoratti.net/net_uml1.htm>. Acesso em: 16 set. 2011.

MILANI, André. **PostgreSQL – Guia do Programador.** 1. Ed. Novatec Editora, 2008.

ROCHA, Luiz C. **Código de barras sem mistérios.** Disponível em: <http://msdn.microsoft.com/pt-br/library/cc580676.aspx>. Acesso em: 10 out. 2011.

TELES, Vinícius Manhães Teles. **Extreme Programming: Aprenda como encantar seus usuários desenvolvendo software com agilidade e alta qualidade**. 1. Ed. Novatec Editora, 2004.

THE POSTGRESQL GLOBAL DEVELOPMENT GROUP. **Manual do PostgreSQL 8.** 1. Ed. The PostgreSQL Global Development Group, 2006.

TIOBE SOFTWARE. **Programming Community Index for April 2012**. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 18 abr. 2012.

UCFG CEEI. **Material sobre UML**. Disponível em: <http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/diagramas/usecases/>. Acesso em: 12 fev. 2012.

ZXING. **ZXing ("Zebra Crossing")**. Disponível em: <http://code.google.com/p/zxing/>. Acesso em: 18 nov. 2011.