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

DANYEL DÊNNIS BEZERRA CARVALHO

AIMT: APLICATIVO DE IDENTIFICAÇÃO DE MOTO-TÁXI DA CIDADE DE PICOS

PICOS - PIAUÍ
2016

DANYEL DÊNNIS BEZERRA CARVALHO

AIMT: APLICATIVO DE IDENTIFICAÇÃO DE MOTO-TÁXI DA CIDADE DE PICOS

Monografia submetida ao Curso de Bacharelado em Sistemas de Informação, como requisito parcial para obtenção de grau de Bacharel em Sistemas de Informação.

Orientador: Prof. Esp. Ismael de Holanda Leal

PICOS - PIAUÍ
2016

C331a Carvalho, Danyel Dênnis Bezerra.

AIMT: Aplicativo de Identificação de moto-táxi da cidade de Picos / Danyel Dênnis Bezerra Carvalho.– 2016.

CD-ROM : il.; 4 ¾ pol. (44 f.)

Monografia (Curso Bacharelado em Sistemas de Informação) – Universidade Federal do Piauí, Picos, 2016.

Orientador(A): Prof. Esp. Ismael de Holanda Leal

1.Sistema *Web*. 2. Aplicativo-Moto-Táxi. 3. *Android*. I. Título.

CDD 005.1

APLICATIVO DE IDENTIFICAÇÃO DE MOTO – TAXI DA CIDADE DE PICOS

DANYEL DENNIS BEZERRA CARVALHO

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

Data de Aprovação

Picos – PI, 22 de FEVEREIRO de 2016

Prof. Esp. Ismael de Holanda Leal
Orientador

Leonardo Pereira de Sousa

Prof. Esp. Leonardo Pereira de Sousa
Membro

Patricia Vieira da Silva Barros

Prof. Esp. Patricia Vieira da Silva Barros
Membro

Dedico este trabalho, antes de tudo, a Deus. Em especial, também dedico a minha família. À minha mãe, Maria Neuma Bezerra Carvalho; ao meu pai, Antônio Daniel de Carvalho Oliveira; aos meus irmãos, Daniela Marina Bezerra Carvalho e Danilo Bezerra Carvalho Oliveira; à minha avó, Marina Antônia Bezerra e a minha amada namorada, Valdilene Maria de Jesus Sousa, que, com simplicidade, união e esforços me ajudou e me apoiou durante toda esta árdua caminhada. Diante das inúmeras dificuldades e obstáculos, por várias vezes pensei em desistir. No entanto, aprendi a visualizar os propósitos da vida e, com isso, passei a renovar minhas esperanças e acreditar que o grande sonho da formatura se concretizaria.

AGRADECIMENTOS

Agradeço primeiramente a Deus, que me deu saúde, força e muita fé para superar as dificuldades em todos os dias da minha vida.

Agradecer minha incrível família: Maria Neuma e Antônio Daniel (pais), Marina Antônia Bezerra (avó), Daniela Marina e Danilo Bezerra (irmãos). Obrigado pelo exemplo de honestidade e amor incondicional. Apesar das dificuldades, me fortaleceram e me apoiaram nas horas difíceis. Muito Obrigado!

Gostaria de agradecer a Valdilene, por toda paciência e sempre me fazer acreditar na realização dos meus sonhos, e ajudou muito para que eu pudesse realizá-los. Minha namorada, companheira na vida e nos sonhos, que sempre me apoiou nas horas difíceis e compartilhou comigo as alegrias.

A todos os professores que me acompanharam durante a graduação, pelos ensinamentos e conselhos que foram essenciais para minha formação.

Tenho muito que agradecer meus professores: Ismael de Holanda Leal, que me orientou neste trabalho, obrigado pela dedicação e incentivo, sem os quais não teria conseguido. A Leonardo Sousa, meu co-orientador. Obrigado pelo incentivo, ensinamentos, conselhos e por sua disponibilidade, sempre que precisei da sua ajuda. A Rayner Gomes, agradeço pela oportunidade que me destes. À Patrícia Medyna, pelo seu grande coração e por sua dedicação para tornar, tanto o Curso quanto os alunos, melhores a cada dia. Enfim, agradeço aos demais professores que, mesmo não sendo citados aqui, contribuíram muito para que eu pudesse completar esta caminhada.

Não posso esquecer os meus amigos, companheiros de vida e irmãos na amizade, que fizeram parte da minha formação da UFPI: Allan Jheyson, Jonnison Lima, José Erenildo, Wilson Sousa, Veron Batista, Micael Araújo, Thiago José, Cliciano Sabino, Denis Paiva, Matheus Rodrigues, José Victor, Kelly Oliveira, Reginaldo Francisco, Francilene Martins, Diego Feitosa, Auricélio Henry, Andrei Maxwel, Rafael Fernandes, Francisco Cavalcante, Gilmar Rodrigues, Acleirton Sá, Gleyson Lima e Salomão Júnior. Vocês vão continuar presentes em minha vida, com certeza.

"O covarde nunca tenta, o fracassado nunca termina e o vencedor nunca desiste."

Norman Vicent Peale

"A mente que se abre a uma nova idéia jamais voltará ao seu tamanho Original."

Albert Einstein

RESUMO

Atualmente, com a evolução das tecnologias houve um grande aumento na quantidade de informações disponibilizadas em meios eletrônicos, acarretando uma sobrecarga de conteúdo irrelevante na rede. Diante disso, houve um aumento significativo na procura por *softwares* para dispositivos móveis, como *smartphones, tablet*. O sistema aqui descrito foi desenvolvido para auxiliar o usuário na possibilidade de escolher os serviços de moto-taxistas, visto que, ao digitar o número do colete do moto-taxista e o número da placa da motocicleta, o Sistema informa a legalidade do moto-taxista. Com o uso deste aplicativo, a população do Município de Picos-PI, a qual utiliza o serviço de moto-táxi em grande massa, poderá escolher o moto-taxista com maior segurança e conforto durante a utilização deste serviço.

Palavras-chave: Moto-Táxi, *Android, Sistema Web,* Aplicativo Móvel, *Web Service*.

ABSTRACT

Now, with the evolution of technology there has been a large increase in the amount of information available in electronic media, leading to irrelevant content network overload. Thus, there was a significant increase in demand for software for mobile devices such as smartphones, tablet. The system described here was developed to assist the user the possibility to choose the moto-taxi drivers services, since when entering the motorcycle taxi driver vest number and the motorcycle license plate number, the system informs the legality of motorcycle cabby. With the use of this application, the population of the municipality of Picos-PI, which uses the motorcycle taxi service in great mass, you can choose the motorcycle-taxi driver with greater safety and comfort when using this service.

Keywords: Moto-Táxi, Android, Web System, Application Mobile, *Web Services.*

LISTA DE FIGURAS

LISTA DE ABREVIATURAS E SIGLAS

ADT	Android Development Tools
AIMT	Aplicativo Identificação de Moto-Taxista
AMPS	Advanced Mobile Phone System
ANATEL	Agência Nacional de Telecomunicações
API	Application Programming Interface
ASF	Apache Software Foundation
AVD	Android Virtual Device
CDMA	Code Division Multiple Access
CRUD	*Create, Read, Update e Delete*
DMT	Departamento Municipal de Transito
EDGE	Enhanced Data for GSM Evolution
FCC	Federal Communication Commission
GPRS	General Packet Radio Service
GSM	Sistema Global para Comunicação Móveis
GSP	Groovy Server Pages
HAPS	High Altitude Platform Station
HSPA	Acesso a pacotes em Alta Velocidade
HTTP	Hypertext Transfer Protocol
IDE	Integrated Development Environment
JSON	Java Script Object Notation
LED	Light Emitting Diodes
OHA	Open Handlest Alliance
Open	Handset Alliance (OHA)
PDAs	Personal Digital Assistant
SDK	Software Development Kit
SGBDOR	Sistema de Gerenciamento de Banco de Dados Objeto-Relacional
SMS	Serviço de Mensagens Curtas
TDMA	Time Division Multiple Access
SMP	Serviço Móvel Pessoal
SMC	Serviço Móvel Celular
URL	Uniform Resource Locator

FM	Frequency Modulation
URI	Uniform Resource Identifier
GHz	Gigahertz
MHz	Megahertz
Mbps	Megabit por segundo
Gbps	Gigabits por segundo

SUMÁRIO

1 INTRODUÇÃO

Nos últimos anos, tem-se presenciado um crescimento no desenvolvimento das tecnologias para dispositivos móveis, como *smartphones* e *tablets*, juntamente com o aumento nas vendas destes aparelhos, os quais incluem o serviço de ligações telefônicas, instalação e execução de aplicativos disponibilizados na *internet*. Para acompanhar tal crescimento, é importante que empresas fabricantes de dispositivos e aplicativos móveis, busquem ou desenvolvam novas tecnologias a fim de atender a demanda, além de conquistar a preferência de operadoras e usuários (LECHETA, 2013).

Diante disso, irá ser desenvolvido um sistema de identificação de moto-taxistas na Plataforma *Android*, por meio de informações de moto-taxistas cadastrados na associação de moto-taxistas de Picos, junto ao DMT (Departamento Municipal de Transito), as quais estarão disponíveis em um banco de dados *online*. Este banco será alimentado com base nas informações cedidas pelo órgão de expedição que faz a fiscalização e conterão dados como: o número de identificação do colete, o número da placa, nome, foto, idade, sexo e se o condutor está apto a desenvolver suas funções.

Dessa forma, o usuário, ao utilizar-se dos serviços do aplicativo para moto-táxi deverá, primeiramente, digitar o número do colete e da placa da motocicleta, de forma a obter informações essenciais, como: dados pessoais do moto-taxista e a sua situação legal no órgão que faz a fiscalização DMT. Com estas informações, o usuário estará mais seguro em utilizar os serviços do moto-táxi, bem como, o moto-taxista desempenhará suas funções de forma eficiente.

1.1 Objetivo

Desenvolver um aplicativo para identificação dos moto-táxi da cidade de Picos, em vistas a atender às necessidades dos usuários de dispositivos móveis, disponibilizando maior conforto e segurança para a população.

1.2 Organização do Trabalho

O trabalho está organizado em 5 capítulos. O capítulo 2 apresenta um breve histórico sobre a evolução do celular e da Plataforma *Android*, bem como sua importância na sociedade atual. Traz ainda o embasamento para a aplicação através de Referencial Teórico, abordando trabalhos e estudos já consolidados por outros autores e estudiosos na área *Android* e no *framework Grails*. No Capítulo 3, são abordados: o sistema, desenvolvimento e resultados, descrevendo alguns fatores que influenciaram na realização do estudo. No capitulo 4 tem-se os Resultados, além dos testes para demonstrar a viabilidade da utilização do *software* desenvolvido neste trabalho. O Capítulo 5, Conclusão, apresenta uma síntese do trabalho, abordando alguns aspectos que influenciaram o desenvolvimento do projeto.

1.3 Metodologia

O estudo foi realizado, baseado em informações de pesquisas bibliográficas e um estudo detalhado das ferramentas necessárias para a realização desta aplicação.

Com o aumento da concorrência entre as empresas, associado às crescentes exigências do mercado, no que se refere à qualidade no desenvolvimento de *softwares*, foi possível notar a necessidade da utilização de metodologias de desenvolvimento mais flexíveis e ágeis, objetivando diminuir o tempo de desenvolvimento e, ao mesmo tempo, mantendo a qualidade do produto (SOUSA; SOUZA, 2012).

2 REFERENCIAL TEÓRICO

No período de realização de um projeto, é importante utilizar alguns recursos para a obtenção de êxito. Quando se trata de uma aplicação para desenvolvimentos móveis, não há diferença, é possível lançar mão de diversas tecnologias para atingir o objetivo final. Neste capitulo serão abordados as principais tecnologias utilizadas neste projeto.

2.1 Histórico do *Android*

A primeira geração de telefones utilizando o sistema Android foi lançada em outubro de 2008. De acordo com a Gartner, As vendas de telefones baseados em Android nos Estados Unidos aumentaram 707% no primeiro trimestre de 2010, em relação ao anterior (GARTNER, 2010). Em março de 2011, um estudo da Nielsen mostrou que o Android tinha 37% da fatia de mercado de smartphones nos Estados Unidos, comparados com 27% do iPhone da Apple e 22% do Blackberry. Em agosto de 2010, mais de 200 mil smartphones Android eram ativados todos os dias, ultrapassando 100 mil por dia em apenas dois meses antes. Em junho de 2011, mais de 500 mil dispositivos Android eram ativados diariamente. Atualmente, existem mais de 300 dispositivos Android diferentes em todo o mundo (DEITEL; DEITEL; MORGANO, 2012).

O sistema operacional (SO) *Android* é uma plataforma desenvolvida pela Google voltada para dispositivos móveis. Em 5 de novembro de 2007, a empresa tornou pública a primeira plataforma Open Source de desenvolvimento para dispositivos móveis baseada na plataforma Java, com sistema operacional Linux, na qual foi chamada de *Android*. Essa plataforma é mantida pela A Open Handset Alliance (OHA), que é um grupo formado por gigantes do mercado de telefonia de celulares liderados pelo Google. Entre alguns integrantes do grupo estão nomes consagrados como a HTC, LG, Motorola, Samsung, Sony Ericsson, Toshiba, HTC, Huawei, Sprint Nextel, China Mobile, T-Mobile, ASUS, Intel, Acer, Dell, Garmin. Pode ainda ser utilizada linguagem Java para o desenvolvimento de aplicações (LECHETA, 2013).

Essas características somam, também, vantagens para os fabricantes de celulares, uma vez que é possível utilizar o sistema operacional do *Android* em seus aparelhos sem custos adicionais. Além disso, a licença *Apache Software Foundation* (ASF) permite que alterações sejam efetuadas no código-fonte para criar produtos customizados sem precisar compartilhar as alterações (LECHETA, 2013).

O *Android* tem muitos diferenciais interessantes e uma arquitetura realmente flexível, focada na integração de aplicações. Não existe diferença entre uma aplicação nativa e uma desenvolvida por outra pessoa (LECHETA, 2013).

Ao se levar em consideração o grande número de integrantes de peso do grupo OHA, amplas inovações, além da aceleração no desenvolvimento de aplicações, serviços, trazendo aos consumidores uma experiência mais rica em termos de recursos, com custos mais acessíveis para o mercado móvel, é possível afirmar que a plataforma *Android* é a primeira completa, aberta e livre para dispositivos móveis.

2.1.1 Celulares: História e Evolução

Na década de 40 a empresa Americana *Bell Company*, utilizando o conceito de células, desenvolveu um sistema que tinha como premissa utilizar a telefonia móvel dentro de determinada área geográfica, o que, o uso do conceito de células, serviu como inspiração para atrair tecnologias desenvolvidas, especialmente, para celulares.

Outro marco importante incidiu nessa mesma década, quando as empresas Americanas AT&T e Bell propuseram a criação da FCC (Federal Communication Commission), o que proporcionou a alocação de um número de frequência de rádio criado, especialmente, para a comunicação móvel. Inicialmente, a FCC disponibilizou um número limitado de frequências, o que, na época, contribuiu para o discurso da inviabilidade comercial desta nova tecnologia.

Foram necessários cerca de 20 anos para a FCC aumentar a quantidade de frequências destinadas à telefonia móvel. No final da década de 60, as empresas Americanas AT&T e Bell começaram a utilizar o sistema com uso de torres, objetivando aumentar a cobertura de usuários por áreas, o que culminou na realidade observada atualmente.

Apesar da empresa Bell ter criado o sistema de comunicação instalado em carros de polícia, a Motorola foi a primeira a incorporar essa tecnologia a dispositivos móveis de comunicação fora de veículos, visando serem utilizados para o uso pessoal. Entretanto, só na década de 1980 surgiu o primeiro celular aprovado pelo FCC: o DynaTAC 8000X, da Motorola. Esta, junto com a empresa Ameritech, iniciaram o uso comercial da telefonia celular no Estados Unidos e, posteriormente, expandiram para o Mundo.

O Aparelho criado pela Motorola pesava cerca de 1kg, possuía capacidade limitada e preço exorbitante, quando comparado aos preços dos modelos atuais. Já se passaram mais de 30 anos desde o início da telefonia móvel comercial e, desde então, os aparelhos celulares passaram a assumir uma ‘‘personalidade’’ parecida com a dos seus donos, enriquecendo relacionamentos, entretendo, expressando individualidade e expandindo o conceito de interconectividade, a partir do uso da internet, através dos dados móveis.

Atualmente, existem cerca de 1,2 bilhões de usuários de celular. O grande aumento do número de usuários aconteceu devido ao crescente barateamento da tecnologia, como também às inovações que permitiram à tecnologia do celular agregar uma quantidade bem maior de funcionalidades, como o acesso à internet, jogos, músicas e imagens digitais. Nessa mesma perspectiva, as indústrias de tecnologias estão buscando cada vez mais o crescimento por meio da oferta de inovações voltadas às necessidades dos usuários.

E, com o aumento do número de usuários e de indústrias de tecnologias especializadas em telefonia móvel há, consequentemente, uma gama maior de empresas que inovam e procuram revolucionar os serviços com acesso à internet sem fio, jogos e aplicativos.

2.2 Gerações dos sistemas de celulares

2.2.1 Sistema de 1ª Geração - 1G

O sistema de 1 geração são analógicos, tem limitações a cada usuário realizar sua alocação de frequência especifica. Qualidade do serviço e inferior por isso o desenvolvimento de novas tecnologias. Os sistemas analógicos podem receber infinitos valores dentro de uma faixa possível.

Os Sistemas 1G possuíam um grande número de padrões fato que motivou o desenvolvimento dos sistemas 2G, a transmissão de voz analógica dos sistemas 1G utilizava a modulação FM (Frequency Modulation).

2.2.2 Sistema de 2ª Geração - 2G

Começou o surgir os sistemas digitais, mas com algumas limitações só podia assumir valores finitos dentro de uma determinada faixa, foram lançadas as principais tecnologias digitais: GSM, CDMA IS 45 E TDMA IS-136. Com estas evoluções os sistemas aumentarão suas capacidades de conexões de dados com transferências de 14 Kbps.

Observar que as gerações dos sistemas móveis teve uma evolução entre a 2 geração e a 3 geração. Com o surgimento de novas tecnologias GPRS (*General Packet Radio Service*) que permitiu os aparelhos móveis acessar internet de forma mais rápida, e assim aplicando seus serviços como SMS, GPS e jogos.

GSM e o CDMA com estas novas tecnologias surgiu oferta de serviços de dados por pacotes sem necessidades de estabelecimento de uma conexão.

2.2.3 Sistema de 3ª Geração - 3G

O sistema de 3G foi lançado no Japão em 2001, utilizando a tecnologia WCDMA. No Brasil a Agência Nacional de Telecomunicações (ANATEL) atribuiu às faixas de frequências de 1.885 – 1900 MHz, 1.950 – 1980 MHz e 2.140 – 2170 MHz para operação do sistema. Sua principal característica de funcionar sem estabelecer uma conexão permanente e com taxas de até 2 Mbps.

Ainda de acordo com a tecnologia WCDMA, há uma geração de sistema, para dispositivos moveis que oferece serviços por pacotes e taxas maiores que 256 Kbps.

2.2.4 Sistema de 4ª Geração – 4G

A quarta geração de dispositivos móveis no Brasil, foi oferecer serviços de multimídia de banda larga para os usuários, prometendo taxas de transmissão de dados muito alta, que poderia chegar até 2 Mbps em seus veículos e de até 20 Mbps em ambientes internos.

> Utiliza muitos recursos e implementações do modelo 3G, com capacidade de transmissão de dados por comutação de pacotes, aproximando assim ao funcionamento das redes de dados da computação. Mais uma vez a cobertura será a limitação a ser vencida, por isso deverá haver sistemas operando nas redes 3G e 4G, em um modelo de multimodo. Já pelo lado da infraestrutura física está sendo desenvolvida para aplicação ainda no 3G, a instalação do HAPS (High Altitude Platform Station), que seriam estações de altitude elevada, conforme o projeto a altitude é de 20 a 50 Km fixo em relação à terra, com a finalidade de prover grande área de cobertura (BOLZANI, 2004).

Devidos a altos custos dos serviços de satélite, apareceram novas ideais de utilizar outras ferramentas outros veículos aéreo tais como balões, dirigíveis, aviões específicos ou similares que devem operar em altitudes entre 3 a 50 km, com pretensão de cobrir muitos 1.000 km de diâmetro.

> A partir de junho de 2000, a ANATEL lançou um novo serviço para comunicação móvel, o SMP (Serviço Móvel Pessoal), este passou a operar na faixa de frequência de 1,8 GHz. Mesmo com o lançamento desse novo modelo, o SMC (com frequência de 0,8 GHz) ainda deverá funcionar paralelamente. Porém, a Agência tem a intenção de que haja uma migração gradativa para o novo serviço. Com a regulamentação do SMP, e a disponibilização de uma nova faixa de frequência, que na época do lançamento foi denominada Banda C, fez surgir mais um grupo de empresas que operam nessa nova faixa de frequência (de 1,8 Ghz) (VARGAS, 2002).

Este sistema teve menos custo na transmissão de dados, e ainda fez redução na latência, possui maior eficiência com largura de banda. Foi projetada para oferecer para seus usuários uma grande taxa de downloads de ate 100 Mbps para usuários em movimentos e de 1 Gbps para usuário parados.

2.3 Desenvolvimentos para Dispositivos Móveis

O número de usuários de dispositivos móveis (*tablet*, *smartphones*, PDAs, entre outros) cresce a cada dia, incentivado pela maneira como os fabricantes vêm oferecendo aparelhos cada vez mais completos (CRUZ et al., 2011). Estes dispositivos possuem diversas utilidades, como *wi-fi*, para transmissão de informações sem fio, e uso do sistema global de posicionamento (GPS).

Com o crescimento destas tecnologias, estão surgindo diversos aplicativos com o objetivo de melhor atender às necessidades dos usuários, dando-lhes conforto e agilidade em suas atividades do dia a dia, tornando-se, desse modo, uma ferramenta de auxílio, devido à existência de uma plataforma de alta qualidade para desenvolvimento dessas aplicações, a chamada Plataforma *Android*.

2.3.1 Plataforma *Android*

É um sistema operacional de código aberto próprio para dispositivos móveis, "baseada no sistema operacional *Linux* e com um ambiente de desenvolvimento flexível e poderoso" (SILVA; BRACHT, 2010). Ela surgiu com objetivo de resolver diversos problemas, como desenvolver aplicativos para dispositivos móveis, além de ser de código aberto. Por isso ela chamou muita atenção dos desenvolvedores por ser uma ferramenta de múltiplos recursos.

O *Android* causou grande impacto quando foi anunciado, devido ao fato de ter a *Google* como empresa responsável por sua criação. Entretanto, a responsabilidade pelo SO não cabe apenas àquela empresa, uma vez que também estão envolvidas várias empresas líderes do mercado de telefonia como a Motorola, LG, Samsung e Sony Ericsson. Esse grupo, chamado de *Open Handset Alliance (OHA)* foi criado com o objetivo de padronizar uma plataforma de código aberto e livre para celulares, justamente para atender a todas as expectativas e tendências do mercado atual. (LECHETA, 2013).

Vale salientar que, além de ser um SO de código aberto, o *Android* disponibiliza algumas alternativas de linguagem de programação para seus desenvolvedores, como a C/C++, .NET *Framework*, *Scala*, Lua, *Groove*, *Python e Java*.

2.3.2 O SDK do *Android*

O *Software Development Kit* (SDK) do *Android*, disponível gratuitamente no site *Android Developers*[1], fornece as ferramentas necessárias para a construção de aplicativos *Android*, incluindo *o Java SE*, o *IDE Eclipse*, o SDK 3.x do Android (DEITEL; DEITEL; MORGANO, 2012).

2.3.3 *Android Studio*

Android Studio é um novo ambiente de desenvolvimento para *Android*, com base no *Intellij IDEA*. De forma semelhante ao *Eclipse*, aliado ao *plugin* ADT (*Android Development Tools*), o *Android Studio* fornece ferramentas de desenvolvimento integradas *Android*, para desenvolvimento e depuração. Em junho de 2013, ele foi disponível para os usuários, na forma de beta. Esta plataforma é a mais recomendável para o desenvolvimento de aplicativos para *Android*, pois possibilita a instalação de *Plugin* ADT, proporcionando o gerenciamento das informações e melhorando o desenvolvimento dos aplicativos (GETTING. 2013).

2.4 *Android Development Tools (ADT)*

O *Plugin* ADT (*Android Development Tools*) para *Eclipse* – uma extensão para o *IDE Eclipse* – permite criar, executar e depurar aplicativo *Android*, exportá-lo para distribuição (por exemplo, carregá-los no *Android Market*) e muito mais. O ADT também contém uma ferramenta de projeto visual de interface gráfica do usuário. Os componentes da interface gráfica do usuário podem ser arrastados e soltos no lugar para formar interface, sem nenhuma codificação. (DEITEL; DEITEL; MORGANO, 2012). Esse *plugin* permite o desenvolvimento de aplicações para o *Android* de forma simples e integrada, possibilitando a criação, construção, o empacotamento, a instalação, depuração e teste das aplicações *Android* de modo rápido e fácil.

[1] www.developer.android.com

2.4.1 O *Emulador* do *Android*

O *Emulador* do *Android*, incluído no SDK do *Android*, permite executar aplicativos *Android* no ambiente simulado dentro dos sistemas operacionais *Windows, Mac OSX ou Linux*. O emulador exibe uma janela de interface de usuário realista. Antes de executar um aplicativo no emulador, é necessário criar um AVD (*Android Virtual Device*, ou Dispositivo *Android Virtual)*, o qual define as características do dispositivo em que se deseja realizar o teste, incluindo o *hardware*, a imagem do sistema, o tamanho da tela, o armazenamento de dados e muito mais. Caso seja necessário testar seus aplicativos em vários dispositivos *Android*, será necessário criar AVDs separados para emular cada dispositivo exclusivo (DEITEL; DEITEL; MORGANO, 2012).

2.4.2 *Framework Grails*

O *framework* é uma abstração que une códigos comuns entre vários projetos de *software* provendo uma funcionalidade genérica (*MÜLLER*, 2008).

Criado em 2006, o *Grails* é um *framework* de desenvolvimento *Java Web* de última geração que se baseia nas melhores ferramentas, técnicas de desenvolvimento *web* e tecnologias de *frameworks Java* já existentes, combinando-os com o poder e inovação do desenvolvimento da linguagem dinâmica (SMITH, 2009). Quando o *Grails* foi lançado, já existiam diversos *frameworks* ativos no mercado, porém o *Grails* não "reinventou a roda", mas que, ao invés disso, tem se aproveitado dos *frameworks* já testados e confiáveis tais como *Spring*, *Hibernate*, *SiteMesh* e outras bibliotecas de código aberto já populares no mundo empresarial *Java (KUMAR*, 2013).

Framework Grails é uma ferramenta de desenvolvimento ágil que atua sobre a linguagem *Groovy*. Sendo desenvolvida *sobre Java*, ela facilita o desenvolvimento de aplicações, simplificando comandos e estruturas que em *Java* normalmente são complexas.

O foco do *Grails* é manter uma alta qualidade do seu *software*, visto que é possível desenvolver aplicações, realizando os testes unitários e de integração, que são muito importantes para manter a qualidade do *software*.

Grails é a próxima geração de *Java framework* de desenvolvimento *web* que gera grandes ganhos de produtividade através da confluência de uma linguagem dinâmica.

2.4.3 Linguagem *GROOVY*

A Linguagem *Groovy* é uma forma facilitada da linguagem *Java*, contendo os mesmos recursos, porém com uma implementação simplificada, permitindo que o processo de desenvolvimento ganhe produtividade e agilidade.

Com o lançamento do *framework Grails*, para desenvolvimento de aplicações *web*, *Groovy* ganhou credibilidade e o interesse da comunidade *Java*, provando assim, que o desenvolvimento de aplicações *web* escaláveis são produtivas e fáceis. *Grails* utilizou desde recursos básicos da *Groovy*, até recursos complexos de aplicações *web*, como persistência em banco de dados, *Ajax*, *web services*, relatórios, processamento em lote e *plugins* que permitem aos desenvolvedores melhorar e criar novas ferramentas que auxiliam o desenvolvimento (JUDD; NUSAIRAT; SHINGLER, 2008).

2.5 *WEB SERVICE*

Aplicações corporativas geralmente precisam se comunicar com algum *servidor* para realizar o sincronismo das informações. Essa comunicação pode ser feita de diversas formas, utilizando o protocolo HTTP. Atualmente *Web Service* é uma das tecnologias mais utilizadas para integrar aplicações. O problema é que o *Android* não tem nenhuma API nativa para acessar *Web Service*. Porém, é possível utilizar qualquer outra biblioteca e incorporá-la ao projeto, normalmente. A biblioteca para acessar *Web Service* precisa ser leve e compacta, uma vez que a aplicação vai ser executada em um dispositivo móvel, que não tem o mesmo poder de processamento de um computador normal (LECHETA, 2013).

As requisições de dados pela *Web Services* que se comunicarem com o servidor, em um determinado serviço ou recurso, podem ser disponibilizadas pelo servidor através do URI. A partir deste URI é o cliente pode acessar o serviço desejado em diferentes tipos de representações, ficando a cargo a decisão pela representação desejada.

2.5.1 Banco de dados *POSTGRESQL*

O programa *Postgresql* é um sistema de gerenciamento de banco de dados objeto-relacional (SGBDOR) baseado no gerenciador de banco de dados *Postgresql*, que fora desenvolvido no departamento de Ciências da Computação, da Universidade da *Califórnia* em Berkeley, sendo distribuído em regime *open source* como uma alternativa aos sistemas de gerenciamento de banco de dados *Oracle ou informix* (MANZANO, 2008). O *Postgresql* será o *servidor* que receber as informações do aplicativo e estas informações vão ser transmitidas por um canal de transferências de dados, sendo sincronizadas até o *Postgresql* pela ferramenta que fará estas transferências e sincronização com o *Web Services*.

3 APLICATIVO PARA MOTO-TAXISTAS

Esse capítulo mostra, especificadamente, os problemas solucionados pelo aplicativo de identificação de Moto-Taxi da cidade de Picos, com o objetivo de proporcionar conforto e segurança aos usuários na hora de escolher o moto-taxista. A Proposta deste trabalho é especificar os problemas enfrentados e os meios utilizados para solucioná-los.

Neste capítulo são detalhados os processos de desenvolvimento do aplicativo em *Android* e também do *software Grails*, para suas instalações, configurações e comunicação.

Foi realizado o estudo sobre a plataforma *Android*, além da instalação dos *plugins* necessários, como por exemplo: IDE *(Integrated Development Environment)*, Desenvolvimento da ferramenta *Android Studio*, o SDK e o *plugin* ADT (*Android Development Tools*), fornecido pela empresa *Google*. Depois das instalações, foi realizada a configuração virtual do emulador AVD (*Android Virtual Device*) para que o simulador pudesse executar o sistema em um dispositivo móvel.

3.1 Instalação e Configuração dos *Softwares*

No desenvolvimento do aplicativo foi utilizado o Sistema Operacional *Windows*, pois este proporciona maiores facilidades no momento da instalação do SO, assim como dos *softwares* que foram utilizados. São itens necessários à instalação: configuração de bibliotecas *e plugins* para a implementação do sistema, detalhados a seguir:

- *Grails: framework web* é uma ferramenta para desenvolvimento da Linguagem *Groovy*, que se configura em um melhoramento do *Java*, o qual se comunica com facilidade com a plataforma *Android Studio*, que é mais robusta é menos estável que o *IDE Eclipse*, o que justifica a sua utilização neste trabalho.
- SDK (*Software Development Kit*): trata-se de um kit de desenvolvimento de aplicativos da *Microsoft*, que inclui, dentre outros, documentação, código e utilitários para que programadores consigam desenvolver aplicações de acordo com um padrão de desenvolvimento para o sistema operativo em questão. Sendo este utilizado para a implementação do aplicativo.

- ADT (*Android Development Tools*): é um *plugin* que contém ferramentas para facilitar a criação de projetos no *Android*, o qual foi utilizado no desenvolvimento do aplicativo, sendo instalado no *Android Studio*.

Inicialmente, para a instalação da plataforma *Android Studio* e do kit de desenvolvimento SDK, também foi necessário a configuração do emulador AVD, responsável por carregar a imagem do sistema e simular *hardware* e *software* de um sistema *Android*, tendo a seguinte configuração: versão 5.0.1 (API level 21) do Sistema Operacional *Android*, *Screen* 4.7" (correspondente ao tamanho da tela do dispositivo), resolução (px) 480 x 800 e memória RAM de 1024 MB.

3.2 Funcionamentos do Sistema AIMT

O sistema de moto-taxista foi desenvolvido para plataforma *Android* com intuito de atender aos usuários da cidade de Picos, que eventualmente precisem dos serviços destes profissionais. No aplicativo estarão disponíveis as informações pessoais e necessárias para que os usuários realizem a escolha do prestador de serviços que mais se adeque às suas necessidades.

A Figura 1 ilustra seu protótipo visual do *layout* e a funcionalidade da aplicação.

Figura 1 – Tela Inicial do Sistema AIMT.

O sistema consiste em exportar as informações dos moto-taxistas que são armazenadas na base de dados para, posteriormente, serem acessadas remotamente.

Ao executar a aplicação, serão exibidas, inicialmente, informações sobre o sistema. A tela principal da aplicação é composta por duas “imagens de texto” e dois “editar texto”. Através destes “editar texto”, o usuário digitará o número do colete do moto-taxista e o número da placa da moto, para que eles possam ser enviados ao servidor web. Além desses dois componentes, foi criado ainda um botão responsável por ativar a ação de enviar os dados para o servidor web.

Posteriormente será exibida a tela informando o processo de armazenamento das informações dos moto-taxistas. Ao instalar o aplicativo no dispositivo móvel, será apresentada a tela inicial e suas funcionalidades, assim como o arquivo principal, e o arquivo do moto-taxista. Ele possui apenas uma imagem para mostrar a foto 3x4 do moto-taxista em questão, os dados do moto-taxista (nome, colete, placa, situação e etc.) e um botão para voltar à tela anterior, como ilustrado na Figura 2.

.

Figura 2 - Layout características dos moto-taxistas

Ao realizar as requisições ao sistema, no seu *layout* do moto-taxista, o usuário digita o número do colete e da placa da moto. No emulador do *Android*, deve-se digitar os dados da busca e, posteriormente, acionar o botão “analisar” para fazer as requisições ao

servidor através do *web Service* e, assim, encontrar as informações do moto-taxista solicitado, conforme a Figura 3.

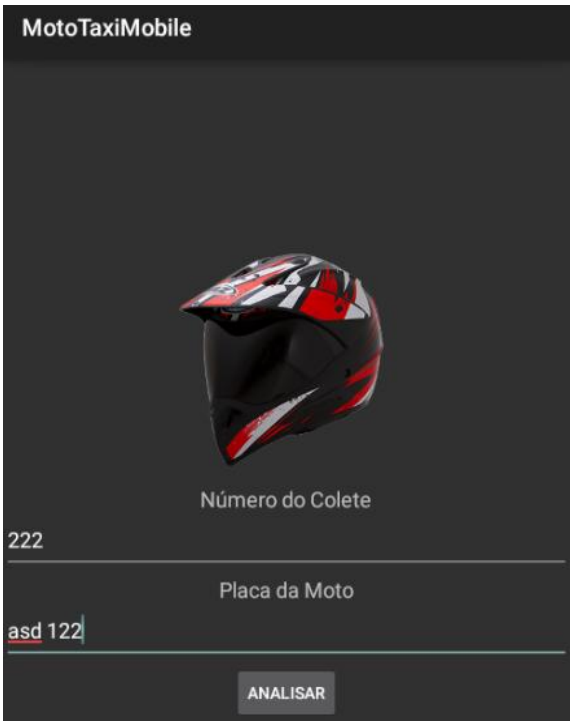

Figura 3 - *Layout* fazer buscar do moto-taxista.

O sistema, funcionando perfeitamente, listou as informações do moto-taxista. Agora o usuário com as informações disponíveis pode fazer sua escolha por optar ou não pelo moto-taxista em questão, como mostrado na Figura 4.

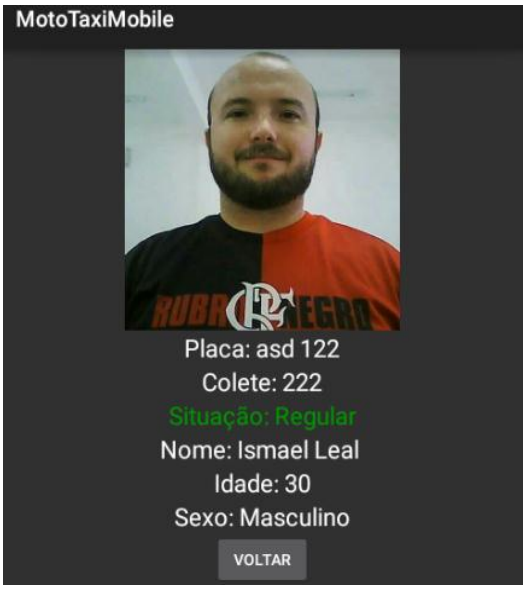

Figura 4 – Layout do resultado da busca

4 DESENVOLVIMENTO DO SISTEMA AIMT

Inicialmente, para criar a aplicação *web*, é necessário acessar o diretório, através do *prompt* de comando, onde se deseja que os arquivos sejam criados pelo *Grails*. O projeto foi criado no disco C:\, na pasta *Grails*. Acessando o diretório do disco C:\, através do *prompt* de comando, utilizando o comando "cd *Grails"*.

Em seguida, foi criada a aplicação, executado o comando "*grails create-app* nome_da_aplicação" e o nome da aplicação que se desejava criar. Por exemplo, "*grails create-app* Moto-Táxi", então uma pasta chamada Moto-Táxi foi criada no diretório "Documento". Nesta pasta estavam contidos todos os arquivos da aplicação *Grails*.

Criada a aplicação *Grails*, é necessário que o diretório onde a aplicação foi criada seja acessado. Para isso foi utilizado o mesmo comando já citado, para acessar a pasta Moto-Táxi: cd Moto-Táxi.

Acessado então o diretório já é possível iniciar a aplicação através do comando "*grails run-app*", que é o comando que inicializar a aplicação no servidor *Tomcat*. O *Grail*s então compila seus arquivos e coloca o servidor para rodar na URL "localhost: 8080/Moto Taxi".

Em seguida, foram criadas as classes de domínio, onde se declaram os atributos que foram persistidos ou armazenados no banco de dados. Estas classes de domínio utilizam a linguagem *Groovy*, que é uma linguagem muito semelhante ao *Java*, porém com pequenas melhorias. Devido a isto, as classes de domínio são criadas com extensão *"Groovy*".

Para realizar as classes de domínio foi utilizado o editor de texto *Gedit*. Acessando o diretório onde se encontravam as classes de domínio através do *Google Chrome* e a classe Moto-taxista, através do editor de texto. A regra é a mesma, primeiro definir os atributos, que neste caso são: nome, e-mail, usuário e senha e posteriormente as *constraints.*

Abrindo os controladores, utilizando um editor de texto *Gedit* e verificando que eles possuem uma grande quantidade de código já pré-definido. Este código é responsável pelo CRUD (*Create, Read, Update e Delete*), que são as quatro operações básicas de um objeto: criar, mostrar, atualizar e excluir.

As visualizações são na verdade as páginas que os usuários verão na tela do sistema. Estas páginas são as GSP (Groovy Server Pages), que contem código HTML, mas exclusivas do *framework Grails*. Vale lembrar que estas páginas são geradas de acordo com os atributos e *constraints* declaradas nas classes de domínio.

Rodando a aplicação utilizando o comando que já foi citado "*Grails run-app*" e acesse o link gerado pelo *Grails* onde a aplicação esta rodando conforme a Figura 5 (geralmente *localhost*:8080/MotoTaxi). A aplicação já é capaz de executar as operações do CRUD.

Figura 5 – Administrador Listagem

Para acessar todas as funcionalidades, o usuário deve clicar no respectivo controlador apresentado na página inicial da aplicação e navegar pelas páginas através dos menus. Através disso, as funcionalidades da aplicação, de acordo com a Figura 6, serão acessadas.

Figura 6 - Criar Administrador

As senhas são armazenadas no banco de dados em forma de *Hash*, que é um tipo de *criptografia*[2]. Esta funcionalidade é programada para que o sistema *web* fique mais seguro

[2] Técnicas criptográficas permitem que um remetente disfarce os dados de modo que um intruso não consiga obter nenhuma informação dos dados interceptados (KUROSE; KEITH, 2006).

e também mais confiável, de modo que as senhas sejam armazenadas de forma *criptografada.* Assim, mesmo que alguém tenha acesso ao banco de dados, as senhas dos usuários não serão reveladas (figura 7).

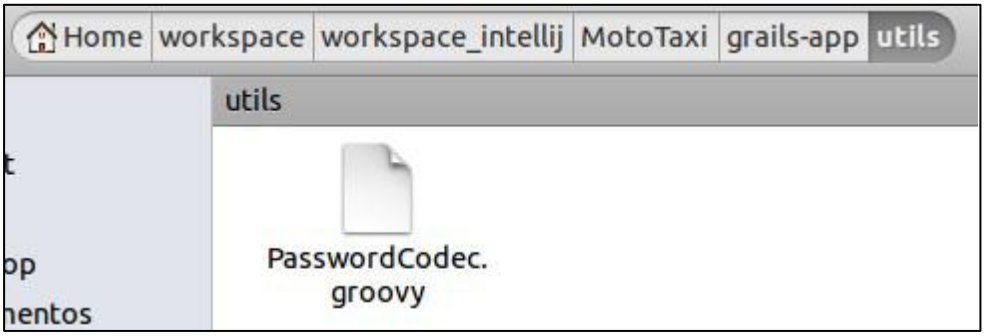

Figura 7 - Criptografia

Agora o sistema esta funcionando a parte de *criptografia* de senha. Ao acessar a página apresentada na figura e preencher os dados, são enviados para a ação "*save*", incluindo o campo senha. Este campo então é recebido, *criptografado* e armazenado no banco de dados pela própria ação *save* (Figura 8).

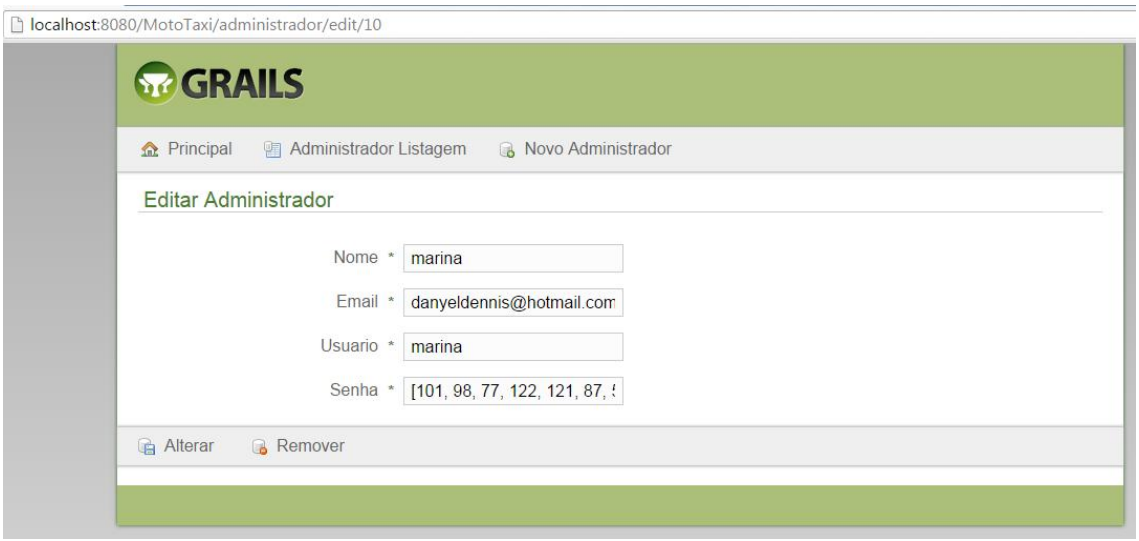

Figura 8 – *Criptografada*

Já que foram criados os campos usuário e senha o sistema contemplará uma função de login, como mostra a Figura 9. Esta função não é gerada pelo Grails e sim criada manualmente na linguagem Groovy.

Framework web Grails ele cria o CRUD, que são as quatro operações básicas de um objeto: criar, mostrar, atualizar e excluir

Figura 9 - *Login*. gsp

Na Figura 10, tem-se o *Login* do Administrador, o qual é responsável por mapear uma URL específica, para sua página que foi desenvolvida como (restrito. gsp). Neste caso, será acessada através de *"localhost*: 8080/MotoTaxi/restrito". Agora sendo inicializada sua aplicação, você poderá acessar a página de *login* do administrador.

Cadastre um administrador e perceba que ele está sendo autenticado normalmente no sistema.

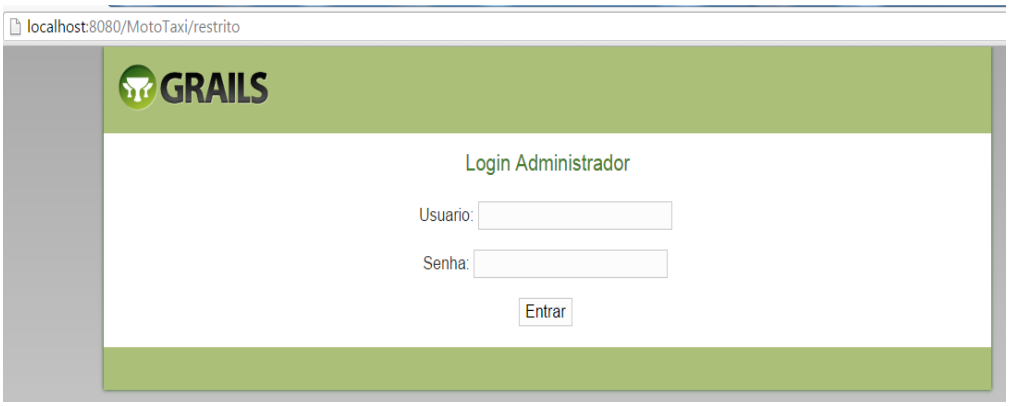

Figura 10 - *Login* Administrador

Na Figura 11 tem-se o *Login* do Moto-taxista, que é responsável por mapear uma URL específica para sua página que foi criada *(Login*. gsp), no caso do *Login* será acessada através de *"localhost:* 8080/MotoTaxi/*login* ". Agora sendo inicializada sua aplicação, você poderá acessar a página de *login* do Moto-Taxista.

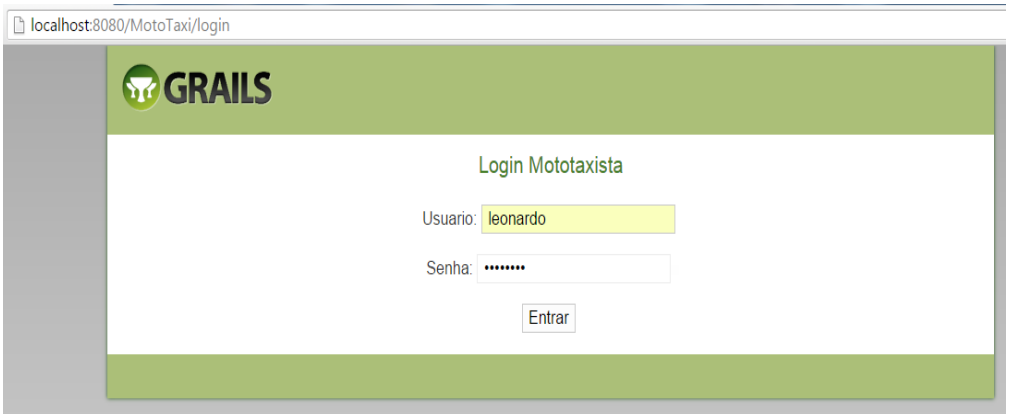

Figura 11- *Login* Moto-taxista

Cadastre um Moto-taxista e perceba que ele está sendo autenticado normalmente no sistema, como mostra a Figura 12, apresentando o *login* de Moto-Taxista.

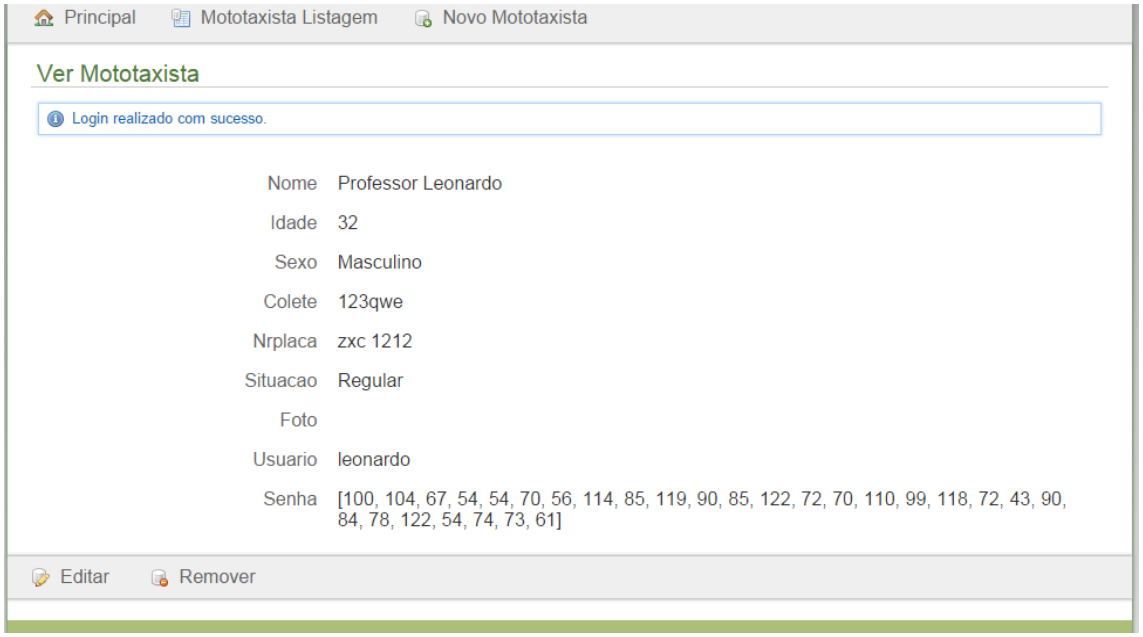

Figura 12 - Dados do Moto-Taxista

Uma última funcionalidade do sistema que é bastante importante é a de disponibilizar os dados dos moto-taxistas cadastrados em formato JSON para que possam ser utilizados na aplicação móvel, de acordo com a Figura 12 acima.

A partir disso, já temos as páginas de *login* e as ações de *login* para moto-taxista e administrador funcionando.

Código desta aplicação “JSONObject dados = request.JSON” é a que recebe os dados (numero da placa e colete) provindos da aplicação móvel e necessários para realizar a consulta no banco de dados e devolver as informações de um determinado moto-taxista, conforme a Figura 13.

Após receber o JSON, os dados do colete e placa são separados em variáveis diferentes para que se possa utilizar novamente o *finder* dinâmico do *Grails* e buscar o moto-taxista através do seu colete e placa. Caso ele seja encontrado, ele é então encapsulado em novo JSON e retornado a aplicação móvel. Caso contrário é retornado então “*null*” e a aplicação informa ao usuário que o moto-taxista não foi encontrado no banco de dados.

```
def getMototaxista() {
    JSONObject dados = request.JSON
    def placa = dados.get("placa")
    def colete = dados.get("colete")

    def mototaxistaInstance = Mototaxista.findByNr_placaAndColete(placa, colete)
    if (mototaxistaInstance) {
        JSONObject mototaxistaJSON = new JSONObject()
        mototaxistaJSON.put("placa", placa)
        mototaxistaJSON.put("colete", colete)
        mototaxistaJSON.put("nome", mototaxistaInstance.nome)
        mototaxistaJSON.put("idade", mototaxistaInstance.idade)
        mototaxistaJSON.put("sexo", mototaxistaInstance.sexo)
        mototaxistaJSON.put("situacao", mototaxistaInstance.situacao)
        mototaxistaJSON.put("foto", mototaxistaInstance.foto.encodeBase64())
        println mototaxistaJSON
        render mototaxistaJSON
    } else
        render null
}
```

Figura 13 – Código comunicação Web Services

DataSource.groovy

```
dataSource {
    pooled = true
    driverClassName = "org.postgresql.Driver"
    dialect = "org.hibernate.dialect.PostgreSQLDialect"
    username = "seu_usario"
    password = "sua_senha"
}
hibernate {
    cache.use_second_level_cache = true
    cache.use_query_cache = false
    cache.region.factory_class = 'net.sf.ehcache.hibernate.EhCacheRegionFactory'
}
// environment specific settings
environments {
    development {
        dataSource {
            dbCreate = "update"
            url = "jdbc:postgresql://localhost:5432/MotoTaxi"
        }
    }
    test {
        dataSource {
            dbCreate = "update"
            url = "jdbc:postgresql://localhost:5432/MotoTaxi"
        }
    }
    production {
        dataSource {
            dbCreate = "update"
            url = "jdbc:postgresql://localhost:5432/MotoTaxi"
            pooled = true
            properties {
                maxActive = -1
                minEvictableIdleTimeMillis=1800000
                timeBetweenEvictionRunsMillis=1800000
                numTestsPerEvictionRun=3
                testOnBorrow=true
                testWhileIdle=true
                testOnReturn=true
                validationQuery="SELECT 1"
            }
```

Figura 14 – Código comunicação com *Postgresql*

Para configurar o *Grails* para trabalhar com o *Postgresql* é bem simples. Basta fazer o *download* do *drive* de *conexão* do *Postgresql* e copiá-lo para a pasta "Mototaxi/*lib*" Após isso você deve acessar a classe *"DataSource*" que fica em "Mototaxi/*grails-app/conf*" e alterá-lo de modo que fique semelhante a Figura 14.

O *Grails* cria automaticamente as tabelas do banco de dados bem como seus relacionamentos, porém não cria o banco em si. Portanto, criei apenas o banco de dados no *Postgresql* e o *Grails* se encarrega de criar as tabelas e relações. Neste caso o banco criado é o "Moto-Táxi"

4 RESULTADOS

Foram realizados vários testes para demonstrar a viabilidade da utilização do *software* que foi desenvolvido neste trabalho. Os testes foram realizados no *Emulador do Android* para mostrar o funcionamento do Aplicativo e se podia ou não se tornar uma possível solução para o objetivo que foi proposto.

Testes de funcionalidade do sistema, todos os usuários utilizar o aplicativo e seus procedimentos de realização de uma buscar das informações do moto-taxista. Na imagem 15 o *emulador AVD* esta começando seu funcionamento carregando os *drives*.

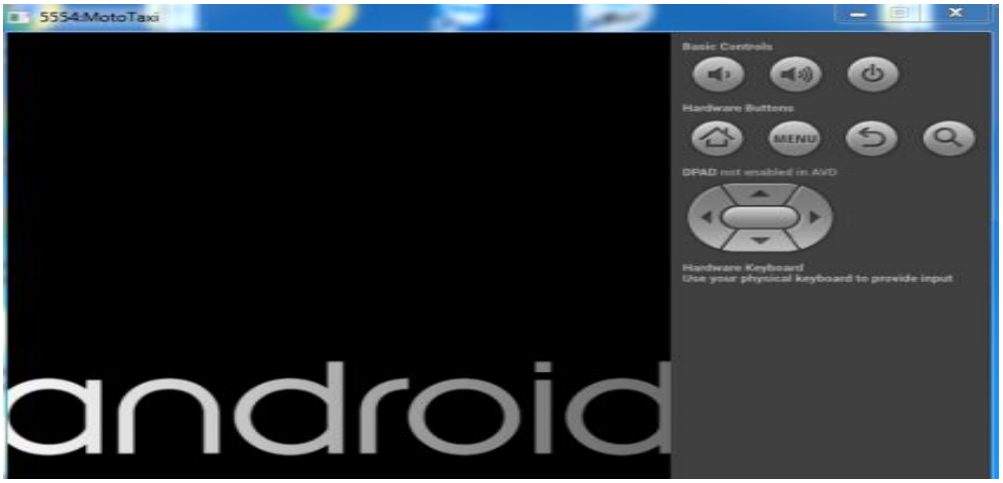

Figura 15 – Emulador AVD Inicializado

No momento da imagem 16 o *layout* do aplicativo onde os usuários vão realizar a buscar pelas informações do moto-taxista.

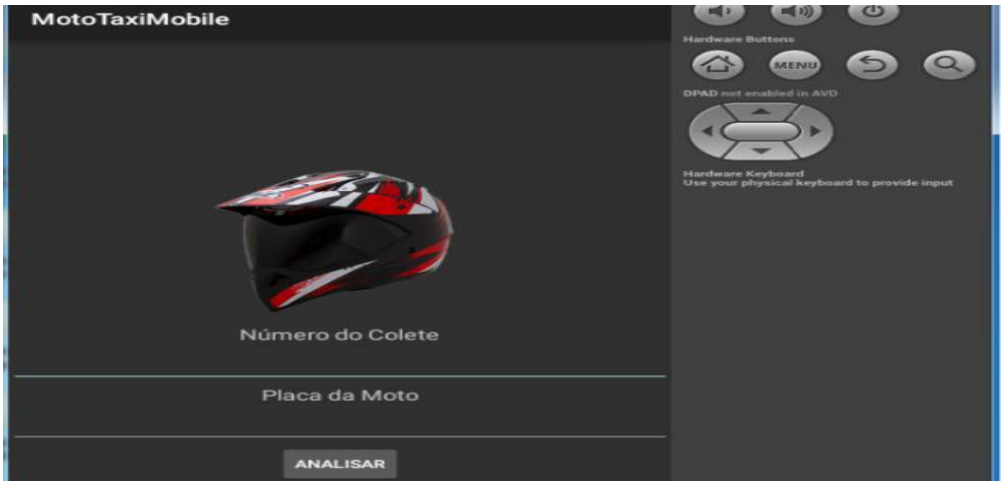

Figura 16 – Layout do Aplicativo

Na próxima imagem 17 e a base de dados do moto-taxistas, no *framework grails e emulador fazendo as buscar dos dados do moto-taxista.* .

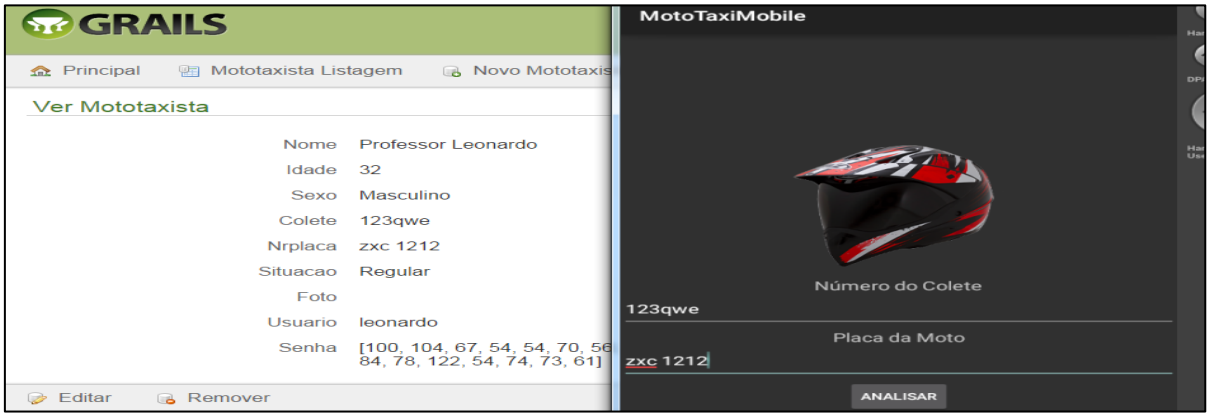

Figura 17 – Base de dado do moto-taxista e layout do Aplicativo

Na imagem sequente 18 e no *emulador AVD* onde o usuário realizou a buscar das informações do prestador de serviços, na próxima figura o resultado da buscar e todas as características do moto-taxista.

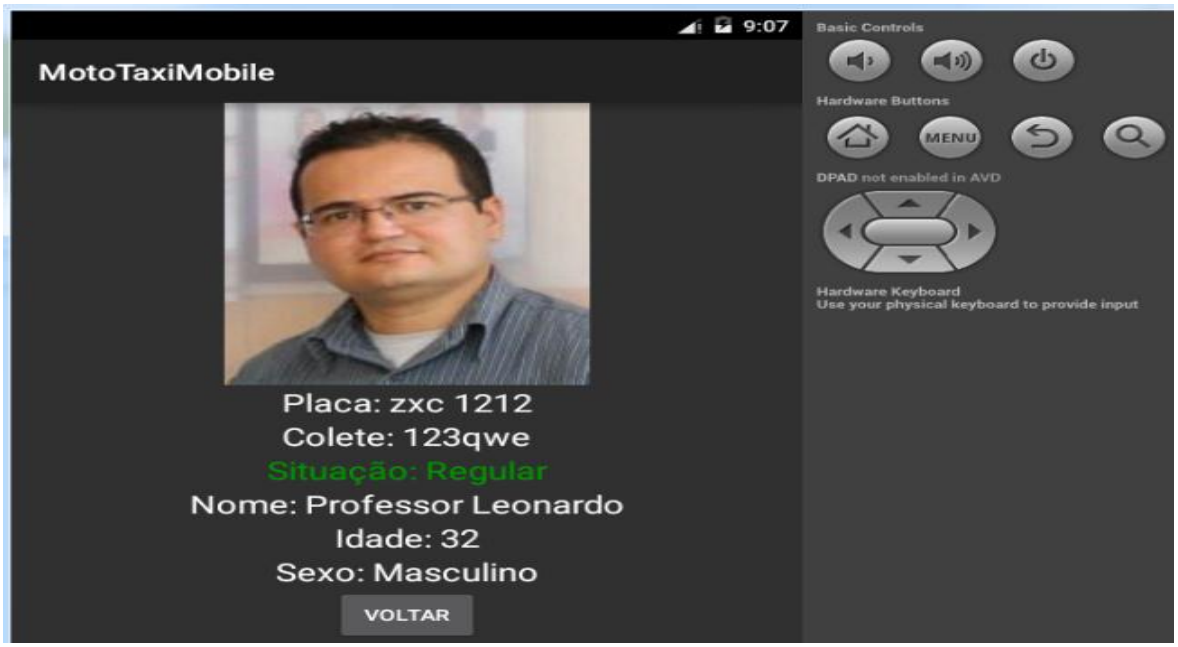

Figura 18 – Emulador AVD

5 CONCLUSÃO

O projeto foi desenvolvido devido a grande evolução das tecnologias para dispositivos móveis, aliado ao crescimento no mercado e as dificuldades das pessoas no seu dia a dia de simplesmente usufruir de serviços de moto-taxistas, principalmente, para as mulheres. Devido a estas necessidades, foi implementado o aplicativo de Moto-Taxista, nomeado AIMT (Aplicativo Identificação de Moto-Taxista).

Primeiramente, foi necessário definir quais tecnologias seriam usadas para o desenvolvimento do aplicativo. Para agilizar meu serviço e aumentar minha produtividade, utilizei a tecnologia de desenvolvimento ágil, com a ideia de usar o *framework Grails*. Posteriormente, para usar tecnologia móvel, o *Android Studio* e, para a comunicação se tornou necessária utilizar *Web Services.*, esta conecta-se com banco de dados *Postgres*ql disponibilizado no servidor, que faz as requisições *HTTP* nessa *Web Service*, a fim de consultar e enviar informações. Com essas ferramentas o projeto foi desenvolvido.

O sistema foi desenvolvido para o usuário, este tem a opção de escolher e evitar usar os serviços do moto-táxi. Ao digitar o número do colete do moto-taxista e o número da placa da motocicleta, serão mostradas informações do moto-táxi e, se o mesmo está ou não legalizado. Com o uso deste aplicativo podem-se abstrair informações necessárias para os usuários, com isso este aplicativo vai contribuir para o auxílio da população de Picos que utiliza o serviço de moto-táxi com o intuito de dar uma segurança e um conforto maior na utilização deste serviço.

Como trabalhos futuros, a possibilidade da implementação GPS no aplicativo para que o usuário pudesse escolher o moto-taxista mais próximo. Após a localização dos motos-taxistas, o sistema poderá fazer recomendações do condutor, informações sobre multas, irresponsabilidades no trânsito, como também acidentes envolvendo o condutor.

REFERÊNCIAS

BOLZANI, C. A. M. **Residências inteligentes:** um curso de domótica. 1. ed. São Paulo: LDF, 2004.

CRUZ, B. H. A. et. al. **Desenvolvimento de uma Aplicação Embarcada em Celular Visando Controle de Robô Via Wi-fi.** Santa Cruz do Sul, 2011.

DEITEL, A.; DEITEL, H. M.; MORGANO, M. **Android para programadores: uma abordagem baseada em aplicativos.** 1. ed. revisada e ampliada. Porto Alegre –RS: Bookman Editora, 2012.

DEVELOPERS. **Android Studio**. 2013. Disponível em:<http://developer.android.com/sdk/installing/studio.html.Acesso em 27 Jan. 2016

GARTNER, 2010. Disponível <http://www.gartner.com/newsroom/id/1372013.Acesso em 01 Mar. 2016

JUDD, C. M.; NUSAIRAT, J. F.; SHINGLER, J. **Beginning Groovyand Grails**. New York: Apress, 2008.

KUMAR, R. **Grails Web Framework**. Sapient Corporation, 2013.

KUROSE, J. F; KEITH, W. R. **Redes de computadores e a internet**: uma abondagem top-down. 3. ed. São paulo: Pearson Addison Wesley,2006.

LECHETA, R. R. **Android:** aprenda a criar aplicações para dispositivos móveis com o Android SDK. 3. ed. revisada e ampliada. São Paulo: Novatec Editora, 2013.

MANZANO, J. A. N.G. **Postgres SQL8.3.0 Interativo:** Quia de Orientação e Desenvolvimento para Windows. 1. ed. São Paulo: Editora Érica Ltda, 2008.

MÜLLER, N. **Metodologias Ágeis Extreme Programming e Scrum para o Desenvolvimento de Software.** 2008. Disponível em: <http://www.oficinadanet.com.br>. Acesso em: 27 jan. 2016.

RAFAEL J. W. A. M. **Desenvolvimento de Aplicativo para Smartphone com a Plataforma Android.** Monografia (Graduação em Engenharia de Computação) - Pontifícia Universidade Católica do Rio De Janeiro, Rio de Janeiro, 2009.

SILVA, L. E P.; BRACHT, E. C. Uma Abordagem para o Cálculo de Balanço Hídrico Climatológico. **Revista Brasileira de Computação Aplicada**, v. 2, n. 1, p. 2-16, 2010.

SMITH, P. L. G. **Grails in Action**. Greenwich: Manning, 2009.

SOUSA, C. L.; SOUZA, J. G. **Metodologia de Desenvolvimento de Software para a Fábricade Software do CEULP/ULBRA**. Encontro de Computação e Informática do Tocantins, In: Palmas, 2012. Disponível em: <http://ulbrato.br/encoinfo/artigos/2012/>. Acesso em: 27 fev. 2016.

VARGAS, G. P. **APOSTILA DE Telecomunicação II. 2002.** Disponível em: <http://www.ebah.pt/content/ABAAAfR_EAA/apostila-telecomunicacao-ii?part=13>. Acesso em: 30 out. 2015.

APÊNDICE A - Apresentação do Moto-Táxi

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="fill_parent"
              android:weightSum="1" android:gravity="center_vertical">
    <ImageView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/imageView" android:src="@drawable/capacete"
android:layout_marginBottom="10dp"/>
    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Número do Colete"
            android:id="@+id/textView"
android:layout_gravity="center_horizontal"
android:layout_marginBottom="5dp"/>
    <EditText
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:id="@+id/edt_colete"
android:layout_gravity="center_horizontal"
android:layout_marginBottom="5dp"/>
    <TextView
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textAppearance="?android:attr/textAppearanceMedium"
            android:text="Placa da Moto"
            android:id="@+id/textView2"
android:layout_gravity="center_horizontal"
android:layout_marginBottom="5dp"/>
```

APÊNDICE B – Código usuários Moto-Taxista e Administrador

```
   def login(){
        def usuario = params.usuario
        def senha = params.senha.encodeAsPassword().toString()
        if (usuario && senha){
            def mototaxistaInstance = Mototaxista.findByUsuarioAndSenha(usuario, senha)
            if (mototaxistaInstance) {
                session.mototaxista = mototaxistaInstance
                flash.message = "Login realizado com sucesso."
                redirect(action: "show", id: mototaxistaInstance.id)
            }else{
               flash.message = "Usuario e/ou senha nao conferem."
               redirect(uri: "/")
            }
         }else{
            flash.message = "Preencha os campos corretamente."
            redirect(uri: "/")
         }
    }
}
```

```
    def login() {
        def usuario = params.usuario
        def senha = params.senha.encodeAsPassword().toString()
        if (usuario && senha) {
            def administradorInstance = Administrador.findByUsuarioAndSenha(usuario, senha)
            if (administradorInstance) {
                session.administrador = administradorInstance
                flash.message = "Administrador logado com sucesso."
                redirect(controller: "mototaxista", action: "list")
         } else {
               flash.message = "Usuario e/ou senha não conferem."
               redirect(uri: "/")
            }
         } else {
            flash.message = "Preencha os campos corretamente."
            redirect(uri: "/")
         }
    }
}
```

APÊNDICE C - Código de Criptografia de Senha

```
import org.apache.commons.codec.binary.Base64

import java.security.MessageDigest

class PasswordCodec {

     static encode = { String s ->
          MessageDigest md = MessageDigest.getInstance("SHA")
          md.update s.getBytes("UTF-8")
          Base64.encodeBase64 md.digest()
     }

}
```

```
def save() {
    def mototaxistaInstance = new Mototaxista(params)

    def senha = params.senha
    mototaxistaInstance.senha = senha.encodeAsPassword().toString()

    if (!mototaxistaInstance.save(flush: true)) {
        render(view: "create", model: [mototaxistaInstance: mototaxistaInstance])
        return
    }

    flash.message = message(code: 'default.created.message', args:
  [message(code: 'mototaxista.label', default: 'mototaxista'),mototaxistaInstance.id])
    redirect(action: "show", id: mototaxistaInstance.id)
}
```

```
def save() {
    def administradorInstance = new Administrador(params)

    def senha = params.senha
    administradorInstance.senha = senha.encodeAsPassword().toString()


    if (!administradorInstance.save(flush: true)) {
        render(view: "create", model: [administradorInstance: administradorInstance])
        return
    }

     flash.message = message(code: 'default.created.message', args:
     [message(code: 'administrador.label', default: 'Administrador'), administradorInstance.id])
    redirect(action: "show", id: administradorInstance.id)
}
```

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

Identificação do Tipo de Documento

( ) Tese

( ) Dissertação

(X) Monografia

( ) Artigo

Eu, Daniel Dênnis Bezerra Carvalho,

autorizo com base na Lei Federal nº 9.610 de 19 de Fevereiro de 1998 e na Lei nº 10.973 de 02 de dezembro de 2004, a biblioteca da Universidade Federal do Piauí a divulgar, gratuitamente, sem ressarcimento de direitos autorais, o texto integral da publicação AIMT: Aplicativo de Identificação de moto-táxi da cidade de Picos

de minha autoria, em formato PDF, para fins de leitura e/ou impressão, pela internet a título de divulgação da produção científica gerada pela Universidade.

Picos-PI 11 de Março de 2016.

Daniel Dênnis Bezerra Carvalho

Assinatura