UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

PROTÓTIPO PARA INFORMATIZAÇÃO DE CONTROLE DE ESTACIONAMENTO EM ÁREA AZUL NO MUNICÍPIO DE BLUMENAU - SC

GUILHERME CRISTIANO GOLL

BLUMENAU
2013

2013/2-12

GUILHERME CRISTIANO GOLL

PROTÓTIPO PARA INFORMATIZAÇÃO DE CONTROLE DE ESTACIONAMENTO EM ÁREA AZUL NO MUNICÍPIO DE BLUMENAU - SC

Trabalho de Conclusão de Curso submetido à Universidade Regional de Blumenau para a obtenção dos créditos na disciplina Trabalho de Conclusão de Curso II do curso de Sistemas de Informação— Bacharelado.

Prof. Jacques Robert Heckmann, Mestre - Orientador

BLUMENAU
2013

2013/2-12

PROTÓTIPO PARA INFORMATIZAÇÃO DE CONTROLE DE ESTACIONAMENTO EM ÁREA AZUL NO MUNICÍPIO DE BLUMENAU - SC

Por

GUILHERME CRISTIANO GOLL

Trabalho aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II, pela banca examinadora formada por:

Presidente:	Prof. Jacques Robert Heckmann, Mestre – Orientador, FURB
Membro:	Prof. Miguel Alexandre Wisintainer, Mestre – FURB
Membro:	Prof. Wilson Pedro Carli, Mestre– FURB

Blumenau, 10 de dezembro de 2013.

Dedico este trabalho à minha família, que tanto me apoiou, e a todos os amigos que não perderam a fé em mim.

AGRADECIMENTOS

A Deus, pelo seu imenso amor e graça.

À minha família, que me deu suporte nas horas difíceis.

Aos meus amigos, pelos anos de amizade, companheirismo e compreensão.

[...] sabemos que não é bem assim. Se fosse fácil achar o caminho das pedras tantas pedras no caminho não seria ruim.

Humberto Gessinger

RESUMO

Este trabalho apresenta um protótipo de sistema de informação com o intuito de informatizar a operação do sistema de Área Azul no município de Blumenau. Para isto desenvolveu-se aplicativos específicos para cada tipo de usuário: condutores, monitores e operadores. O conceito de cliente/servidor é aplicado, através do qual atende-se às requisições de todos os envolvidos através de um único servidor. A tecnologia para dispositivos móveis com o sistema operacional Android é utilizada. Os aplicativos móveis permitem que os condutores efetuem o registro de estacionamento em vagas e permite que os monitores consultem estes registros. Como resultado obteve-se um protótipo que permite o uso flexível da Área Azul através de *tablets* e *smartphones*.

Palavras-chave: Dispositivos móveis. Sistema operacional Android. Área Azul. Estacionamento público.

ABSTRACT

This paper introduces an Information System prototype which aims to computerize the operation of the Blue Area in Blumenau city. For this purpose, specific apps were developed for each kind of user: drivers, monitors and operators. The client-server concept is applied by means of which all the requests are attended by only one server. The application uses mobile devices technology with Android Operating System. The mobile apps allows the drivers to register their parking act on public parking spots and enable monitors to check these parking registrations. As a result a prototype was obtained to ease the Blue Area use through tablets and smartphones.

Key-words: Mobile devices. Android operational system. Blue Area. Public parking lot.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE SIGLAS

API – *Application Programming Interface*

CORBA – *Common Object Request Broker Architecture*

DAO – *Data Access Object*

DCOM – *Distributed Component Object Model*

DETRAN – Departamento Estadual de Trânsito de Santa Catarina

DSC – Departamento de Sistemas e Computação

DWR – *Direct Web Remoting*

FURB – Fundação Universidade Regional de Blumenau

GPS - *Global Positioning System*

HTTP – *HyperText Transfer Protocol*

IDE – *Integrated Development Environment*

JSP – JavaServer *Pages*

NFC – *Near Field Communication*

OCR – *Optical Character Recognition*

PMB – Prefeitura Municipal de Blumenau

SBC – Sociedade Brasileira de Computação

SDK – *Software Development Kit*

SETERB – Serviço Autônomo Municipal de Trânsito e Transportes de Blumenau

SIG - Sistemas de Informações Geográficas

SOAP – *Simple Object Access Protocol*

WAR – *Web application ARchive*

XML - *eXtensible Markup Language*

SUMÁRIO

1 INTRODUÇÃO

A Área Azul trata-se de um sistema para controle de estacionamento rotativo regularizado. Foi implantada no município de Blumenau no dia 3 de setembro de 1984, através do decreto 2.303. Inicialmente eram 21 ruas abrangidas, totalizando 385 vagas. Os motivos que desencadearam-na, entre outros, foram: a localização geográfica da área central, dispondo apenas de três vias principais de escoamento do tráfego, a respectiva concentração comercial (foco gerador de tráfego), demanda por espaço superior à oferta de vagas e a elevada média de veículos por habitante (SERVIÇO AUTÔNOMO MUNICIPAL DE TRÂNSITO E TRANSPORTES DE BLUMENAU, 2012).

Desde então a Área Azul passou por alterações, mas sempre relacionadas aos métodos de fiscalização e não aos meios (infraestrutura e dispositivos). Com o aumento do número de veículos tornou-se necessária a implantação de mais vagas controladas, bem como o aumento no número de monitores. Segundo o Departamento Estadual de Trânsito de Santa Catarina (2013), em dezembro de 2002 haviam 113.548 veículos emplacados no município. Atualmente, são 230.687 veículos, ou seja, um aumento de cerca de 103% em quase 11 anos.

De acordo com o Serviço Autônomo Municipal de Trânsito e Transportes de Blumenau (2013), hoje, os usuários da Área Azul contam com 2.107 vagas, distribuídas em 58 ruas, sendo 376 vagas exclusivas para motos e 1.731 vagas para os demais veículos. O processo de fiscalização destas vagas continua o mesmo: monitores efetuam a verificação visual das vagas e, no caso de não encontrarem o respectivo cartão de estacionamento preenchido em local visível, efetuam a emissão da Notificação de Regularização (NR), em papel. Ou seja, a verificação é efetuada de forma visual, sujeita à atenção do monitor. A emissão e controle da Notificação de Regularização é feita em papel, sujeita ao extravio (tanto da via entregue ao usuário, quanto do formulário de controle do monitor).

A falta de um sistema móvel para controle da Área Azul torna o serviço dos monitores mais suscetível a falhas e abre espaço a melhorias e soluções. Segundo Tanji (2013), informações divulgadas pela empresa de consultoria IDC indicam um aumento de 79,5% na venda de *smartphones* com o sistema operacional Android no primeiro quadrimestre de 2013, se comparado ao mesmo período de 2012. Este aumento gera uma demanda por aplicativos móveis que atendam à necessidade e expectativa dos consumidores.

Segundo Sena (2012) a utilização de dispositivos móveis em atividades simples do cotidiano não é sinônimo de preguiça. Em um mundo cada vez menor e que exige maior

agilidade, seu uso pode economizar tempo e estresse, causados pela ausência de métodos mais simples para realização destas tarefas utilizando uma tecnologia adequada.

Com o crescente surgimento de novas tecnologias e a demanda de utilização de dispositivos móveis no cotidiano, o presente trabalho propõe o desenvolvimento de um protótipo de sistema informatizado para o controle de estacionamento da Área Azul no município de Blumenau, no estado de Santa Catarina. Para isto, utiliza-se de tecnologias móveis com a plataforma Android.

1.1 OBJETIVOS DO TRABALHO

O objetivo geral do trabalho é apresentar um protótipo de sistema informatizado para controle de estacionamento em locais públicos, no município de Blumenau, regidos pela Área Azul, para dispositivos móveis.

Os objetivos específicos do trabalho proposto são:

a) permitir aos usuários do sistema o registro, a aquisição de créditos, a consulta de locais de estacionamento e a consulta das notificações recebidas através do sistema;

b) permitir aos monitores emitir notificações e consultar dados dos veículos;

c) permitir aos administradores do sistema efetuar a manutenção dos locais de estacionamento.

1.2 ESTRUTURA DO TRABALHO

No primeiro capítulo tem-se a introdução ao tema principal deste trabalho com a apresentação da justificativa e dos objetivos.

No segundo capítulo apresenta-se a fundamentação teórica pesquisada sobre o sistema operacional Android, o Google *Maps*, os Sistemas de Informações Geográficas (SIG), o *Global Positioning System* (GPS), os *Web Services*, o *Simple Object Access Protocol* (SOAP), o *Direct Web Remoting* (DWR), o PayPal, o sistema atual de funcionamento, além de trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento do protótipo iniciando-se com o levantamento de informações, tendo na sequência a especificação técnica, detalhes sobre a implementação e, por fim, apresenta-se os resultados e discussão em torno do mesmo.

No quarto capítulo tem-se as conclusões deste trabalho bem como apresentam-se

sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os assuntos: sistema operacional Android, o Google *Maps*, Sistemas de Informações Geográficas (SIG), *Global Positioning System* (GPS), *Web Services*, *Simple Object Access Protocol* (SOAP), *Direct Web Remoting* (DWR), PayPal, o sistema atual de funcionamento, além de trabalhos correlatos.

2.1 ANDROID

O Android é uma plataforma para dispositivos móveis desenvolvida pela Google. Silva (2011) diz que "pode-se dizer que a plataforma Android é a primeira plataforma móvel completa, aberta e livre".

Segundo Android Developers (2013a), o Android é a plataforma móvel mais popular do mundo. São centenas de milhões de aparelhos em mais de 190 países. A cada dia mais de um milhão de novos dispositivos executando esta plataforma são ativados pela primeira vez. Para desenvolver para esta plataforma é necessário efetuar o *download* do Android SDK, uma plataforma que contém as bibliotecas e ferramentas de desenvolvimento necessárias para desenvolver, testar e depurar aplicativos para Android. É possível efetuar o *download* da suíte completa para desenvolvimento, chamada ADT Bundle. Nela encontra-se o Eclipse IDE + ADT *plugin*, o Android SDK Tools, o Android Platform-tools, a plataforma Android mais recente (atualmente é a KitKat 4.4) e imagem do sistema operacional Android mais recente para utilização no emulador (ANDROID DEVELOPERS, 2013b).

É possível, ainda, efetuar o *download* apenas do pacote Android SDK Tools. Esta opção é utilizada por desenvolvedores que já possuem uma IDE. Neste caso, o usuário deverá efetuar a instalação e configuração deste pacote separadamente. Se a IDE utilizada for o Eclipse, é possível, também, adicionar o ADT *plugin*, que se trata de um *plugin* para a IDE que permite rápido acesso às configurações do Android SDK Tools (ANDROID DEVELOPERS, 2013b).

2.2 GOOGLE MAPS

O Google *Maps* é um serviço gratuito de mapas disponibilizado pela Google. Através dele é possível visualizar mapas, rotas, endereços, localização e contato de empresas entre outros dados (GOOGLE MAPS, 2013).

Através de um navegador de internet compatível com este serviço, é possível ter acesso ao conteúdo disponibilizado. Aos que possuem um dispositivo móvel com Android, versão 2.1 (ou superior) é possível, ainda, ter acesso a recursos de navegação, visualização de tráfego em tempo real e mapas de locais cobertos como aeroportos e shoppings (GOOGLE MAPS, 2013).

Para os desenvolvedores de aplicativos móveis, a Google disponibiliza uma suíte de *Application Programming Interface* (API) do Google *Maps*. Aos desenvolvedores *web* é possível criar páginas em *HyperText Markup Language* versão 5 (HTML5) utilizando a API JavaScript do Google *Maps* versão 3.

Com a API do Google *Places* é possível encontrar lugares próximos em uma grande variedade de categorias. A API de preenchimento automático do Google *Places* auxilia os usuários a encontrar o que procuram com mais rapidez, sugerindo lugares próximos à medida que o texto é digitado.

Com a API do Google *Static Maps* é possível criar imagens do Google *Maps* no aplicativo, incorporando uma URL com uma *tag* de imagem (GOOGLE DEVELOPERS, 2013).

2.3 SISTEMA DE INFORMAÇÃO GEOGRÁFICA

Segundo Pena (2013), os Sistemas de Informação Geográfica (SIG) são equipamentos e meios tecnológicos para se estudar o espaço terrestre. Resultam da combinação de três tipos distintos de tecnologias:

a) o sensoriamento remoto: utilização de ferramentas, como satélites e radares, para captar informações e imagens acerca da superfície terrestre;
b) GPS: que pode emitir informações de qualquer lugar do mundo, a partir de coordenadas geográficas;
c) o geoprocessamento: etapa de tratamento das informações oriundas do sensoriamento remoto e do GPS para a produção de mapas, cartogramas, gráficos e sistematizações em geral.

2.4 GPS

O *Global Positioning System* (GPS) foi desenvolvido pelo Departamento de Defesa dos Estados Unidos. A Marinha, a Força Aérea e o Exército apresentaram suas ideias e conceitos, e em 1973 o governo Norte Americano aprovou o projeto que deu origem ao NAVSTAR. O primeiro satélite foi lançado em 1974. Entre 1978 e 1985 foram lançados outros 11 satélites para testes (TOMTOM, 2013a).

> O GPS (*Global Positioning System* ou Sistema de Posicionamento Global) é um sistema espacial baseado em rádio navegação [...]. Tem como princípio a medida da distância entre a antena do satélite e a do receptor [...]. O GPS é tido como uma ferramenta de posicionamento e transferência de tempo, utilizada no auxílio de atividades de Topografia, Fotogrametria, Sensoriamento Remoto e Geodésia. Tais atividades têm como finalidade comum o mapeamento e a obtenção de dados geográficos (latitude, longitude e altitude) referentes a uma área específica. (DRAGO; DISPERATI, 1996, p. 1-2).

Seu propósito, inicialmente, era exclusivamente militar. Entretanto, no dia 1 de setembro de 1983 ocorreu uma tragédia aérea: um voo em direção à Seul, na Coréia do Sul, desviou-se da rota e invadiu o espaço aéreo da URSS. O mesmo foi abatido por um caça soviético e os 269 passageiros e tripulação morreram. Duas semanas após o incidente, o então Presidente dos EUA, Ronald Reagan, propôs que o GPS fosse disponibilizado para utilização civil (TOMTOM, 2013b).

Segundo Decicino (2009), há várias aplicações para o GPS, a mais comum delas é em automóveis. É oferecida com mapas de cidades, com funcionalidades como traçar rotas entre pontos e a visualização geral da área.

Atualmente, existem dois sistemas de posicionamento por satélite: o GPS, desenvolvido e mantido pelos Estados Unidos, e o Glonass, desenvolvido na Rússia. Outros dois sistemas estão atualmente em fase de desenvolvimento: o Compass, pela China, e o Gelileo, pela Europa (FRANCISCO, 2013).

2.5 Coordenadas geográficas

De acordo com Freitas (2008) coordenadas geográficas são linhas imaginárias pelas quais a Terra é dividida. Essas linhas são os paralelos e meridianos. Através deles pode-se obter localizações precisas em qualquer ponto do planeta.

Alguns itens importantes nas coordenadas geográficas, segundo Freitas (2008), são:

a) o plano equatorial: plano imaginário que divide a Terra em polo Sul e polo Norte, de tamanhos iguais;

b) os paralelos: linhas imaginárias paralelas ao plano equatorial;
c) os meridianos: linhas imaginárias paralelas ao meridiano de Greenwich que ligam os pontos norte e sul;
d) latitude: é a distância medida em graus de um determinado ponto do planeta entre o arco do meridiano e a linha do equador;
e) longitude: é a localização de um ponto da superfície medida em graus, nos paralelos e no meridiano de Greenwich.

2.6 WEB SERVICES

Segundo a *World Wide Web Consortium* (W3C), *Web Services* proveem meios padrões de interoperabilidade entre diferentes aplicações de software e em execução em variadas plataformas ou *frameworks*. São caracterizados por sua grande interoperabilidade e extensibilidade, bem como suas descrições de métodos, graças ao uso de XML. Podem ser utilizados de forma livre ou acoplada, a fim de alcançar operações complexas. Programas com serviços simples podem interagir entre si ao ponto de entregar um serviço sofisticado e com valor agregado. A W3C *Web Services Activity* é responsável por desenhar a infraestrutura, definir a arquitetura e criar as tecnologias centrais para *Web Services* (WORLD WIDE WEB CONSORTIUM, 2013).

Esta tecnologia é utilizada pelos Correios, por exemplo, que dispõem de um *Web Service* para cálculo de prazos e preços de encomendas em lojas virtuais e sites. Este *Web Service* é destinado aos clientes SEDEX, e-SEDEX e PAC que necessitam calcular o preço e prazo de entrega de encomendas em suas páginas, de forma customizada (EMPRESA BRASILEIRA DE CORREIOS E TELÉGRAFOS, 2013).

Pamplona (2010) diz que "*Web Services* é a tecnologia ideal para comunicação entre sistemas, sendo muito usado em aplicações B2B". Para comunicar-se com o *Web Service* é necessário implementar o *Simple Object Access Protocol* (SOAP).

2.6.1 Simple Object Access Protocol

No desenvolvimento de aplicações, é importante permitir comunicação via Internet entre os programas. Atualmente as aplicações utilizam o *Remote Procedure Call* (RPC) entre objetos como DCOM e CORBA. Porém o HTTP não foi desenhado para isto. Além disso, o

RPC possui um problema de compatibilidade e segurança: *firewalls* e servidores de *proxy* geralmente irão bloquear este tipo de tráfego. A melhor forma de prover comunicação entre aplicações é sob HTTP, pois o mesmo é suportado por todos os navegadores de internet e servidores. O *Simple Object Access Protocol* (SOAP) é o protocolo utilizado na comunicação através de *Web Services*, mais especificamente no envio e retorno dos dados. É baseado em *eXtensible Markup Language* (XML) e livre de plataforma e/ ou linguagem de programação (W3SCHOOLS, 2013).

2.7 DIRECT WEB REMOTING

O *Direct Web Remoting* (DWR) é uma biblioteca RPC que torna fácil a chamada de funções Java a partir de JavaScript e vice-versa. Possui uma extensa base de usuários e tem sido utilizado em vários projetos (DWR, 2013). Com ele é possível eliminar todos os passos do ciclo de requisição-resposta do AJAX. Isto significa que o lado cliente da aplicação não precisará lidar com um objeto `XMLHttpRequest` diretamente, ou com as respostas do servidor (IBM, 2005).

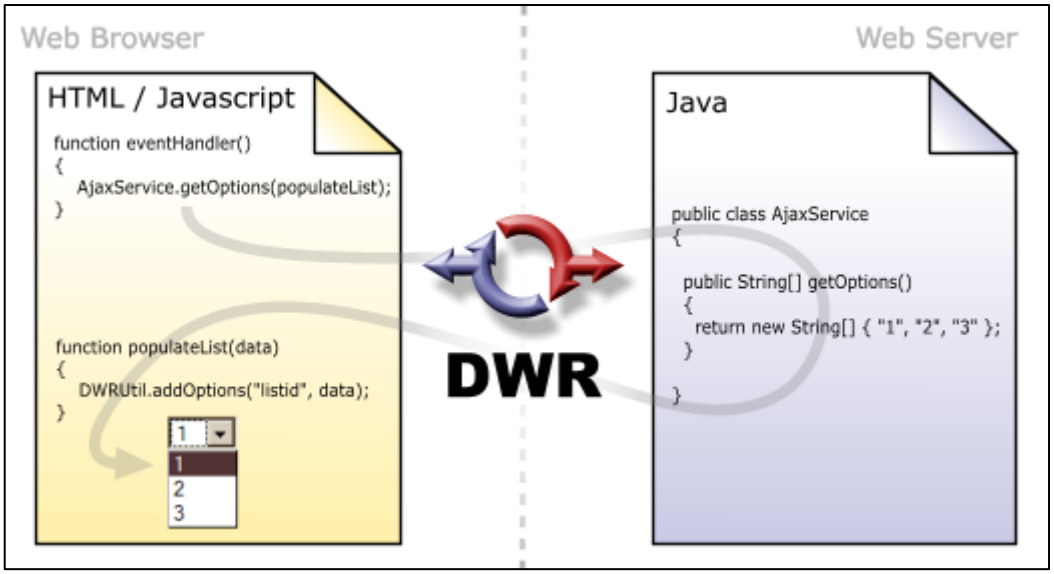
O DWR é implantado dentro do projeto *Web* da aplicação em desenvolvimento, como um *servlet*. Desempenha dois papéis principais:

a) para cada classe mapeada, o DWR gera automaticamente JavaScript para incluí-lo na página *Web*. O JavaScript gerado contém funções que representam os métodos da classe Java e também executa as requisições do tipo `XMLHttpRequest`. Estas requisições são enviadas para o *servlet* DWR;
b) em seu segundo papel, o DWR traduz a requisição em um método de chamada para um objeto Java no lado servidor e envia o valor de retorno do método chamado de volta para o lado cliente (página *Web*), codificada em JavaScript. Além disso, possui funções JavaScript utilitárias que ajudam a desenvolver tarefas simples de interface de usuário.

A Figura 1 mostra como o DWR pode alterar o conteúdo de uma lista de seleção como resultado de um evento JavaScript. Quando aberto no navegador de internet (*Web Browser*) os artefatos do DWR mapeados via JavaScript podem ser chamados através de funções (*functions*). Eventos como o clique do cursor sobre um botão, por exemplo, podem chamar estas funções. O código implementado nas mesmas efetua a chamada do método junto ao DWR, que converte os parâmetros (quando houver) e faz a chamada junto aos métodos Java.

O retorno do processamento atua de forma semelhante, ou seja, através de uma função de retorno na qual o resultado deve ser tratado.

Figura 1 – Funcionamento do DWR



Fonte: DWR (2013).

2.8 PAYPAL

O PayPal é uma plataforma de pagamento *online* de compras feitas na internet, no celular ou *tablet*. Fundado em 1998, na Califórnia, Estados Unidos da América, como uma solução pioneira para o envio e recebimento de pagamentos *online*. Quatro anos depois foi adquirido pelo grupo eBay Inc., maior plataforma global de comércio eletrônico (PAYPAL, 2013a).

Barros (2013) diz que "o PayPal é o serviço mais popular de pagamentos digitais do mundo. Ele está disponível para Android, iOS, Windows Phone e permite que os usuários gerenciem as suas contas onde estiverem". Os pagamentos podem ser feitos, também, através de QR Codes, transferências ou recebimentos.

De acordo com PayPal (2013a) atualmente são mais de 123 milhões de clientes em 190 países. Os dados utilizados por estes clientes são armazenados sigilosamente e não são compartilhados, nem mesmo, com o vendedor do produto.

Para utilizar o PayPal em um aplicativo Android é necessário utilizar o PayPal Android SDK, que provê bibliotecas nativas para o processamento simples de pagamentos, tornando fácil aceitar pagamentos através do PayPal ou cartões de crédito no aplicativo. O *download* deste SDK é gratuito. Fica sob responsabilidade do SDK a interface de pagamento (inserção e

validação de dados de cartão de crédito), a validação do pagamento junto ao PayPal e a disponibilização de uma constatação do pagamento efetuado. Para o desenvolvedor, fica a responsabilidade de receber a constatação do pagamento enviada pelo SDK, enviá-la para o servidor da aplicação e liberar o produto ou serviço ao seu cliente (PAYPAL, 2013b).

2.9 SISTEMA ATUAL

O Serviço Autônomo Municipal de Transportes e Trânsito de Blumenau – SETERB foi criado oficialmente em 27 de março de 1979, pela Lei Municipal nº 2.347, e em 22 de Dezembro de 2003 passou a receber a denominação utilizada até os dias atuais. É considerada uma entidade autárquica subordinada ao Governo Municipal com autonomia econômico-financeira dentro dos limites da lei. Em 1991 passou a gerenciar os serviços de trânsito da cidade, assumindo a administração da Guarda de Trânsito, considerada a mais antiga do País, já que foi criada em 1955, e o Estacionamento Regulamentado, denominado Área Azul, criado em 1984 por Decreto Municipal e hoje regulamentada pela Lei nº 4181, de 11 de Fevereiro de 1993 (SERVIÇO AUTÔNOMO MUNICIPAL DE TRÂNSITO E TRANSPORTES DE BLUMENAU, 2012).

É composta atualmente por três diretorias: Diretoria Administrativa Financeira, Diretoria de Trânsito e Diretoria de Transportes, à qual este trabalho está relacionado.

Através de análise informal, realizada com base nos procedimentos executados por um condutor de veículo automotor e das atividades de um monitor de Área Azul, chegou-se ao seguinte cenário que representa o fluxo das atividades que compõem a Área Azul:

a) o condutor do estacionamento adquire o talão de Estacionamento da Área Azul em um local credenciado;
b) o condutor estaciona o veículo em um local identificado pela placa “Estacionamento Regulamentado”;
c) o condutor preenche uma via do talão de estacionamento com data e hora do seu efetivo estacionamento;
d) durante o período de permanência do veículo na vaga, a qualquer momento o monitor poderá realizar a verificação da regularidade e validade do talão;
e) o condutor remove seu veículo da vaga, até o horário máximo para permanência na vaga.

No caso do motorista não possuir o talão de estacionamento necessário e ainda sim utilizar uma vaga devidamente sinalizada, durante a atividade exposta no item “d”, o mesmo é

sujeito a receber a devida notificação, também conhecida como “Amarelinha”, emitida pelo monitor responsável pela área. Neste caso, esta notificação deverá ser paga no prazo máximo em até 15 dias, conforme Artigo 10 da Lei Municipal Ordinária nº 7721/2011.

2.10 TRABALHOS CORRELATOS

Destacam-se atualmente alguns sistemas de controle semelhantes a este trabalho, os quais são relatados abaixo.

A empresa Declink, estabelecida na cidade do Rio de Janeiro e fundada em 1990, possui um sistema chamado Vaga AZUL – Controle de Estacionamento em Logradouro Público. O aplicativo propõe-se a controlar o uso dos espaços, a demanda e, através de relatórios estatísticos do sistema, elaborar o planejamento. O funcionamento é semelhante ao desenvolvido neste trabalho, porém não há módulo para controle nem consulta do registro de estacionamento por parte do monitor, ou seja, da guarda responsável por verificar a validade do *ticket* de estacionamento (DECLINK, 2013).

No município de Belo Horizonte, no estado de Minas Gerais, a Empresa de Transportes e Trânsito de Belo Horizonte – BHTRANS – é a responsável pelas atividades relativas ao transporte e trânsito em âmbito municipal. O sistema da Prefeitura não é informatizado. O controle feito pela prefeitura é muito semelhante à Área Azul, utilizada no município de Blumenau. A empresa é responsável pelo gerenciamento do Estacionamento Rotativo, desde a confecção e comercialização dos talões, pesquisas de campo, implantação e manutenção da sinalização, monitoramento do desempenho operacional, até a fiscalização das áreas regulamentadas (BHTRANS, 2013).

No segundo semestre de 2012, o então graduando do curso de Bacharelado em Ciência da Computação da Fundação Universidade Regional de Blumenau (FURB), Felipe Demarchi, desenvolveu em seu Trabalho de Conclusão de Curso um sistema para dispositivos móveis com a plataforma Android, a mesma que será utilizada neste trabalho. O sistema foi desenvolvido com o intuito de migrar o sistema Aruana-Maleta, utilizado para a coleta de dados de manejo em campo para processamento em um módulo *desktop*, para a plataforma *mobile*. O mesmo aliou os recursos móveis disponíveis em *tablets*, bem como a praticidade de uso desses, para substituir a utilização da maleta grande e pesada utilizada pelo Aruana-Maleta (DEMARCHI, 2012).

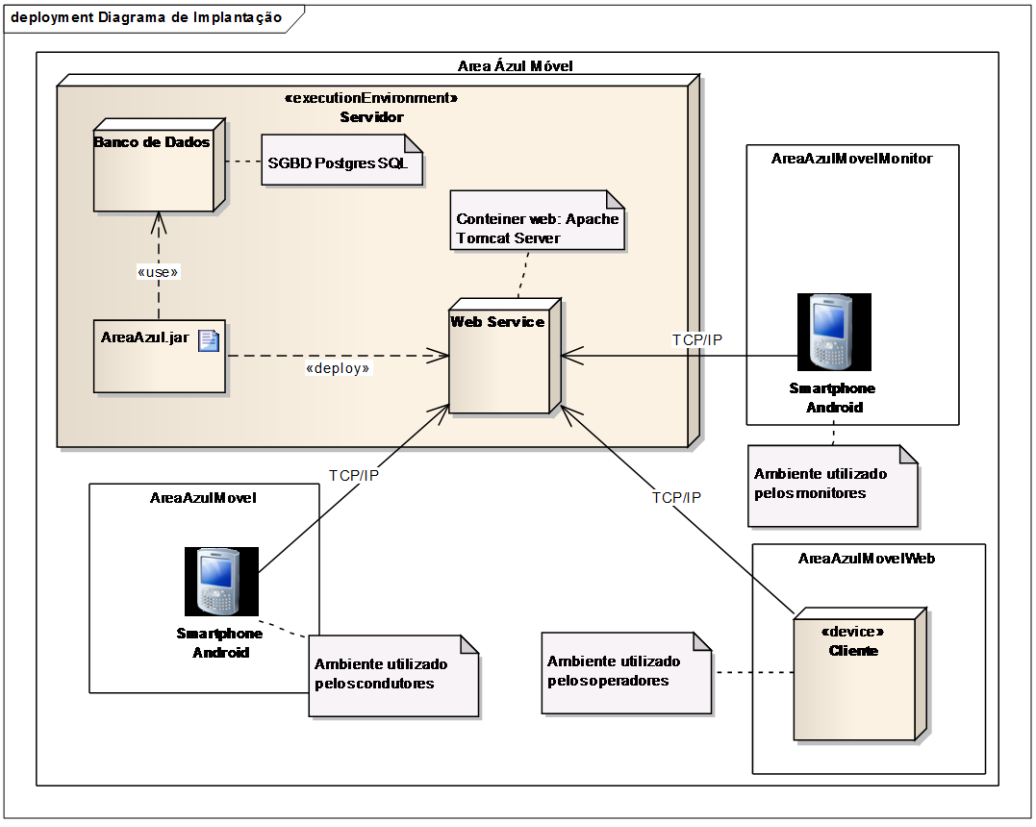
3 DESENVOLVIMENTO DO PROTÓTIPO

Apresenta-se neste capítulo informações técnicas referentes ao desenvolvimento do protótipo, com a descrição do levantamento de informações, os requisitos funcionais e não funcionais, os diagramas de casos de uso, os diagramas de atividades, o modelo de entidade e relacionamento, os diagramas de classes, as técnicas e ferramentas utilizadas, a operacionalidade da implementação e os resultados e discussões.

3.1 LEVANTAMENTO DE INFORMAÇÕES

O Área Azul Móvel é um protótipo de sistema de informação que visa trabalhar paralelamente com o sistema atual, manual, que controla as áreas de Estacionamento Regulamentado – Área Azul – no município de Blumenau. Para tal, foi desenvolvida uma suíte de protótipos de aplicativos para dispositivos móveis, *tablets* ou *smartphones* com a plataforma Android, e um protótipo de sistema *Web*. Na Figura 2 é apresentado o seu diagrama de implantação para um melhor entendimento.

Figura 2 – Diagrama de implantação



O AreaAzulMovel é o protótipo de aplicativo disponibilizado para os condutores de veículos para que efetuem o registro de estacionamento em locais indicados, adquiram créditos de estacionamento, cadastrem-se no sistema e consultem uma relação de ruas com estacionamento regulamentado.

Para o registro de estacionamento é utilizada uma API do Google *Maps*. Através dela é exibido um mapa, atualizado dinamicamente com base na melhor posição identificada do usuário (GPS ou outros serviços de localização da plataforma Android).

No mapa são exibidas a posição atual do usuário, bem como duas circunferências com diâmetro de 20 metros e 100 metros, respectivamente. A primeira circunferência é preenchida na cor vermelha e exibe as vagas de estacionamento próximas ao usuário, através das quais o mesmo poderá efetuar seu registro de estacionamento. A segunda circunferência não possui cor de preenchimento, apenas borda indicadora do limite e exibe as vagas próximas ao usuário, mas que não podem ser utilizadas pelo mesmo para registro de estacionamento. É essencial que o usuário encontre-se, de fato, próximo ao ponto de estacionamento no qual seu veículo fora estacionado para efetuar o registro de estacionamento.

A aquisição de créditos feita através do aplicativo móvel consome a API do PayPal, própria para operações que utilizam cartão de crédito.

O AreaAzulMovelMonitor é o protótipo de aplicativo disponibilizado para os monitores da Área Azul. Com ele, o usuário poderá efetuar a verificação de registros de estacionamentos – aqueles efetuados através do AreaAzulMovel –, emitir notificações quando não houver um registro de estacionamento ativo e consultar informações detalhadas de um veículo a partir de sua placa. O uso desta aplicação é restrito aos monitores da Área Azul e, portanto, seu *download* será restrito. Ainda como medida de segurança, não foi disponibilizada interface de cadastro de novos usuários através deste aplicativo, pois, em caso de violação e *download* indevido do aplicativo, pessoas com má fé não terão acesso a informações e procedimentos sigilosos.

O AreaAzulMovelWeb é o protótipo de sistema *Web* para uso dos operadores de Área Azul. Através deste sistema, pode-se efetuar o cadastro de novos usuários, efetuar a manutenção dos pontos de estacionamento em Área Azul (verificação, inclusão, ativação e inativação de vagas) e ainda consultar e dar baixa nas notificações emitidas eletronicamente. Para seu desenvolvimento foi utilizada a tecnologia Java*Server Pages* (JSP), em conjunto com *framework* para *front-end* Twitter Bootstrap. As requisições feitas através deste módulo passam pela interface do DWR, que traduz as requisições feitas em JavaScript (cliente) para classes e métodos construídos em Java (servidor).

Para atender às requisições efetuadas pelos três protótipos, há uma aplicação do tipo servidor, construída na linguagem Java com banco de dados Postgres SQL. Utilizando da tecnologia de *Web Services*, esta aplicação recebe as solicitações de processamento e disponibiliza um retorno das mesmas à interface que o invocou. Os métodos disponíveis permitem: cadastro de novos usuários, *login* nos aplicativos, aquisição de créditos, *download* da lista de vagas de estacionamento, vinculação de veículos ao perfil de condutores, consulta de dados de veículos, registro e consulta de estacionamentos e emissão e consulta de notificações.

3.2 ESPECIFICAÇÃO

Nesta seção é apresentada a especificação do protótipo Área Azul Móvel, modelado utilizando-se a ferramenta Enterprise Architect. Utiliza-se, ainda, a notação *Unified Modeling Language* (UML) para a criação dos diagramas de casos de uso, classes e atividades.

3.2.1 Requisitos do sistema

O Quadro 1 apresenta os requisitos funcionais previstos para o protótipo e sua rastreabilidade, ou seja, sua vinculação com o(s) caso(s) de uso associado(s).

Quadro 1 – Requisitos funcionais

Requisitos Funcionais	**Caso de Uso**
RF01 – O sistema deverá permitir que o condutor registre-se no aplicativo.	UC01, UC03
RF02 – O sistema deverá permitir aos usuários (condutor, monitor e operador) efetuar o *login*.	UC02, UC04
RF03 – O sistema deverá permitir ao condutor associar veículo ao seu perfil.	UC06
RF04 – O sistema deverá permitir ao condutor adquirir créditos.	UC05
RF05 – O sistema deverá permitir aos usuários (condutor, monitor e operador) a consulta de locais para estacionamento.	UC07
RF06 – O sistema deverá permitir ao condutor realizar o registro (*check-in*) em vagas de estacionamento.	UC06
RF07 – O sistema deverá validar a posição geográfica do condutor ao tentar efetuar o registro em uma vaga.	UC06
RF08 – O sistema deverá validar os créditos do condutor ao tentar efetuar o registro em uma vaga.	UC06
RF09 – O sistema deverá permitir a consulta apenas de notificações de estacionamento emitidas para o veículo em Área Azul.	UC11
RF10 – O sistema deverá permitir ao monitor a emissão de	UC12

notificações (amarelinhas).	
RF11 – O sistema deverá restringir a emissão de notificações por parte dos monitores e guardas de trânsito.	UC12
RF12 – O sistema deverá armazenar um histórico de estacionamento nas vagas, contendo o usuário, veículo e horário de entrada (registro) na mesma.	UC06,UC12,UC13
RF13 - O sistema deverá manter uma lista atualizada dos pontos de estacionamento no município.	UC08,UC09,UC10
RF14 - O sistema deverá permitir o cadastro, ativação e inativação de pontos de estacionamento apenas para o administrador do sistema.	UC08,UC09,UC10
RF15 - O sistema deverá emitir notificações nos últimos 10 e 5 minutos restantes de estacionamento, e após o estacionamento no local expirar.	UC06
RF16 - O sistema deverá restringir o acesso às informações completas de um veículo para os monitores e guardas de trânsito.	UC13

O Quadro 2 lista os requisitos não funcionais previstos para o protótipo.

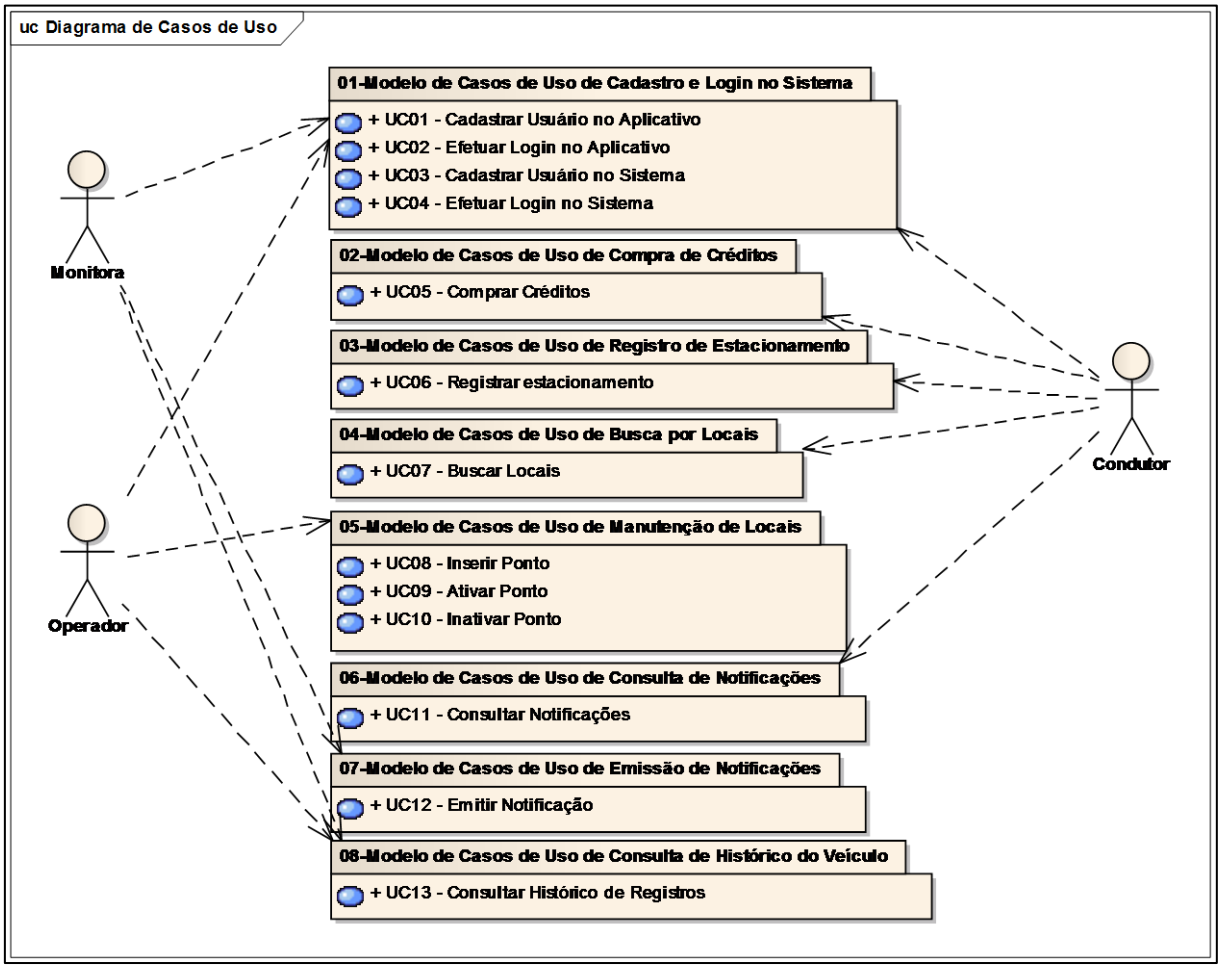
Quadro 2 – Requisitos não funcionais

Requisitos Não Funcionais
RNF01 – O sistema servidor deverá utilizar o SGBD Postgres SQL 9.0.
RNF05 – O sistema servidor deverá utilizar a plataforma *web*.
RNF02 – O sistema cliente deverá ser compatível com o sistema operacional Android versão 4.2.2 para *Tablets* e *Smartphones*.
RNF03 – O sistema deverá armazenar as senhas dos usuários criptografadas na base de dados.
RNF04 – O sistema deverá utilizar tecnologia *Global Positioning System* (GPS) para rastrear a posição dos usuários ao efetuar o registro de estacionamento.
RNF06 – O sistema deverá prover segurança nas transações envolvendo dados de cartão de crédito.
RNF07 – Os veículos deverão ser identificados através das placas.
RNF08 – As transações de compra de créditos e registro de estacionamento não devem levar mais de 60 segundos para serem completadas. Caso levem deverão ser canceladas.

3.2.2 Diagrama de casos de uso

Esta subseção apresenta, através da Figura 3, o diagrama de casos de uso necessário para o entendimento do protótipo desenvolvido, com as atividades exercidas por cada um dos três atores: operador, monitor e condutor. O diagrama de casos de uso é apresentado em nível de pacotes e utilizou-se a ferramenta Enterprise Architect para a criação. A descrição dos principais casos de uso está apresentada no Apêndice A.

Figura 3 – Diagrama de casos de uso



O condutor é o ator responsável por utilizar de forma regular as vagas de estacionamento controladas pela Área Azul. Para isto, ele poderá fazer o *download* da aplicação a partir de endereço *web* a ser disponibilizado. Com o aplicativo instalado em seu dispositivo, ele deverá acessar a interface inicial, na qual poderá optar por efetuar um novo cadastro ou efetuar o *login*. Com o *login* realizado, o usuário poderá efetuar a aquisição de créditos (com o uso de cartão de crédito), consultar todos os locais de estacionamentos disponíveis, vincular veículos ao seu perfil, efetuar o registro de estacionamento e consultar as notificações emitidas para os veículos associados ao seu perfil.

O monitor é o ator responsável por verificar a regularidade dos veículos estacionados em locais controlados pela Área Azul. Utiliza-se do protótipo devido – AreaAzulMovelMonitor – para efetuar a verificação de registros de estacionamentos. O monitor não poderá emitir notificação de estacionamento sem que antes tenha feito uma verificação da placa do veículo e local. Para efetuar a verificação, o monitor insere a placa do veículo e o número de identificação do ponto de estacionamento.

O operador é o ator responsável por manter atualizada a lista de locais de

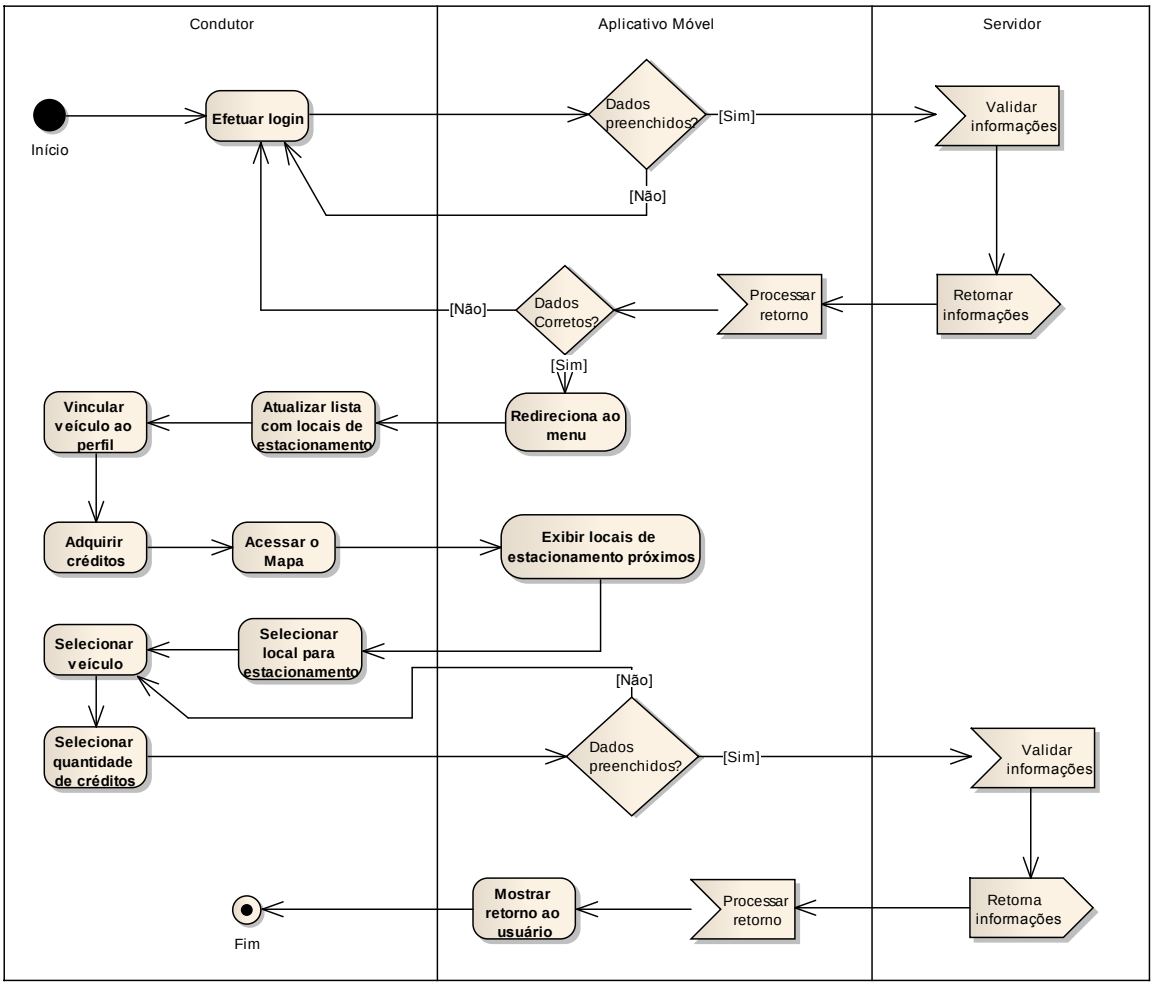
estacionamento e efetuar a baixa de notificações pendentes. Faz uso do sistema *Web* – AreaAzulMovelWeb – para estas operações e também para o cadastro de novos usuários no sistema. Apenas o operador poderá efetuar o cadastro de novos operadores e monitores.

A seguir apresenta-se os diagramas de atividades necessários para entendimento do fluxo das operações básicas, entre elas: o registro de estacionamento efetuado pelos condutores e a consulta de registro de estacionamento e emissão de notificação efetuados pelos monitores.

3.2.3 Diagrama de atividades de registro de estacionamento

Na Figura 4 é apresentado o diagrama de atividades de registro de estacionamento. O papel humano na operação é desempenhado pelo ator condutor que, em posse do aplicativo instalado em seu dispositivo móvel, efetuará a atividade abaixo. O diagrama representa as atividades desde o *login*, efetuado pelo condutor, até o término do registro, através da mensagem de confirmação “Procedimento efetuado com sucesso”.

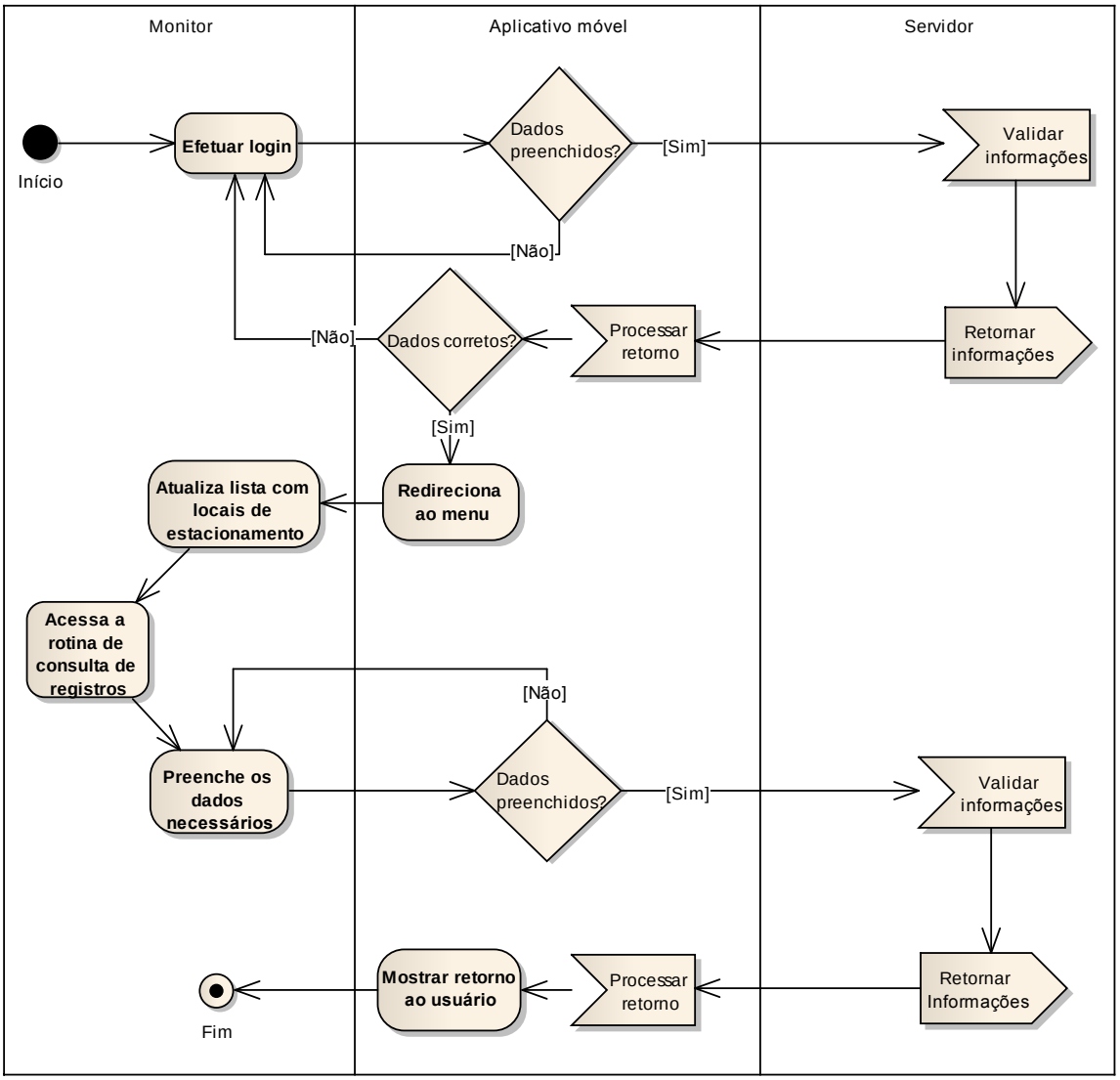
Figura 4 – Diagrama de atividades de registro de estacionamento



3.2.4 Diagrama de atividades de consulta de registro de estacionamento

Na Figura 5 é apresentado o diagrama de atividades de consulta de registro de estacionamento. O papel humano nesta etapa da operação é desempenhado pelo ator monitor que, em posse do aplicativo apropriado (AreaAzulMovelMonitor), efetuará as atividades abaixo. Para a consulta dos registros de estacionamento o monitor precisará informar a placa do veículo e o número de identificação do ponto de estacionamento. Esta ação é executada logo após o usuário acessar a rotina de consulta de registros.

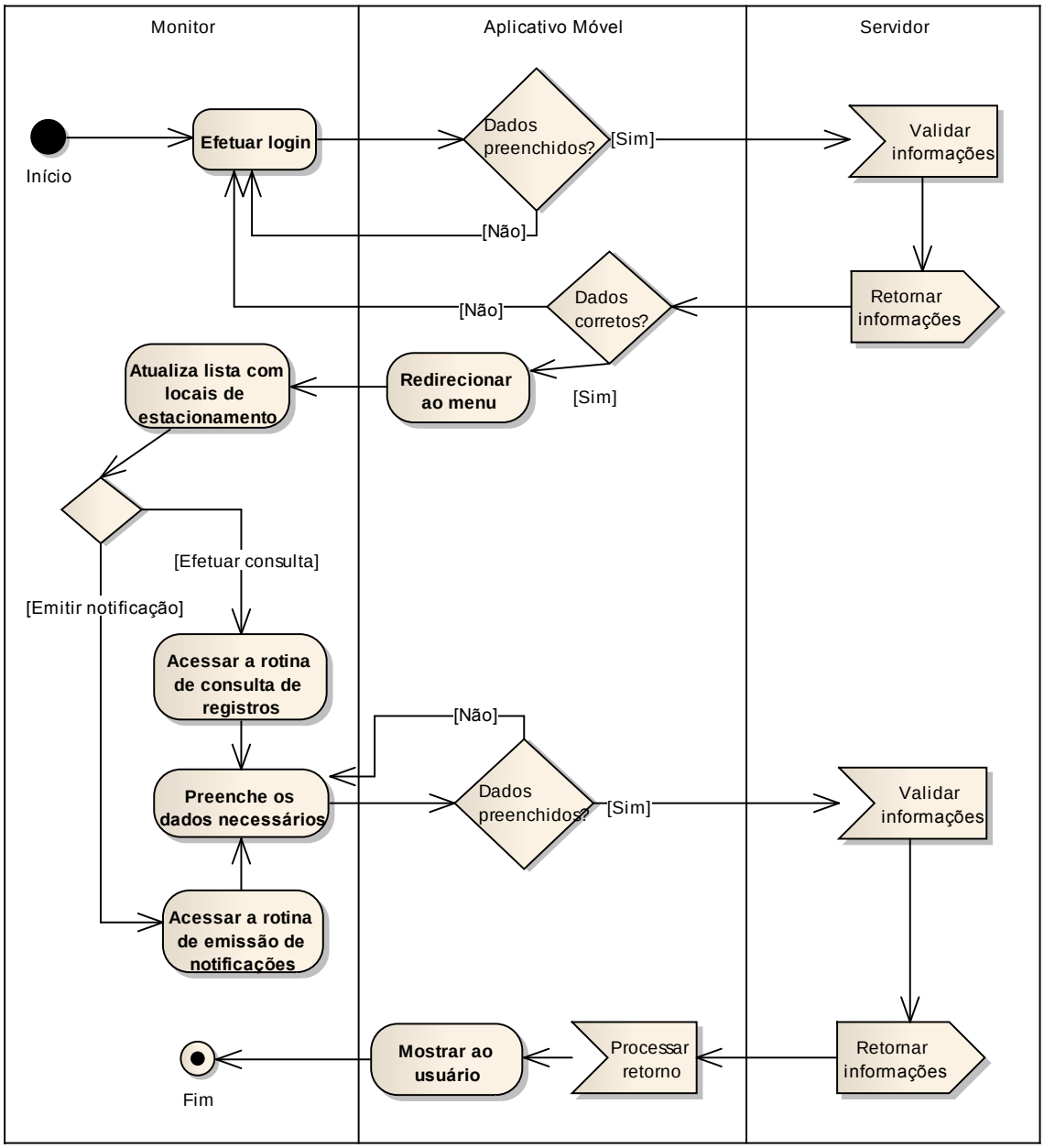
Figura 5 – Diagrama de atividades de consulta de registro de estacionamento



3.2.5 Diagrama de atividades de emissão de notificação

Na Figura 6 é apresentado o diagrama de atividades de emissão de notificação de irregularidade de estacionamento em local regulamentado pela Área Azul. O papel humano, assim como o item 3.2.4, é desempenhado pelo ator monitor.

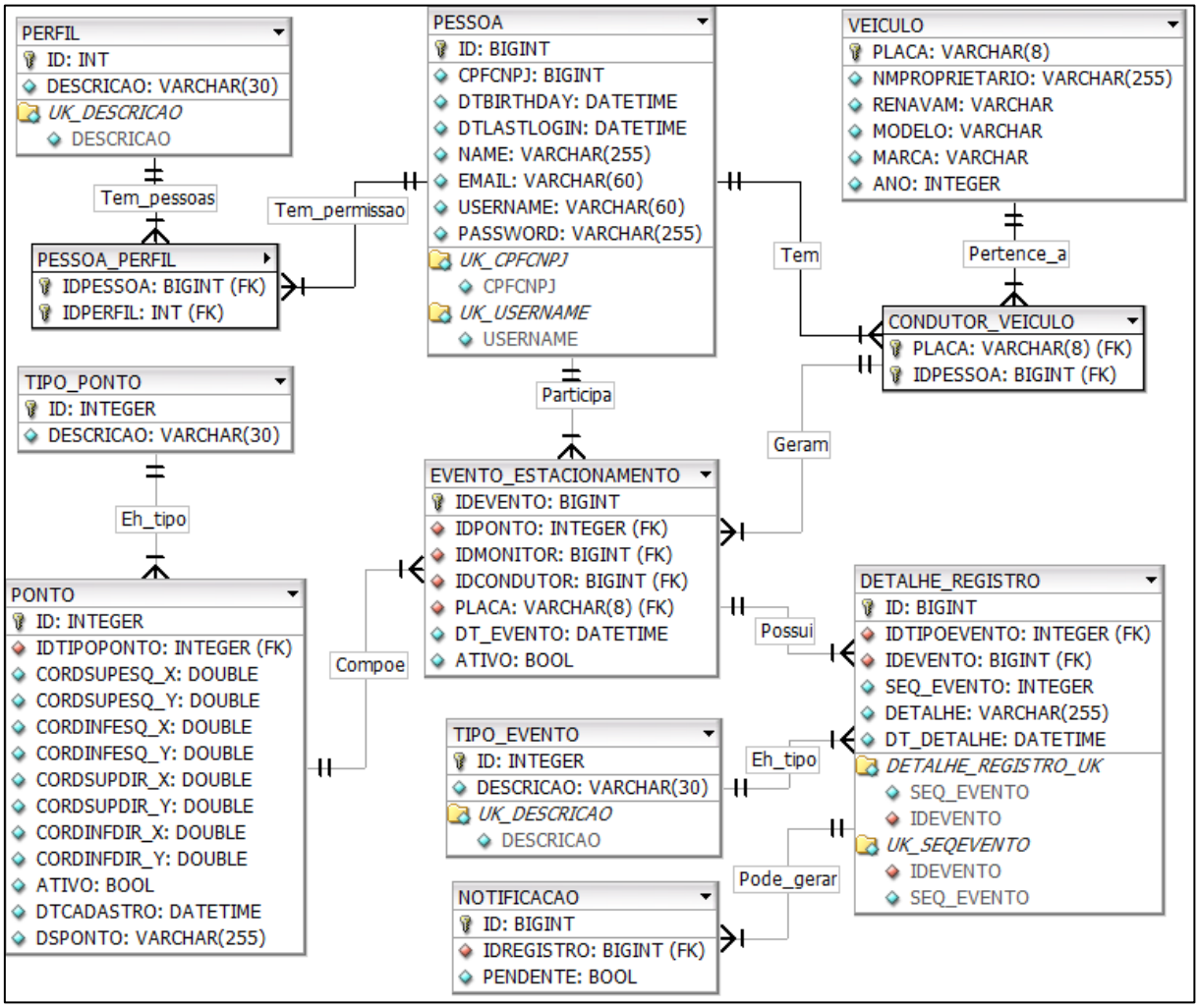
Figura 6 – Diagrama de atividades de emissão de notificação



3.2.6 Modelo de entidade e relacionamento

Na Figura 7 é exibido o modelo de entidade e relacionamento (MER) da aplicação desenvolvida. Para o modelo de entidade e relacionamento utiliza-se a notação *crows foot* e a ferramenta DB Designer 4. O dicionário de dados está apresentado no Apêndice B.

Figura 7 – Modelo de entidade e relacionamento



Abaixo é apresentada uma breve descrição das entidades criadas para o desenvolvimento do protótipo:

a) `perfil`: enumeração dos perfis disponíveis para acesso ao sistema;
b) `pessoa`: armazena as informações dos usuários;
c) `pessoa_perfil`: relacionamento que armazena as ligações entre os usuários e seus perfis de acesso;
d) `veiculo`: armazena os veículos cadastrados no sistema;
e) `condutor_veiculo`: entidade que armazena as relações entre os condutores e veículos. Permite a consulta dos veículos associados a uma pessoa e vice-versa;
f) `tipo_ponto`: enumeração dos tipos de pontos disponíveis;
g) `ponto`: armazena a relação dos pontos/locais disponíveis;
h) `evento_estacionamento`: armazena os eventos de estacionamento, sejam eles verificações, registros ou notificações;
i) `detalhe_registro`: armazena detalhes referentes aos eventos de

estacionamento;

j) `tipo_evento`: enumeração dos tipos de detalhes de evento possíveis;

k) `notificacao`: entidade que possui detalhes específicos sobre as notificações emitidas.

3.2.7 Diagrama de classes

A Figura 8 apresenta o diagrama de classes em nível de pacotes e sua respectiva comunicação. Os pacotes são definidos da seguinte forma:

a) `br.com.dao`: contém as classes do tipo *Data Access Object* (DAO). Estas classes são responsáveis por efetuar todo tipo de operação de acesso a objetos da base de dados, como consultas, persistência e exclusão de dados. Este pacote é utilizado pelos *Web Services* sob demanda;

b) `br.com.model`: contém os modelos cujas definições (atributos e métodos) originam-se do mundo real, representados através de classes;

c) `br.com.services.veiculo`: contém a classe `VeiculoService` que possui a notação `@WebService` (as classes que possuem esta notação tornam-se *web services* da aplicação e é possível efetuar chamadas externas, ou seja, através de outros aplicativos). Especificamente, contém os métodos para a inclusão e a consulta de dados dos veículos;

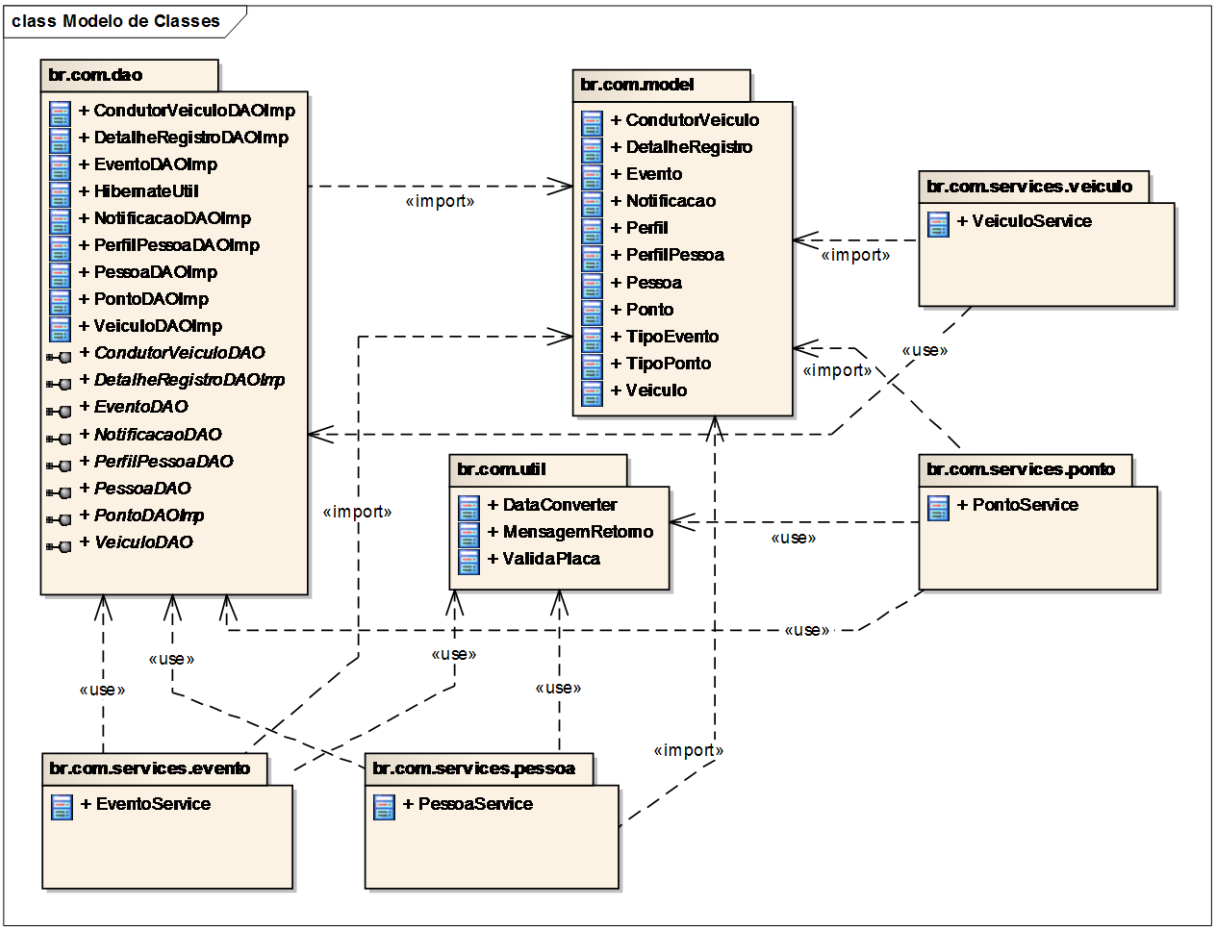
d) `br.com.services.ponto`: contém a classe `PontoService` que possui a notação `@WebService`. Possui os métodos que envolvem a inclusão, a consulta, a ativação e a inativação de pontos de estacionamento;

e) `br.com.services.evento`: contém a classe `EventoService` que possui a notação `@WebService`. Possui os métodos que envolvem a inclusão e a consulta de eventos de estacionamento, bem como a emissão e consulta de notificações;

f) `br.com.services.pessoa`: contém a classe `PessoaService` que possui a notação `@WebService`. Possui os métodos que envolvem a inclusão, a validação de dados (*login*) e a vinculação de usuários aos veículos;

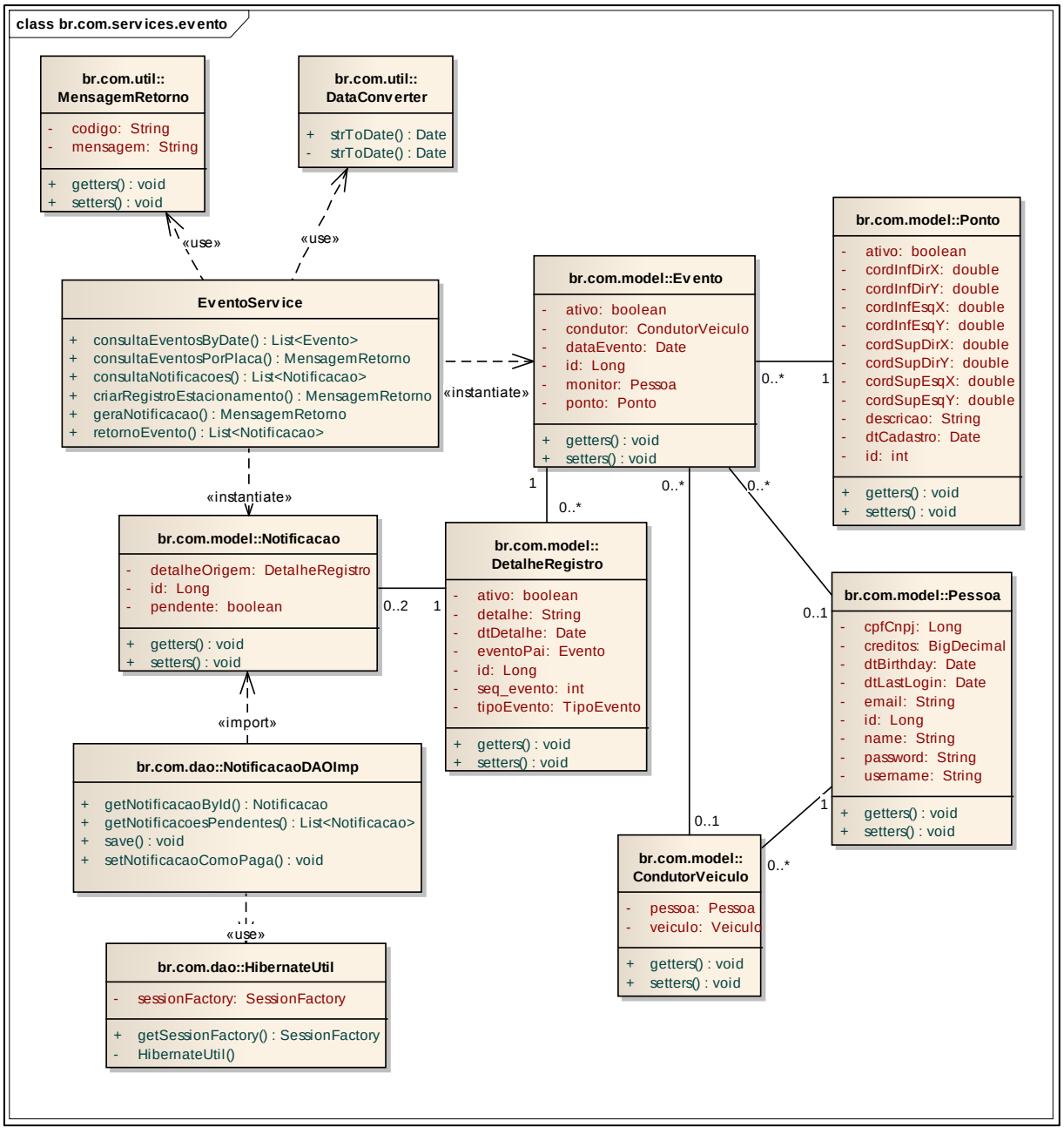
g) `br.com.util`: contém classes utilitárias reutilizadas durante o processo de desenvolvimento, para agilizar o tratamento e validação de campos constantes, como a formatação da data e da placa dos veículos e modelo da mensagem de retorno.

Figura 8 – Diagrama de classes em nível de pacotes



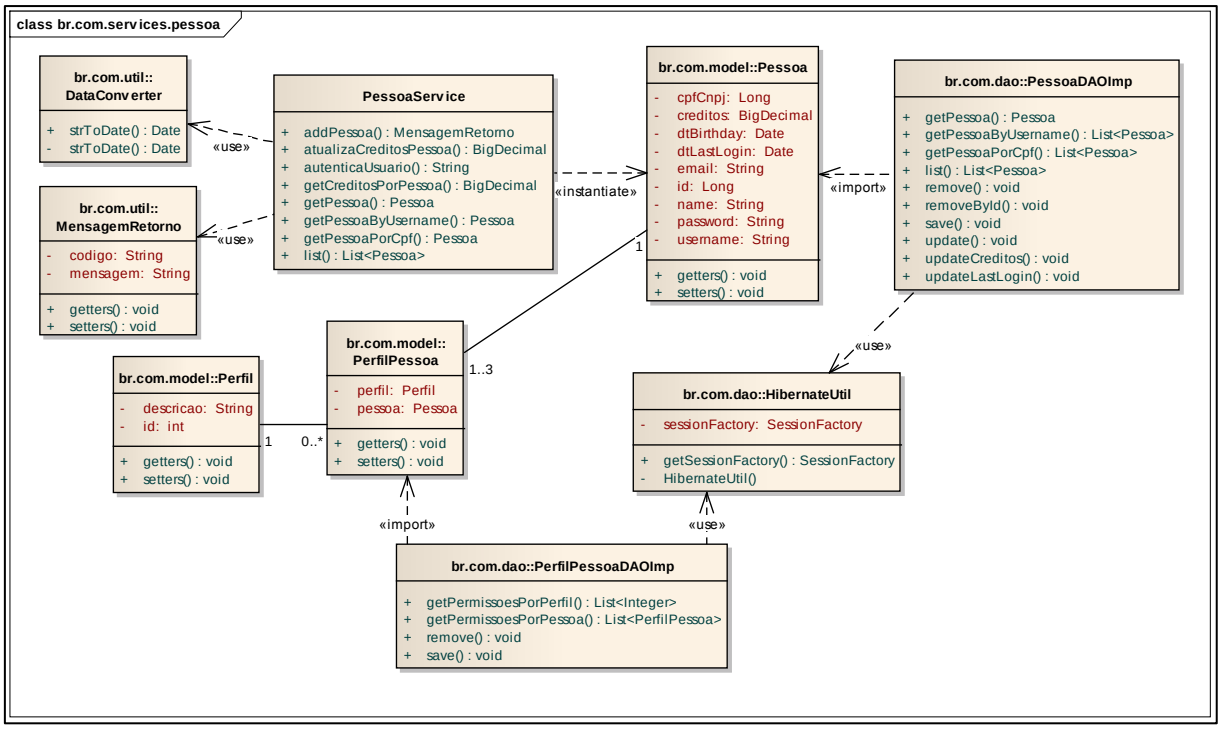
A comunicação com interfaces externas é efetuada através das classes `EventoService`, `VeiculoService`, `PessoaService` e `PontoService`. Embora haja relacionamentos entre várias classes, as classes acima não se relacionam diretamente, apenas compartilham de outras classes, como os modelos e objetos de persistência. Na Figura 9 é apresentado o diagrama de classes que envolve os serviços disponibilizados na classe `EventoService`.

Figura 9 – Diagrama de classes do serviço `EventoService`



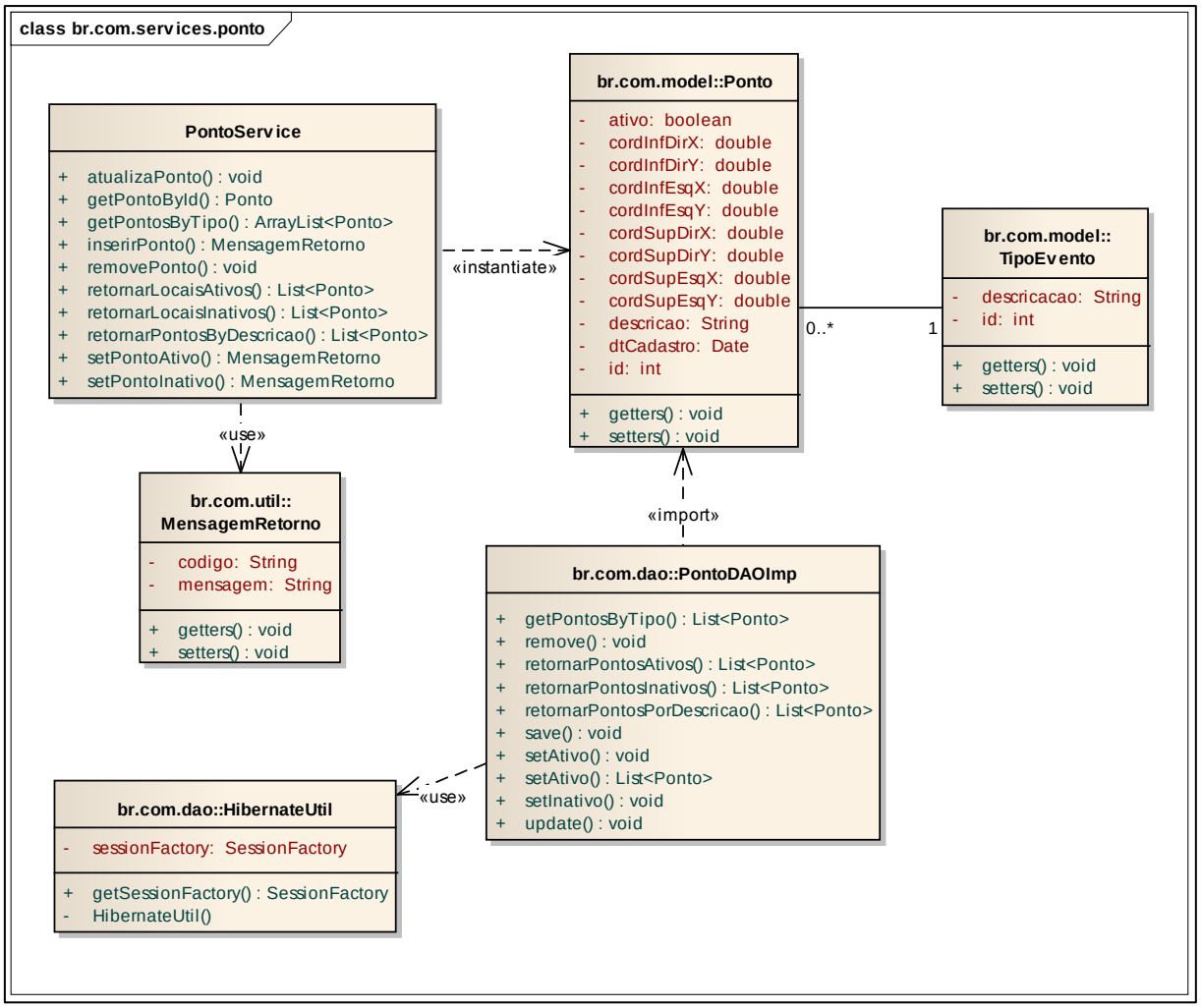
Na Figura 10 apresenta-se o diagrama de classes dos serviços disponibilizados pela classe `PessoaService`. Todos os métodos públicos desta classe podem ser chamados através do protocolo SOAP.

Figura 10 – Diagrama de classes do serviço `PessoaService`



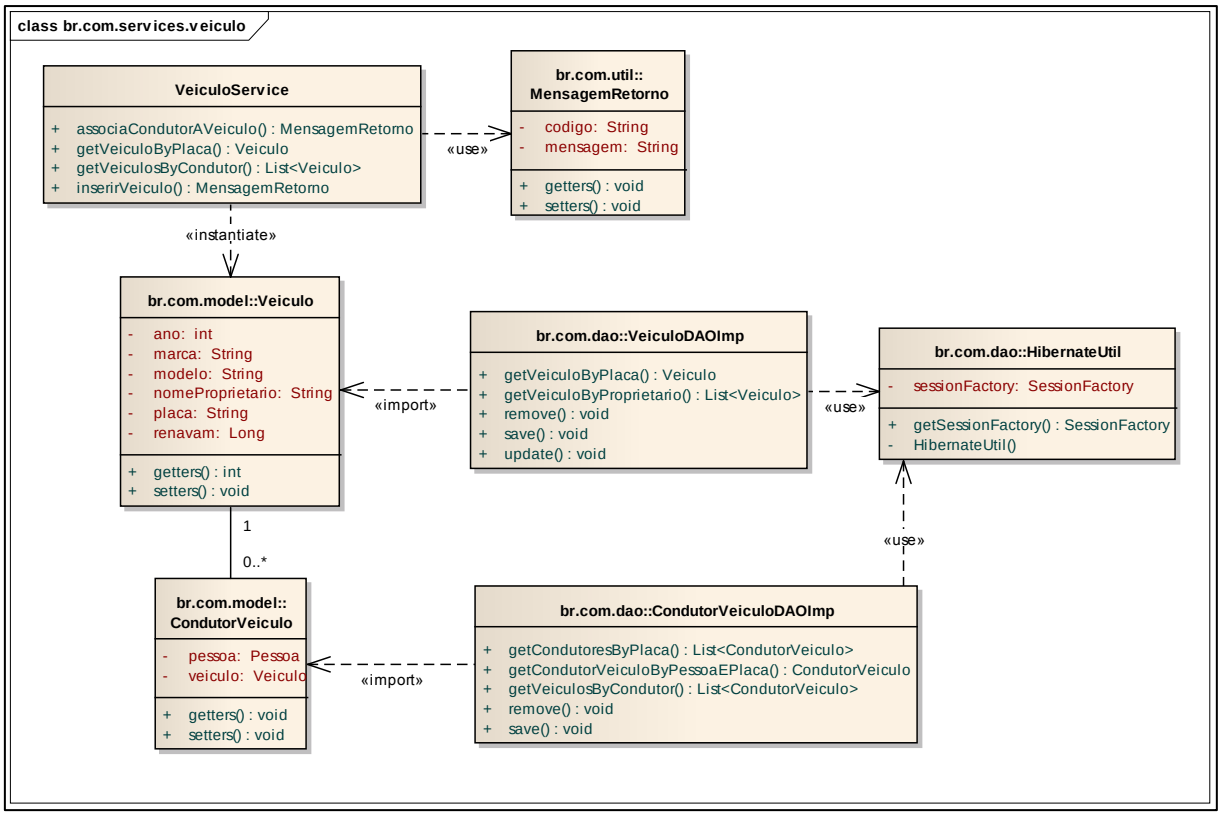
Na Figura 11 apresenta-se o diagrama de classes do serviço `PontoService`. Estes serviços são utilizados pelos usuários condutores e monitores, a fim de consultarem locais de estacionamento, e pelos operadores, a fim de realizarem operações mais complexas e restritas, como a ativação, inativação e cadastro de pontos.

Figura 11 – Diagrama de classes do serviço `PontoService`



Na Figura 12 apresenta-se o diagrama de classes do serviço `VeiculoService`. Na classe `VeiculoService` estão disponíveis os métodos para associar um condutor a um veículo, consultar veículos pela placa, inserir novos veículos e consultar os veículos associados a uma pessoa.

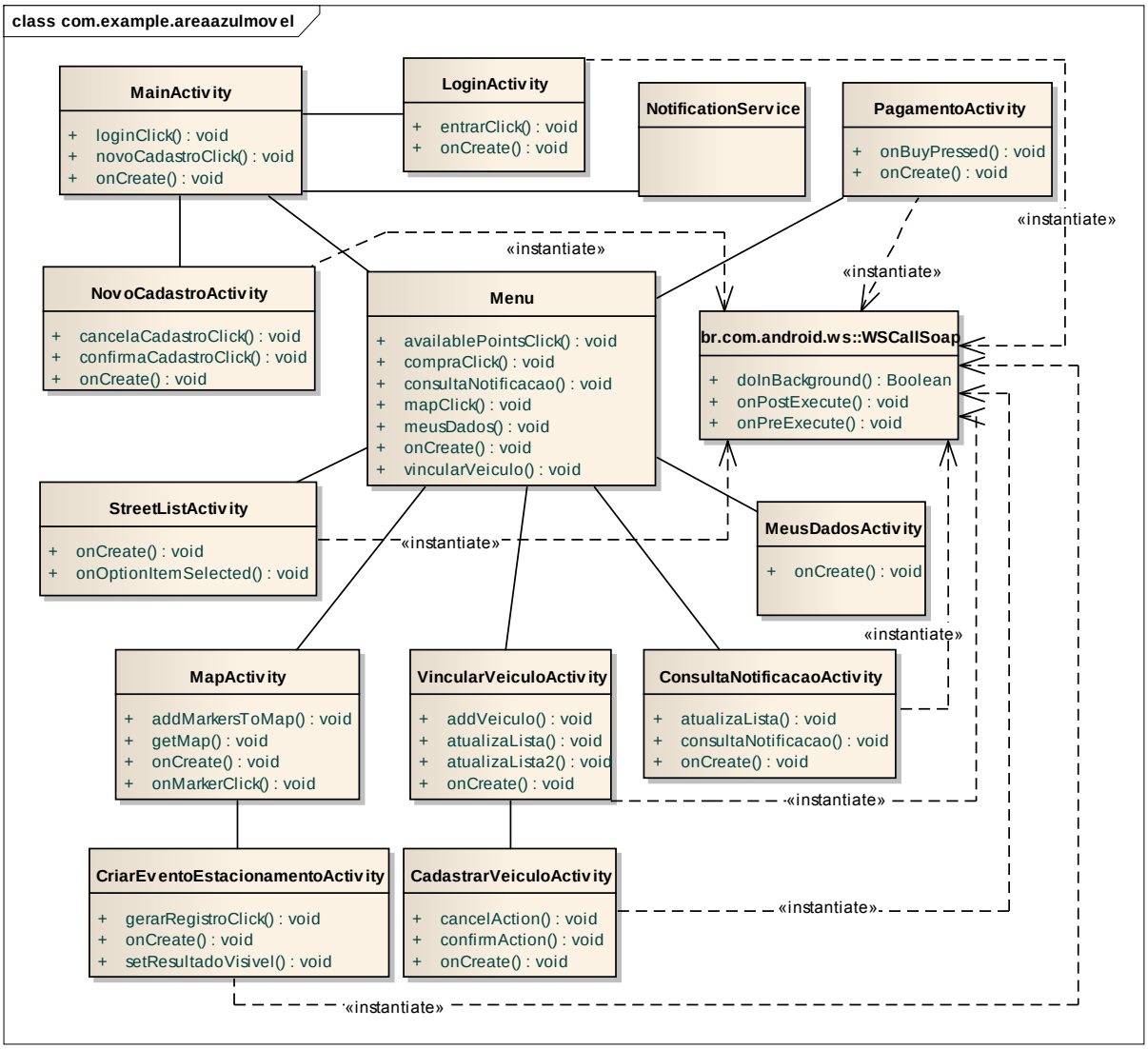
Figura 12 – Diagrama de classes do serviço `VeiculoService`



Observa-se, ainda, que os diagramas apresentados nas Figuras 9, 10, 11 e 12 possuem classes utilizadas entre si. Devido ao tamanho do diagrama geral, optou-se por dividir o diagrama do sistema servidor em sub diagramas, de acordo com os objetivos específicos de cada serviço, facilitando assim sua compreensão ainda que gerando redundância na representação de algumas entidades.

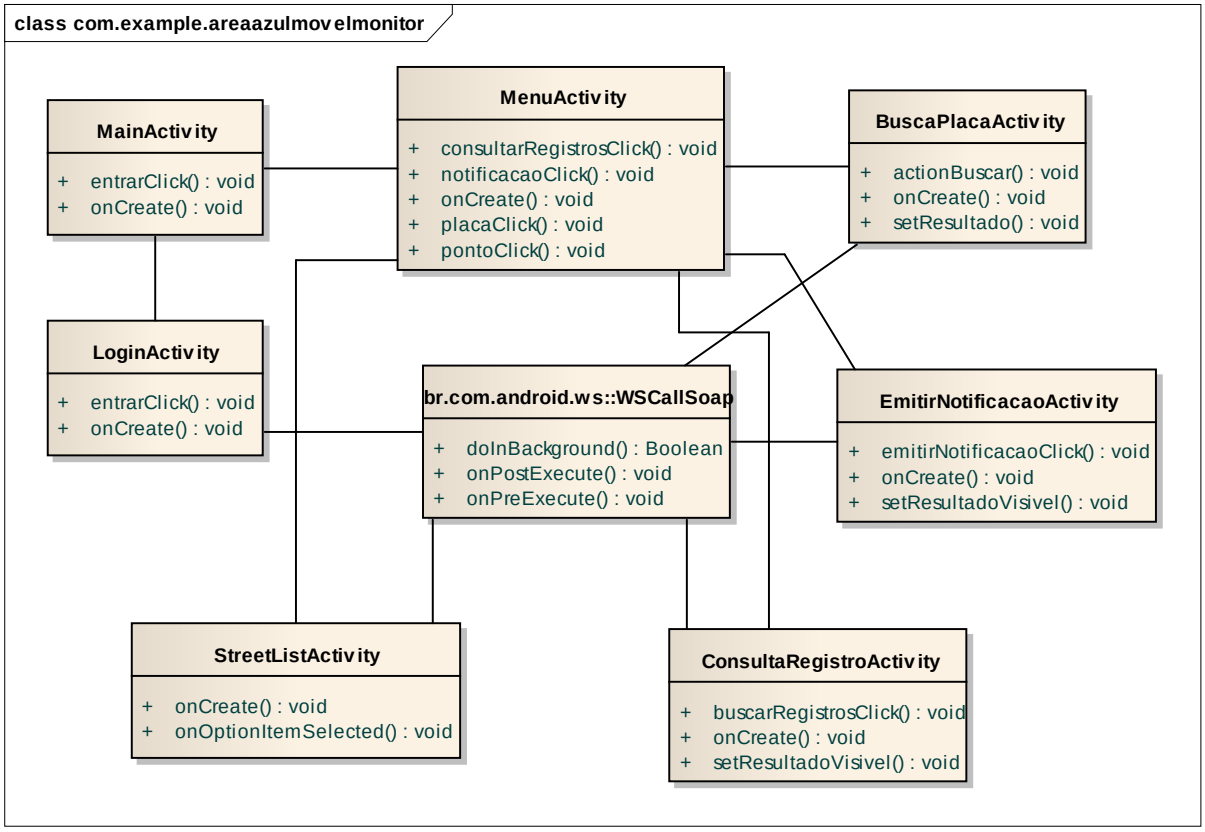
Na sequência, é apresentado na Figura 13 o diagrama de classes do aplicativo AreaAzulMovel. Este aplicativo consome os recursos do servidor através da classe `WSCallSoap`. Ao efetuar o *login* no aplicativo, a *activity* `Menu` é chamada e através dela as demais *activities* serão chamadas.

Figura 13 – Diagrama de classes do aplicativo AreaAzulMovel



Na Figura 14 é exibido o diagrama de classes do aplicativo AreaAzulMovelMonitor. Nota-se uma quantidade inferior de classes, pois a quantidade de funcionalidades é menor se comparada às funcionalidades do aplicativo AreaAzulMovel.

Figura 14 – Diagrama de classes do aplicativo AreaAzulMovelMonitor



3.3 IMPLEMENTAÇÃO

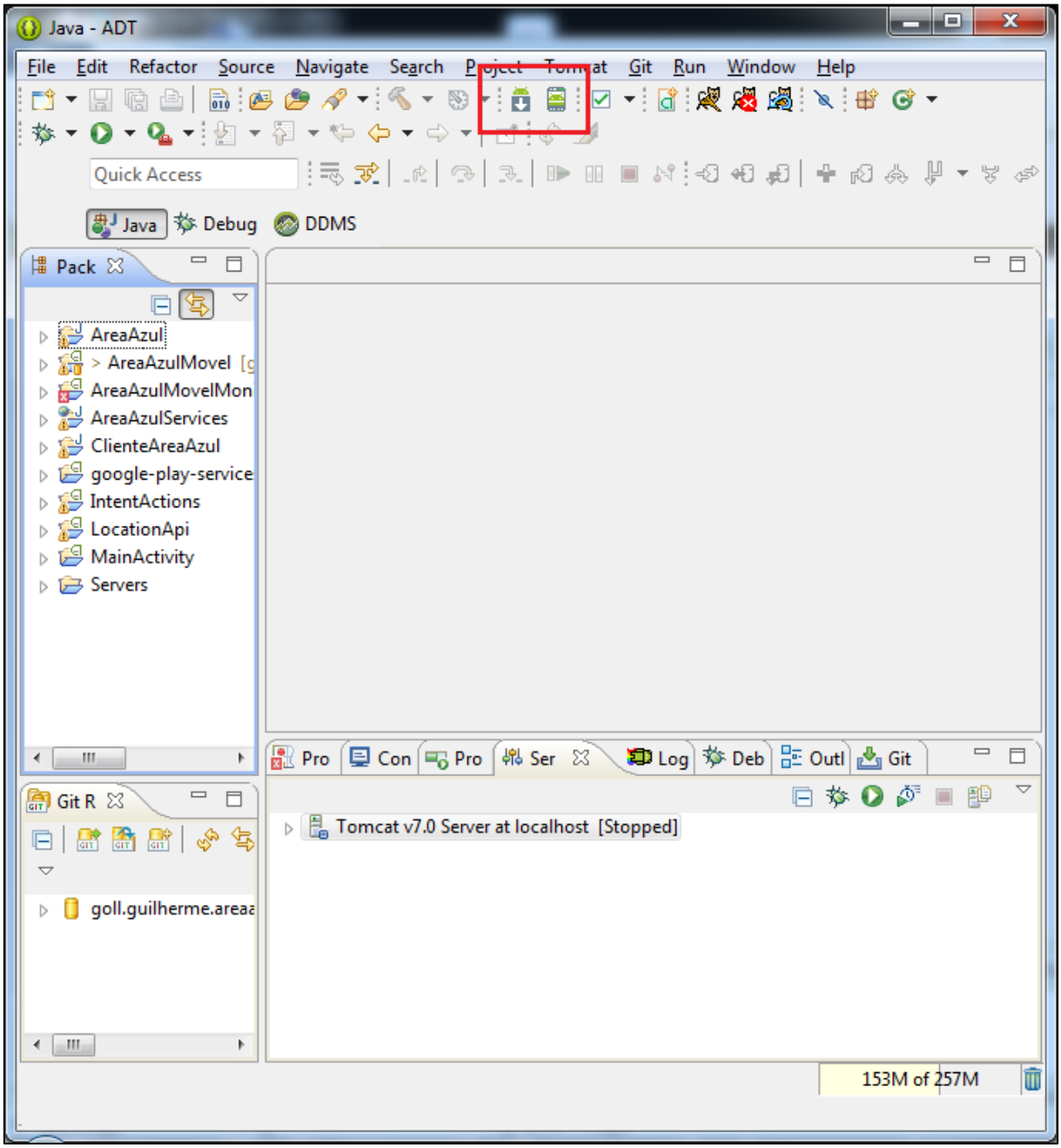
A seguir são apresentadas as técnicas e ferramentas utilizadas, bem como a operacionalidade do protótipo.

3.3.1 Técnicas e ferramentas utilizadas

Para desenvolvimento do protótipo servidor e dos protótipos de dispositivos móveis foi utilizada a IDE Eclipse com o *plugin* Android *Developer Tools* (ADT). Esta ferramenta provê as funcionalidades padrões da IDE Eclipse com as ferramentas adicionais necessárias para desenvolvimento de aplicativos para a plataforma Android. A linguagem de desenvolvimento utilizada, tanto para o protótipo servidor quanto para a plataforma móvel, é Java.

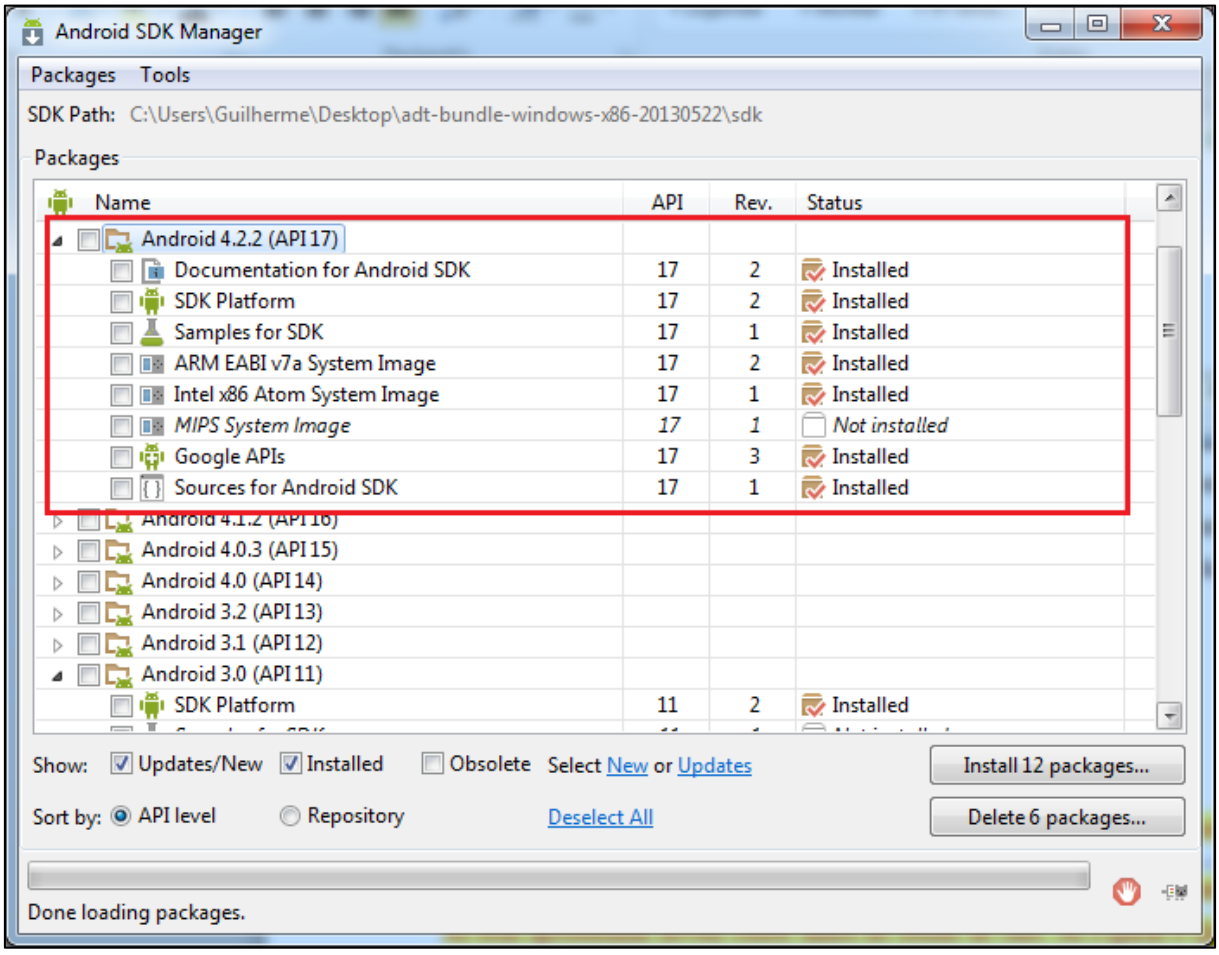
Na Figura 15 é exibida a sua tela inicial, destacando em vermelho os atalhos do ADT.

Figura 15 – Ambiente Eclipse com o *plugin* Android *Developer Tools*



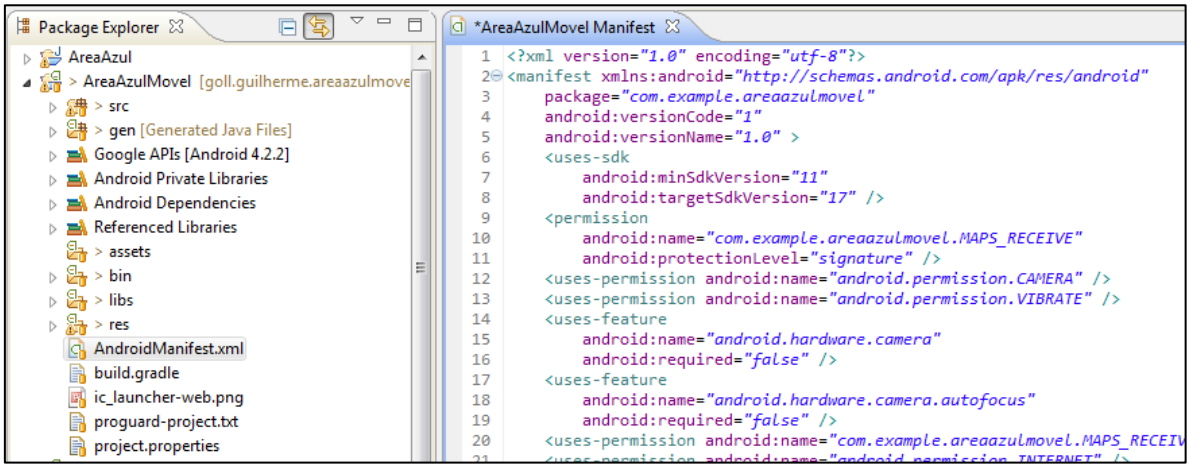
Para o desenvolvimento nesta plataforma móvel foi necessário o *download* da API 17, compatível com o Android versão 4.2.2, através da ferramenta Android SDK *Manager*, que pode ser visualizada na Figura 16 e que pode ser acessada através da IDE Eclipse ou em seu diretório de instalação.

Figura 16 – Android SDK *Manager*



Os aplicativos desenvolvidos para a plataforma Android seguem determinada estrutura de arquivos. Na Figura 17 é apresentada esta estrutura e também o arquivo `AndroidManifest.xml`, explicados adiante.

Figura 17 – Estrutura de aplicativo para Android



Entre os arquivos e diretórios mais importantes, pode-se citar:

a) `AndroidManifest.xml`: arquivo obrigatório para todo aplicativo Android.

Nele estão registradas todas as classes que herdam de `Activity`, além das permissões que o aplicativo precisará solicitar ao usuário (quando da instalação do mesmo no dispositivo). Estão presentes, ainda, a versão compatível com o aplicativo e parâmetros de configuração como temas e configurações adicionais de sistema;

b) `src`: diretório no qual o código-fonte do aplicativo está armazenado;

c) `libs`: diretório no qual as bibliotecas externas estão armazenadas. Como exemplo, pode-se citar a biblioteca do PayPal, através da qual a aquisição de créditos é realizada;

d) `res`: diretório no qual todos os arquivos de imagens, *layouts* de tela (no formato XML) e contexto de valores estão armazenados. Estes arquivos costumam ser modificados regularmente durante o desenvolvimento de um aplicativo.

As interações com o usuário são efetuadas através de *activities*, classes Java que estendem a superclasse `Activity`. Quase todas as *activities* interagem com o usuário. Elas são gerenciadas através de uma pilha de *activities*. Sempre que uma *activity* é iniciada é colocada no topo da pilha e torna-se a *activity* em execução, enquanto a *activity* que a chamou vai para segundo plano até que a nova *activity* seja encerrada (ANDROID DEVELOPERS, 2013c).

Segundo Android Developers (2013c) uma *activity* possui basicamente quatro estados:

a) ativa ou em execução: quando a *activity* está em primeiro plano na interface (topo da pilha);

b) pausada: quando a *activity* perdeu o foco mas continua visível. Seus dados permanecem intactos, mas ela pode ser encerrada pelo sistema em situações de memória extremamente baixa;

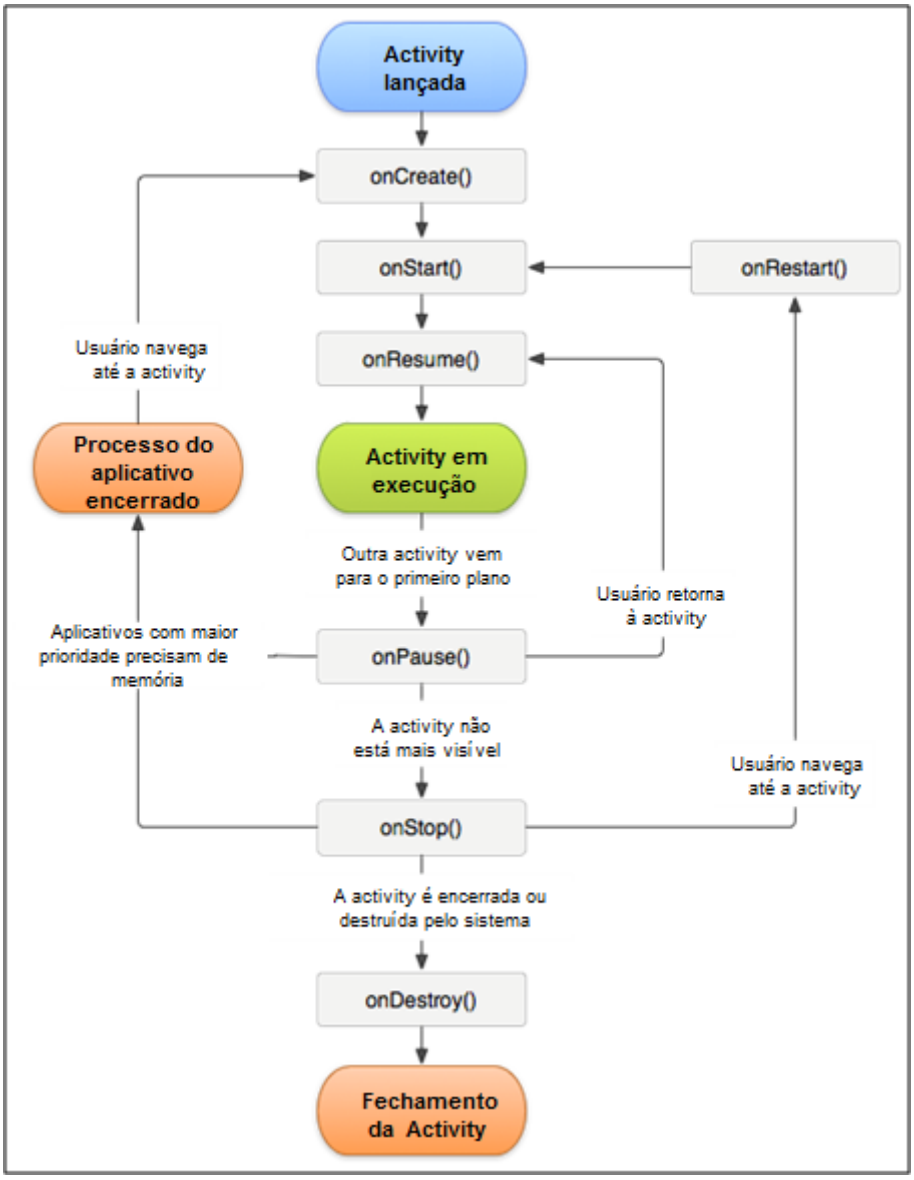
c) parada: quando a *activity* foi completamente sobreposta por outra *activity*. Seus dados permanecem intactos, porém poderá ser eliminada mais facilmente pelo sistema, sempre que este precisar de memória para qualquer propósito;

d) quanto a *activity* é pausada ou parada, o sistema poderá eliminá-la. Quando ela for exibida novamente para o usuário, deverá ser completamente reiniciada e restaurada para seu estado anterior.

Na Figura 18 é exibido o diagrama de ciclo de vida de uma *activity*. Os retângulos representam os métodos de *callback* que podem ser implementados para executar operações durante a mudança de estados de uma *activity*. Os objetos ovalados coloridos representam os

estados principais que uma *activity* pode apresentar.

Figura 18 – Ciclo de vida da *Activity*



Fonte: Android Developers (2013b, tradução nossa).

Para a parte servidor foi utilizado ainda o contêiner *Web* Apache Tomcat 7.0. Com ele pode-se publicar serviços *web* para aplicações construídas na linguagem Java. No protótipo desenvolvido fora criada a interface servidor com o uso de dois projetos:

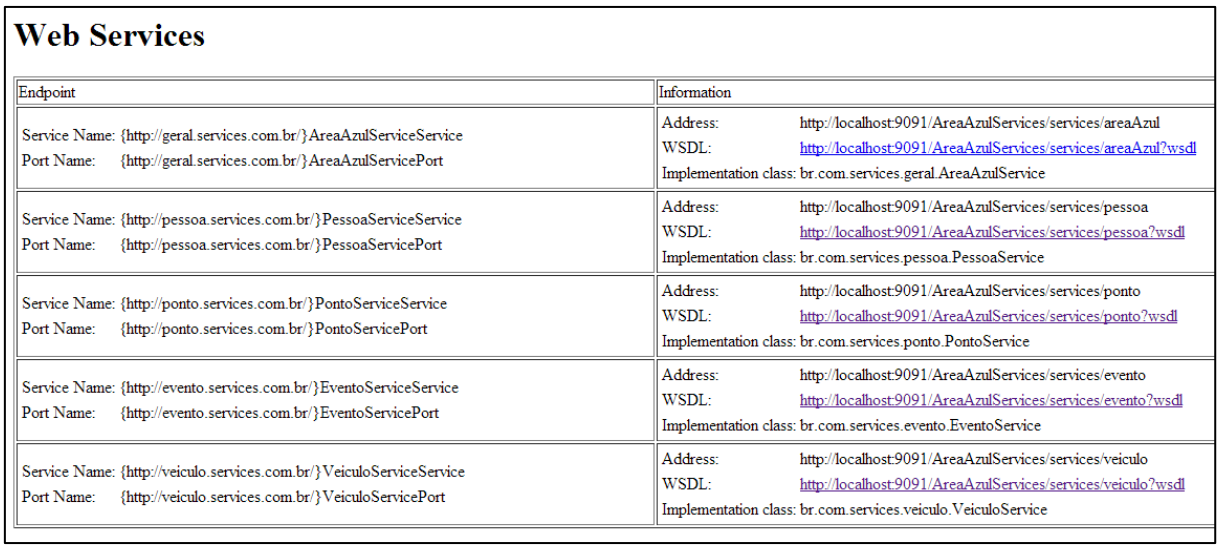
a) `AreaAzulServices`: no qual foi desenvolvida a publicação do w*eb service*, mapeamento de classes com o DWR para chamadas através de JavaScript, e a interface *web* do AreaAzulMovelWeb;

b) `AreaAzul`: no qual foi implementada a camada de processamento e retorno de requisições contendo toda a regra de negócio do protótipo (*back-end*).

O projeto `AreaAzulServices` pode ser executado de duas formas: através da IDE

Eclipse ou efetuando-se o *deploy* do arquivo `AreaAzulServices.war` dentro do diretório `webapps` do Apache Tomcat. Ao ser iniciado o serviço do contêiner *web*, inicia-se a publicação do serviço referente ao projeto `AreaAzul`. O contêiner *web* consome os artefatos mapeados e permite que aplicações utilizando o protocolo SOAP acessem os serviços e métodos disponíveis. Os mesmos são listados através da interface chamada *endpoint*, que pode ser visualizada na Figura 19.

Figura 19 – *Endpoint* de *Web Service*



Web Services

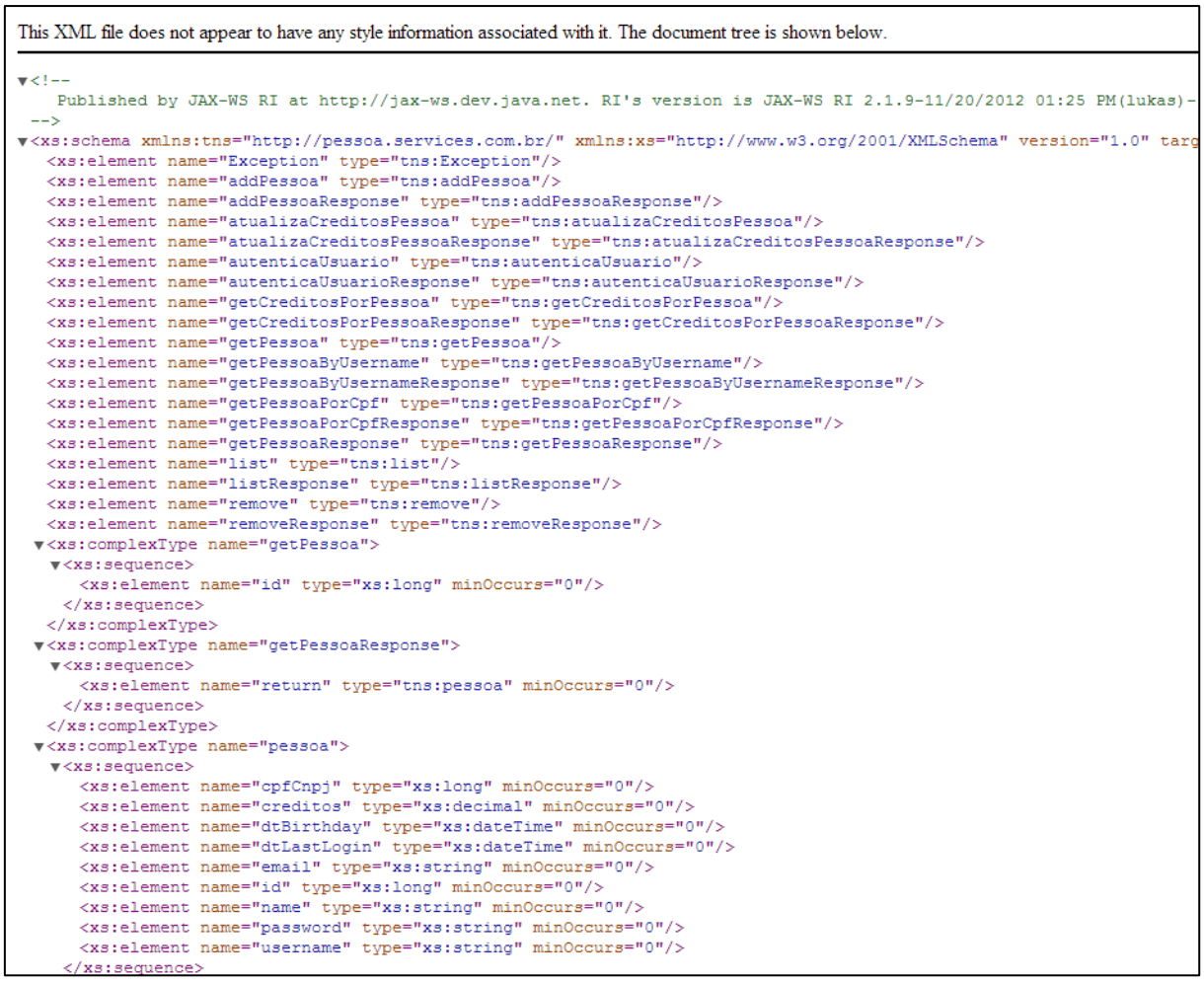
Endpoint	Information
Service Name: {http://geral.services.com.br/}AreaAzulServiceService Port Name: {http://geral.services.com.br/}AreaAzulServicePort	Address: http://localhost:9091/AreaAzulServices/services/areaAzul WSDL: http://localhost:9091/AreaAzulServices/services/areaAzul?wsdl Implementation class: br.com.services.geral.AreaAzulService
Service Name: {http://pessoa.services.com.br/}PessoaServiceService Port Name: {http://pessoa.services.com.br/}PessoaServicePort	Address: http://localhost:9091/AreaAzulServices/services/pessoa WSDL: http://localhost:9091/AreaAzulServices/services/pessoa?wsdl Implementation class: br.com.services.pessoa.PessoaService
Service Name: {http://ponto.services.com.br/}PontoServiceService Port Name: {http://ponto.services.com.br/}PontoServicePort	Address: http://localhost:9091/AreaAzulServices/services/ponto WSDL: http://localhost:9091/AreaAzulServices/services/ponto?wsdl Implementation class: br.com.services.ponto.PontoService
Service Name: {http://evento.services.com.br/}EventoServiceService Port Name: {http://evento.services.com.br/}EventoServicePort	Address: http://localhost:9091/AreaAzulServices/services/evento WSDL: http://localhost:9091/AreaAzulServices/services/evento?wsdl Implementation class: br.com.services.evento.EventoService
Service Name: {http://veiculo.services.com.br/}VeiculoServiceService Port Name: {http://veiculo.services.com.br/}VeiculoServicePort	Address: http://localhost:9091/AreaAzulServices/services/veiculo WSDL: http://localhost:9091/AreaAzulServices/services/veiculo?wsdl Implementation class: br.com.services.veiculo.VeiculoService

Por trás de cada *endpoint* publicado há um XML no formato *Web Services Description Language* (WSDL), através do qual pode-se consultar os métodos disponibilizados e seus respectivos parâmetros para consumo e formato de retorno, porém de forma superficial, ou seja, sem a definição de tipos de dados ou de nomes de variáveis. Na Figura 20 exibe-se a interface WSDL do serviço `PessoaService`, através da qual operações de cadastro e consulta são efetuados.

Figura 20 – Acesso à WSDL do serviço `PessoaService` através do navegador

Para a definição detalhada dos parâmetros de cada método e respectivo tipo de retorno, cada WSDL define, através da propriedade `schemaLocation`, o endereço do arquivo XML *Schema Definition* (XSD). Na Figura 21 é apresentado o arquivo XSD do serviço `PessoaService`.

Figura 21 – Acesso ao XSD do serviço `PessoaService` através do navegador



Para o desenvolvimento da parte *web* do protótipo utilizou-se a tecnologia JavaServer *Pages* (JSP) com o *framework* Twitter Bootstrap. O JSP é uma tecnologia que provê formas rápidas e simplificadas de construir conteúdo *web* dinâmico. Ele permite o desenvolvimento rápido de aplicações *web* independentes da plataforma servidor e cliente. Para isto foi implementado o JavaServer *Pages Standard Tag Library* (JSTL), que trata-se de uma coleção de bibliotecas de *tags* que implementam funcionalidades para uso geral, comuns em muitas aplicações *web*, como tarefas estruturais (iterações e condições), *tags* para manipulação de documentos XML e *tags* SQL (ORACLE, 2013).

O Twitter Bootstrap, por sua vez, é um intuitivo e poderoso *framework*, com características visuais simples para desenvolvimento *web*. Ele disponibiliza uma série de componentes que podem ser utilizados no desenvolvimento de páginas *web* com a aplicação de estilos e formatações, o que torna o processo de codificação mais rápido e fácil (BOOTSTRAP, 2013).

Cada tela do AreaAzulMovelWeb está contida em um arquivo do tipo `.jsp`,

construídas utilizando as linguagens HTML e JavaScript para a parte estrutural (tabelas, ações e funções) e o *Cascading Style Sheets* (CSS) do *framework* Twitter Bootstrap para a formatação do conteúdo. Para utilizar de funcionalidades adicionais, como o CSS do Twitter Bootstrap e as classes mapeadas através do DWR, é necessário fazer referências no arquivo `.jsp` que indiquem a importação destes artefatos externos. A importação é feita na seção `<head>` do arquivo `.jsp`. Na Figura 22 é apresentado o arquivo `index.jsp`, que contém importações das interfaces DWR e do CSS `bootstrap.css`.

Figura 22 – Arquivo `index.jsp`

3.3.2 Operacionalidade da implementação

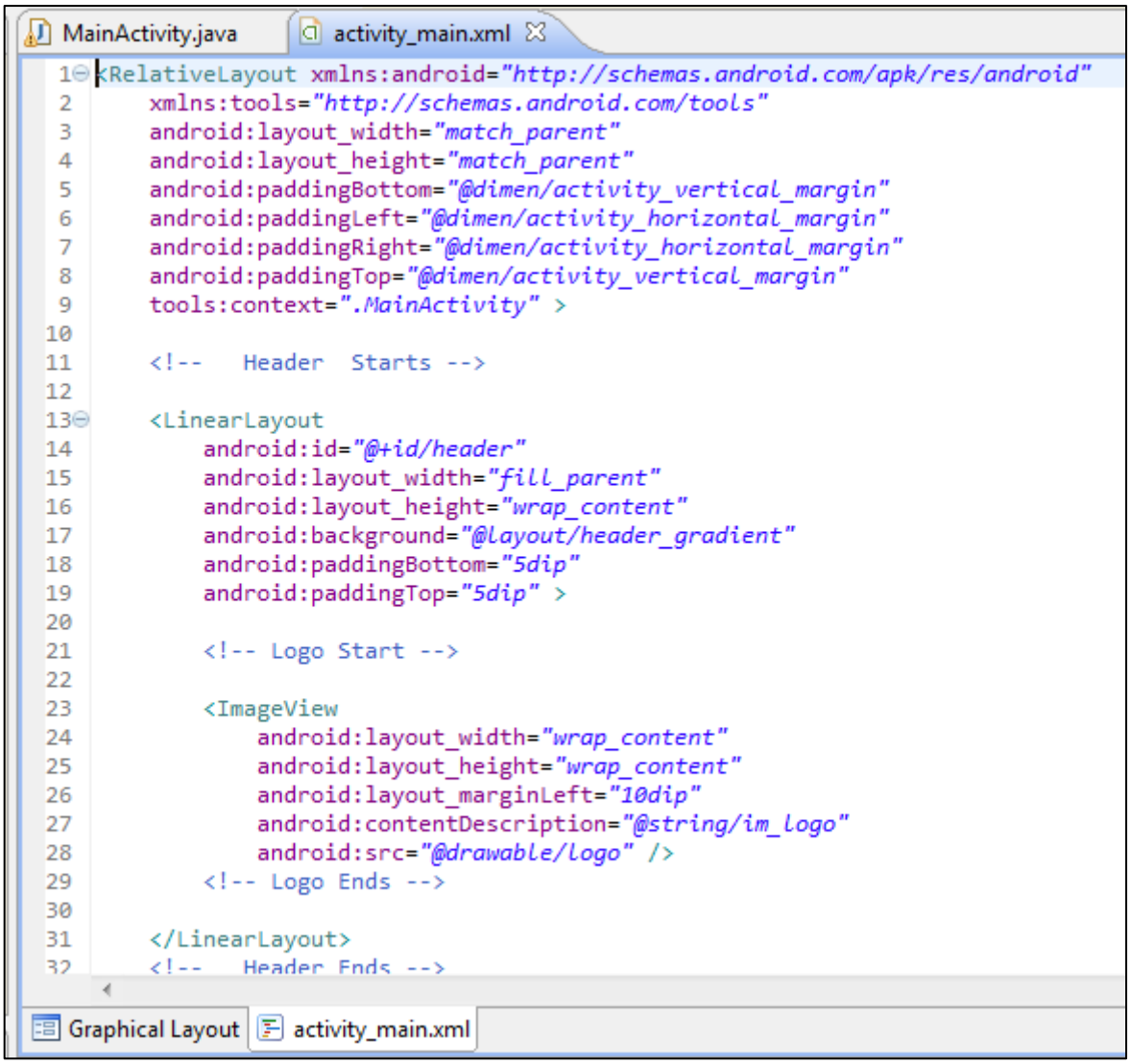
Nesta subseção será apresentada a operacionalidade do sistema sob três perspectivas: do condutor, do monitor e do operador.

Sob a perspectiva do condutor, este efetuará o acesso ao aplicativo através do respectivo ícone em seu *smartphone* ou *tablet*. A interface inicial do sistema permite ao usuário efetuar o *login* ou criar um novo cadastro, caso não possua. Estas funcionalidades atendem aos requisitos funcionais RF02 e RF01, respectivamente. O cadastro efetuado através deste aplicativo concede acesso somente ao módulo para condutores. Na Figura 23 exibe-se a interface inicial.

Figura 23 – Interface inicial do aplicativo para o condutor

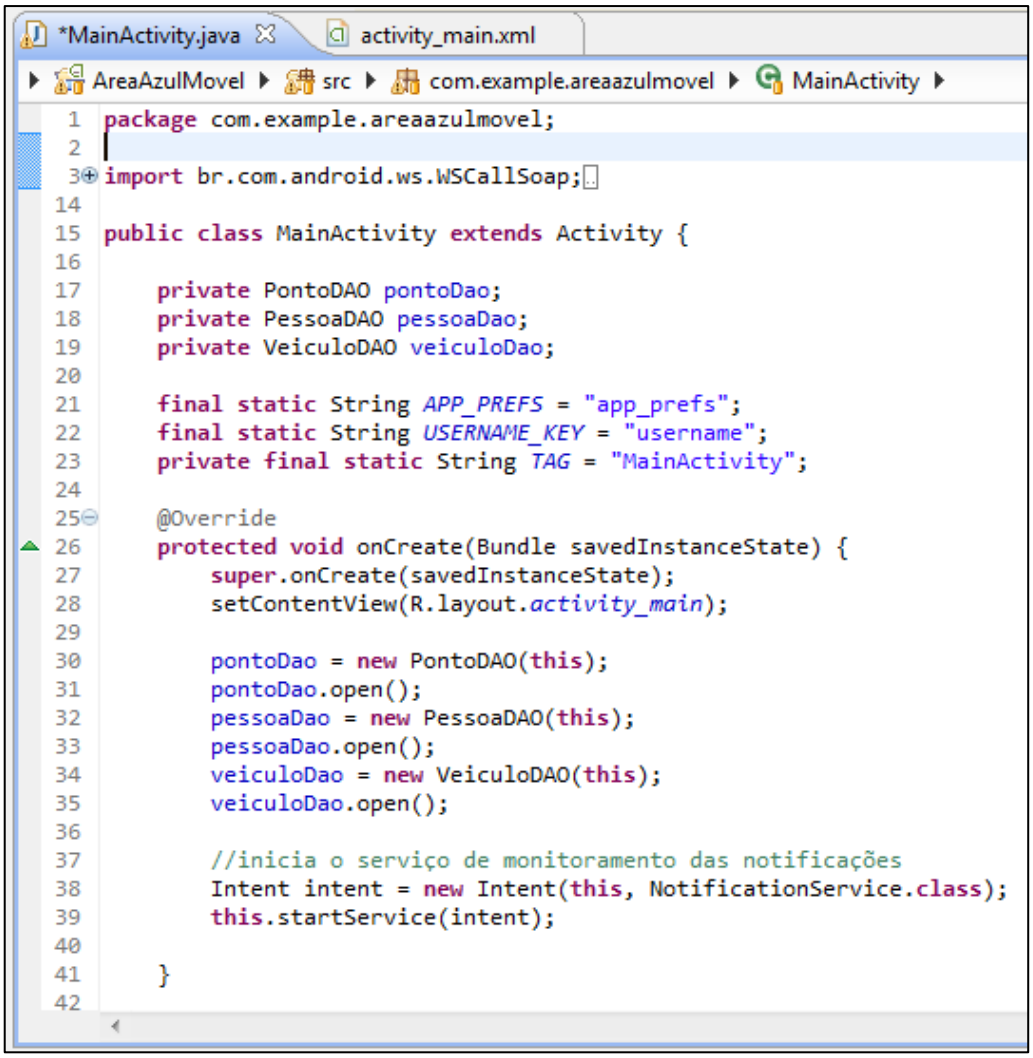
O arquivo que contém o *layout* desta interface inicial chama-se `activity_main.xml` e está associado à *activity* `MainActivity`, através da atribuição da propriedade `tools:context=".MainActivity"`. Na Figura 24 exibe-se o arquivo `activity_main.xml`.

Figura 24 – Arquivo `activity_main.xml`



De forma semelhante, na *activity* determina-se qual o arquivo XML será utilizado para interação com o usuário. Sempre que uma *activity* é chamada, seu clico de vida é implementado. O primeiro método chamado é o `onCreate()` e nele é que o método `setContentView()` deve ser implementado de forma a referenciar o *layout* XML com a interface para o usuário. Na Figura 25 exibe-se a classe `MainActivity`. Esta classe está marcada com a propriedade `<category android:name="android.intent.category.LAUNCHER" />` no arquivo `AndroidManifest.xml`, o que significa que quando o aplicativo for aberto esta é a *activity* que deve ser chamada.

Figura 25 – Classe `MainActivity`



Para cada botão é implementado um respectivo método que atua como um *listener* (ouvinte). O *listener* é responsável por efetuar os procedimentos necessários, entre eles a chamada de uma nova *activity*, quando seu gatilho for disparado. Na Figura 26 é exibida a implementação do *listener* para o botão `Login`. Quando o mesmo é pressionado é criado um novo *intent* (intenção) e a execução deste *intent* é realizada através da chamada dos métodos `startActivity` ou `startActivityForResult`.

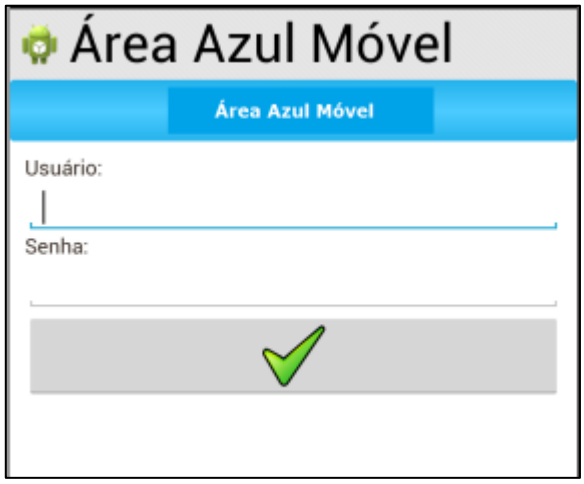
Figura 26 – *Listener* para o botão `Login`

```
/*
 * Abre a tela de login
 */
public void loginClick(View v) {
    Intent it = new Intent(this, LoginActivity.class);
    startActivityForResult(it, 1);
}
```

Após pressionado o botão `Login`, é criado um novo *intent* referenciando a *activity*

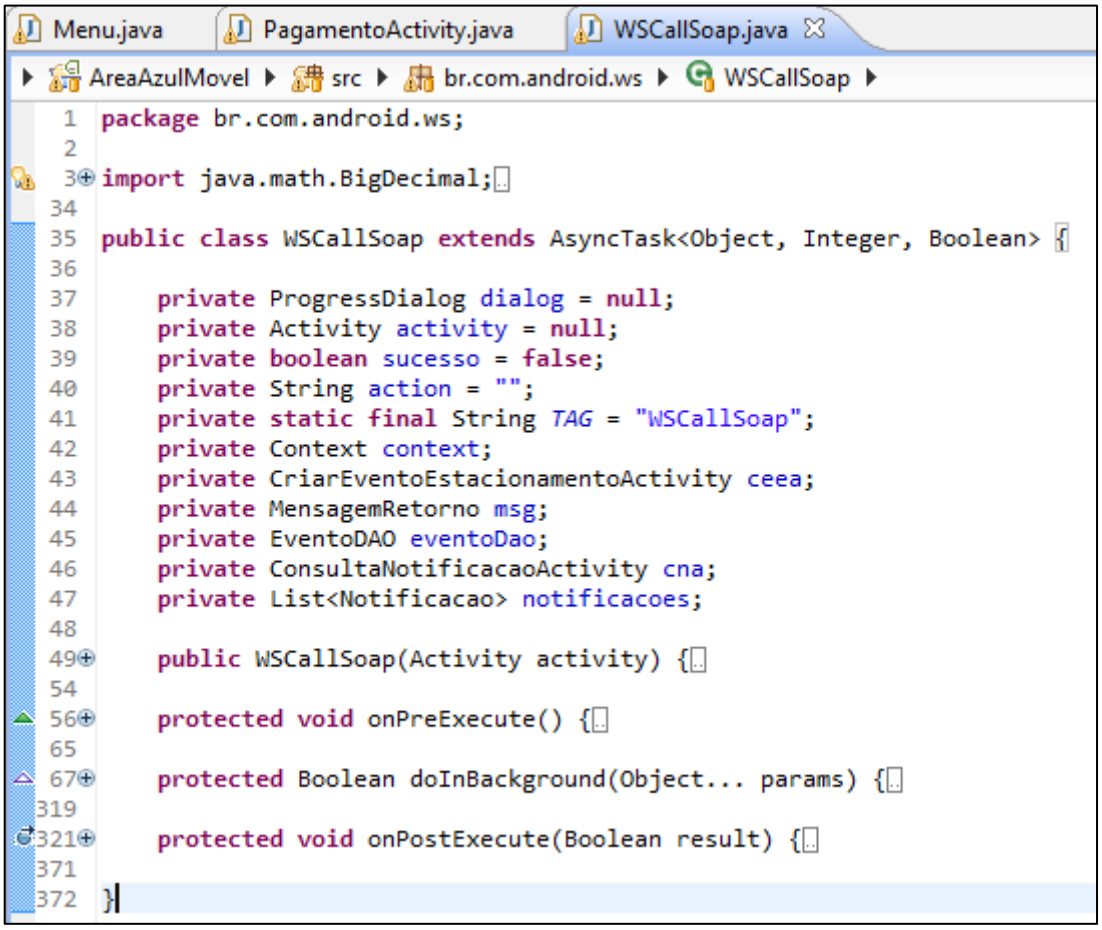
`LoginActivity`. O método `startActivityForResult` é então executado e a *activity* que originou a chamada, `MainActivity`, entra no estado de pausa, passando para o segundo plano na pilha de *activities* e dando espaço à *activity* `LoginActivity` para interagir com o usuário. A tela de *login* é aberta para o usuário, que deverá preencher as informações e confirmar a operação. Na Figura 27 é exibida a tela de *login* da aplicação.

Figura 27 – Tela de *login*



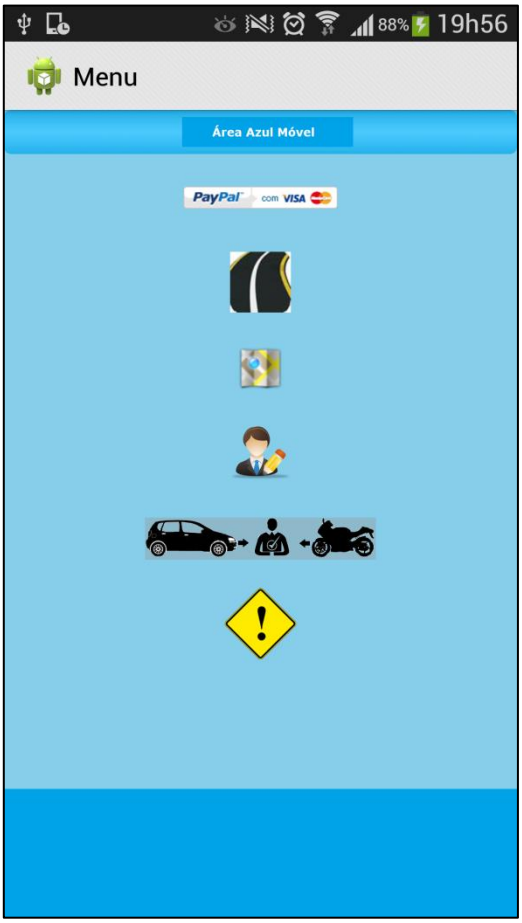
Quando o botão de confirmação é pressionado uma verificação é efetuada a fim de confirmar se usuário e senha estão preenchidos e não são nulos. É disparada então uma tarefa em segundo plano, responsável for efetuar a chamada do w*eb service*. A classe responsável por efetuar todas as chamadas ao w*eb service* é a `WSCallSoap`, que estende a classe `AsyncTask`, pertencente à biblioteca do Android. Todas as classes que estendem a classe `AsyncTask` devem implementar seu ciclo de vida, semelhante ao aplicado pela classe `Activity`. Primeiramente é chamado o método `onPreExecute`(), antes da execução da tarefa, na sequência é chamado o método `doInBackground`(), no qual o conjunto principal de procedimentos é executado, e por fim é chamado o método `onPostExecute`(), que possibilita à tarefa efetuar procedimentos adicionais, como a entrada de dados (*input*) na interface do usuário. Na Figura 28 é exibida a implementação da classe `WSCallSoap` e seu ciclo de vida. Além dos métodos padrões, é possível criar novos métodos, como é o caso do construtor da classe.

Figura 28 – Implementação da classe WSCallSoap e seu ciclo de vida



Após efetuado o *login*, o sistema resgata, em segundo plano, os dados do usuário autenticado e os veículos associados ao seu perfil. Para isto são iniciadas duas instâncias da classe a `WSCallSoap`, e no método `onPostExecute`() atualiza-se os dados do usuário autenticado e os veículos associados ao seu perfil. O usuário é redirecionado ao menu do sistema, exibido na Figura 29.

Figura 29 – Menu do aplicativo AreaAzulMovel



Através do menu o usuário poderá adquirir novos créditos de estacionamento, visualizar uma lista das ruas disponíveis para estacionamento, acessar o mapa para efetuar o registro de estacionamento, consultar seus dados, efetuar a vinculação de novos veículos ao seu perfil e consultar notificações emitidas contra veículos associados ao seu perfil.

Ao acessar o módulo de aquisição de créditos, a API do PayPal é chamada logo que o método `startActivity` executado, passando como parâmetro o *intent* com a classe `PagamentoActivity`, conforme é possível observar na Figura 30. Esta funcionalidade atende ao requisito funcional RF04.

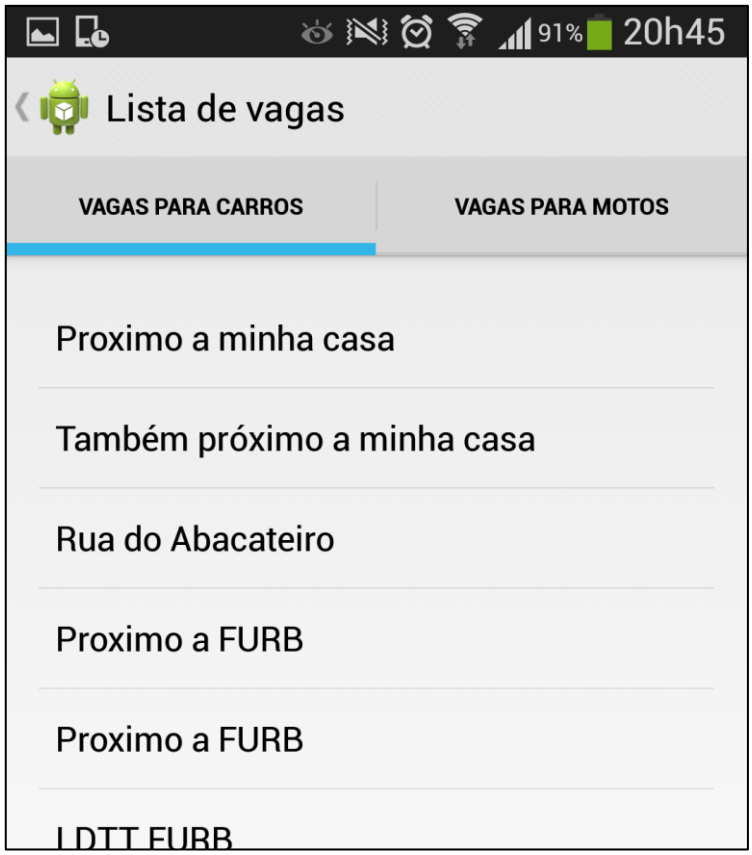
Figura 30 – Chamada da API do PayPal

```
49    /*
50     * Chama activity PagamentoActivity que implementa a API do PayPal
51     */
52    public void compraClick(View v) {
53        Intent intent = new Intent(this, PagamentoActivity.class);
54        startActivity(intent);
55    }
```

Ao acessar o módulo de consulta da lista de ruas com vagas para estacionamento, é exibida esta lista, dividida entre vagas para carro e vagas para moto, conforme pode-se

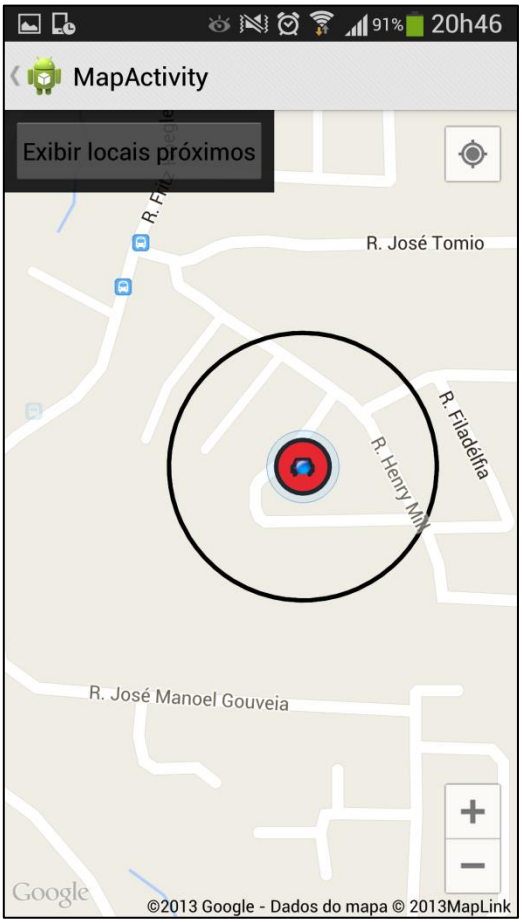
observar na Figura 31. Esta funcionalidade atende ao requisito funcional RF05.

Figura 31 – Lista de vagas disponíveis



Ao acessar o mapa, a posição do usuário é definida com base no melhor provedor de serviço de localização disponível. Este provedor é escolhido automaticamente pela API do Google *Maps*. Ao clicar sobre o ícone no canto superior esquerdo da tela, o usuário é redirecionado para a posição onde encontra-se e são exibidos dois círculos. O primeiro deles possui um raio de 20 metros, preenchido na cor vermelha, e delimita a área na qual o usuário poderá efetuar registros de estacionamento. O segundo círculo possui um raio de 100 metros, sem cor de preenchimento, e exibe, apenas superficialmente, quais outras áreas de estacionamento estão à disposição no referido raio. A Figura 32 exibe o mapa com os locais de estacionamento próximos, representados pelo ícone de um automóvel.

Figura 32 – Mapa com os locais de estacionamento



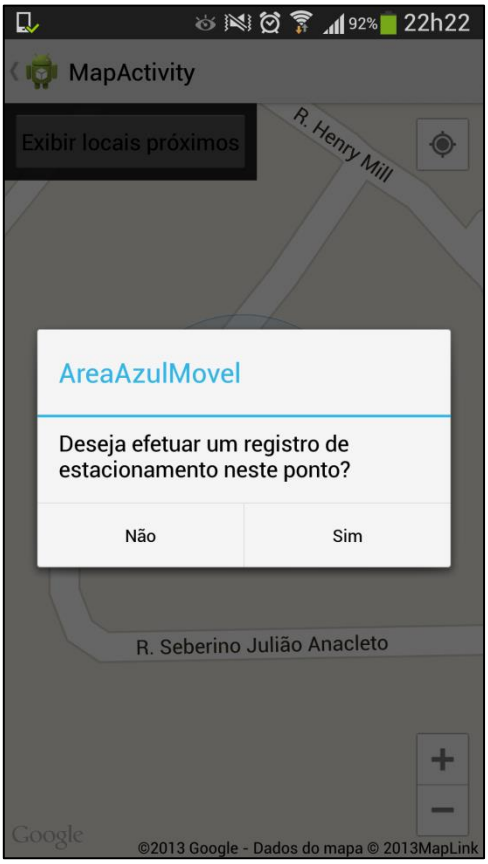
Para acompanhar o deslocamento da aferição da posição do usuário, o mapa é atualizado a cada 5 segundos, o que atende ao requisito funcional RF06. Para isto é criada a variável `REQUEST` na *activity* `MapActivity`, que pode ser vista na Figura 33.

Figura 33 – Definição da variável

```
40    private static final LocationRequest REQUEST = LocationRequest.create()
41            .setInterval(5000) // 5 seconds
42            .setFastestInterval(16) // 16ms = 60fps
43            .setPriority(LocationRequest.PRIORITY_HIGH_ACCURACY);
44
```

Ao clicar sobre qualquer vaga dentro do primeiro raio (de cor vermelha) é aberta uma caixa com informações sobre o local de estacionamento contendo número de identificação do ponto e sua descrição. Ao clicar sobre esta caixa de informações é aberta uma caixa de diálogo conforme Figura 34.

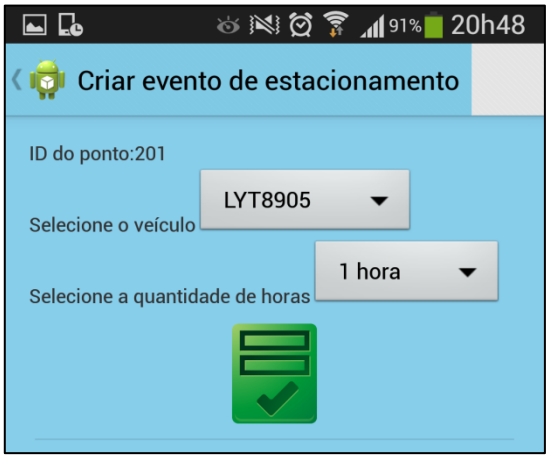
Figura 34 – Diálogo para registro de estacionamento



Ao clicar na opção “Não”, a caixa de diálogo é fechada e o usuário retorna para o mapa. Ao clicar na opção “Sim”, o número de identificação do ponto é armazenado em uma variável do tipo `Bundle`, que pode ser transportada para outra *activity*, o que dispensa a necessidade de armazenar a informação no banco de dados.

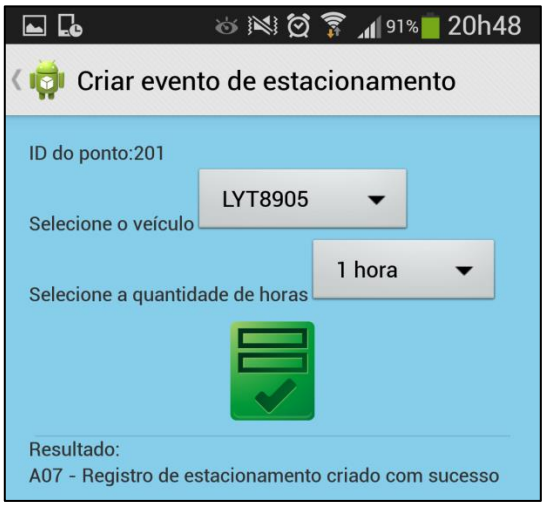
O usuário é redirecionado para a *activity* `CriarEventoEstacionamentoActivity`. Esta *activity* implementa o *layout* `activity_criar_evento_estacionamento.xml` e a interface resultante pode ser observada na Figura 35. O número de identificação do ponto é carregado quando a *activity* `CriarEventoEstacionamentoActivity` é iniciada. O usuário seleciona o veículo que deverá ser utilizado no registro de estacionamento, a quantidade de horas (implicitamente é a quantidade de créditos) que pretende permanecer na vaga e confirma a operação. Esta funcionalidade atende aos requisitos funcionais RF06 e RF12.

Figura 35 – Interface para registro de estacionamento



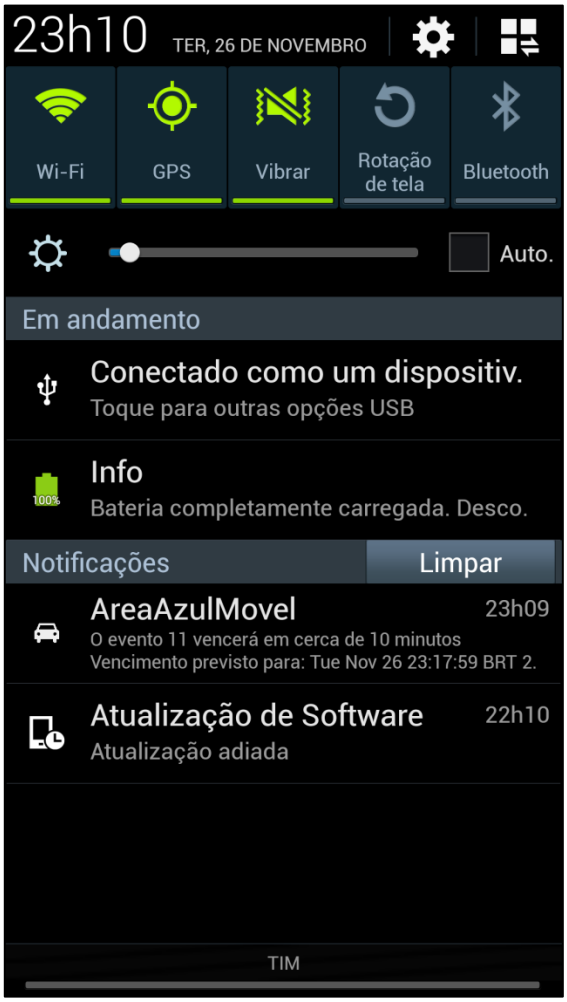
Novamente a classe `WsCallSoap` é chamada e o resultado do processamento é exibido logo abaixo do botão de confirmação, como pode ser visto na Figura 36. Caso o usuário não possua créditos suficientes será exibida mensagem indicando a falta de créditos para concluir a ação. Esta validação atende ao requisito funcional RF08.

Figura 36 – Resultado do registro de estacionamento



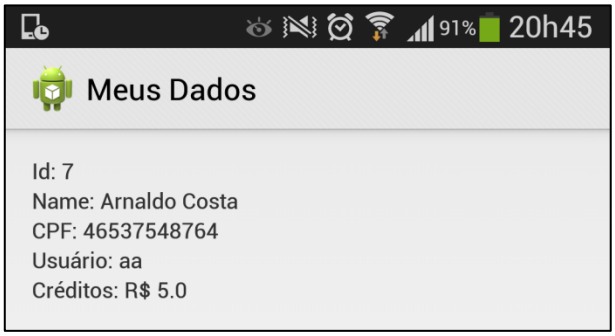
Desde a inicialização do aplicativo um serviço passa a ser executado em segundo plano. O objetivo deste serviço é monitorar os eventos de estacionamento efetuados pelo usuário e manter o mesmo avisado quando o tempo limite de estacionamento estiver próximo de expirar. Para isto, a cada dois minutos efetua-se uma busca pelos eventos de estacionamento pendentes. Sempre que é encontrado algum registro, verifica-se quantos minutos faltam para que o mesmo expire. A primeira notificação será emitida quando restarem cerca de dez minutos, conforme Figura 37. A notificação seguinte será emitida quando restarem cerca de cinco minutos. E a última notificação será exibida quando o registro de estacionamento estiver vencido. Esta rotina atende ao requisito funcional RF15.

Figura 37 – Notificação de vencimento de registro de estacionamento



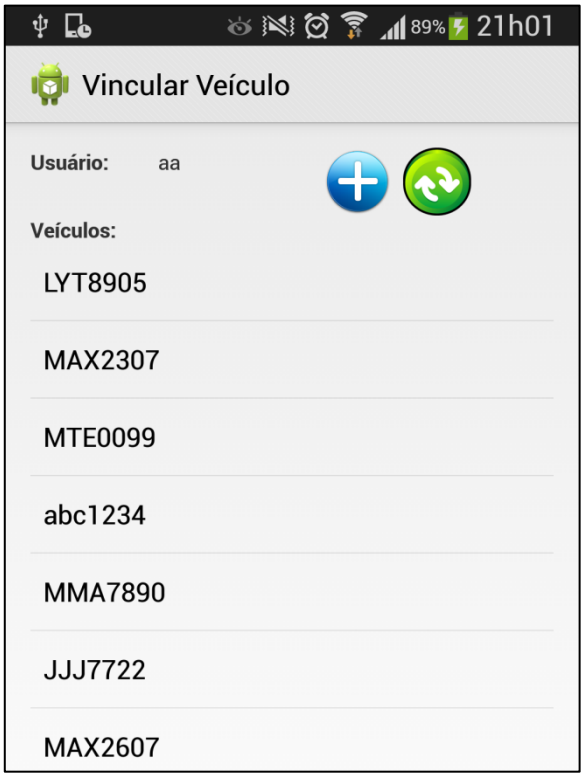
De volta ao menu principal, ao acessar o módulo de consulta de dados, os dados do usuário autenticado são exibidos, conforme exposto na Figura 38.

Figura 38 – Dados do usuário autenticado



Ao acessar o módulo de veículos o usuário pode consultar os veículos associados ao seu perfil e adicionar novos veículos. Esta funcionalidade atende ao requisito funcional RF03. Os dados sobre um veículo estão limitados à placa, quando consultados por um condutor, conforme exibido na Figura 39.

Figura 39 – Tela de consulta e vinculação de veículos



É disponibilizada ainda a consulta de notificações, o que atende ao requisito funcional RF09. O usuário seleciona a placa do veículo que deseja consultar e confirma a operação. A classe `WsCallSoap` chama o método `consultaNotificacoes`() do *web service* `EventoService`. Este método é responsável por verificar a ligação entre a pessoa e o veículo que está sendo informado e verificar se já existe algum registro de estacionamento efetuado entre os mesmos. Caso não exista nenhum registro de estacionamento efetuado, significa que possivelmente alguém está tentando consultar as notificações pendentes para um veículo "qualquer", que não necessariamente faça parte dos veículos utilizados por esta pessoa no registro de estacionamentos em Área Azul. Na Figura 40 é apresentado o método `consultaNotificacoes`() do w*eb service* `EventoService`.

Figura 40 – Método responsável por retornar as notificações emitidas

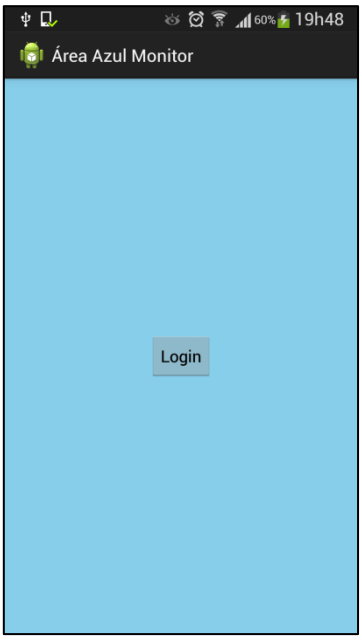
```
727     @WebMethod
728     public List<Notificacao> consultaNotificacoes(
729             @WebParam(name = "codPessoa") long codPessoa,
730             @WebParam(name = "placa") String placa) {
731         CondutorVeiculoDAOImp condutorImp = new CondutorVeiculoDAOImp();
732         CondutorVeiculo condutor = condutorImp
733                 .getCondutorVeiculoByPessoaEPlaca(codPessoa, placa);
734         // verifica se existe associação entre o solicitante e a placa do
735         // veículo;
736         if (condutor == null) {
737             // significa que não há ligação entre a pessoa e o veículo
738             return null;
739         }
740         EventoDAOImp eventoImp = new EventoDAOImp();
741         List<Evento> eventos = eventoImp.getEventosByCondutor(condutor);
742         if (eventos.size() == 0) {
743             // significa que não existe registros feitos entre o condutor e o
744             // veiculo. Para evitar consultas de notificações com placas feitas
745             // por qualquer pessoa
746             return null;
747         } else {
748             NotificacaoDAOImp notificacaoImp = new NotificacaoDAOImp();
749             List<Notificacao> notificacoes = notificacaoImp
750                     .getNotificacoesPendentesByVeiculo(placa);
751             // limpa os detalhes de origem para retornar apenas a o ID e a
752             // situação da notificação pendente
753             if (notificacoes == null) {
754                 return null;
755             } else {
756                 for (int i = 0; i < notificacoes.size(); i++) {
757                     notificacoes.get(i).setDetalheOrigem(null);
758                 }
759                 return notificacoes;
760             }
761         }
762     }
763 }
```

Caso haja notificações pendentes de pagamento no sistema elas serão listadas, conforme exibido na Figura 41. Caso não haja notificações pendentes, ou o usuário não tenha efetuado nenhum registro de estacionamento (com o veículo informado) será exibida a mensagem “Não há notificações pendentes para este veículo ou você não possui permissão para visualizá-las. Para ter acesso às notificações emitidas para este veículo, certifique-se de que você tenha efetuado ao menos 1 registro de estacionamento utilizando o mesmo”.

Figura 41 – Consulta de notificações

Sob a perspectiva do monitor, este efetuará o acesso ao aplicativo através do respectivo ícone em seu *smartphone* ou *tablet*. A interface inicial do sistema permite ao usuário efetuar o *login*. O cadastro de novos monitores só poderá ser realizado através da aplicação disponível para operadores, a fim de garantir que, caso o aplicativo de monitores seja acessado por uma pessoa não autorizada, esta não possa criar um cadastro sem autorização. Na Figura 42 pode-se observar a interface inicial do aplicativo AreaAzulMovelMonitor.

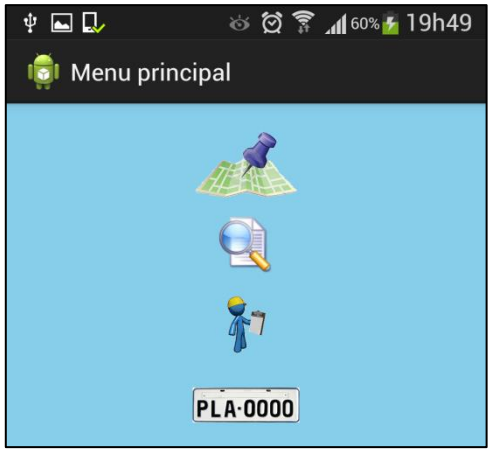
Figura 42 – Tela inicial do aplicativo AreaAzulMovelMonitor



Durante o *login* no aplicativo é executada em segundo plano a tarefa de atualização da

lista de ruas com vagas disponíveis. Assim como no aplicativo AreaAzulMovel, no aplicativo AreaAzulMovelMonitor a classe responsável por efetuar os procedimentos em segundo plano é a classe `WsCallSoap`, que estende a classe `AsyncTask` e implementa seu ciclo de vida. Após o *login* no aplicativo, o usuário é redirecionado menu do sistema, exibido na Figura 43.

Figura 43 – Menu principal do aplicativo AreaAzulMovelMonitor



A interface do menu principal foi definida no arquivo `activity_menu.xml`. Através do menu principal o usuário poderá acessar uma lista das ruas com vagas da Área Azul, consultar registros de estacionamentos efetuados pelos condutores, emitir notificações e consultar informações completas sobre um veículo. A Figura 44 exibe a tela de consulta de locais de estacionamento, listando o número de identificação e descrição das vagas ativas.

Figura 44 – Consulta de locais de estacionamento com Área Azul

Ao acessar a tela de consulta de registros, conforme exibido na Figura 45, o usuário

informará a placa do veículo e o número de identificação do ponto que deseja consultar e clicará sobre o ícone da lupa, efetuando assim a consulta dos registros de estacionamento junto ao servidor. É de responsabilidade do monitor utilizar o número de identificação correto do local, apresentado na consulta de locais. O resultado é apresentado abaixo do botão de consulta, com o resultado desta.

Figura 45 – Tela de consulta de registros de estacionamento

Ao acessar a opção de emissão de notificação, conforme exibido na Figura 46, o usuário informará a placa do veículo para o qual deseja emitir a notificação e o número de identificação do ponto no qual encontra-se e clicará sobre o botão para confirmar a operação. O resultado do processamento será exibido abaixo do botão pressionado. Esta funcionalidade atende aos requisitos funcionais RF10 e RF11.

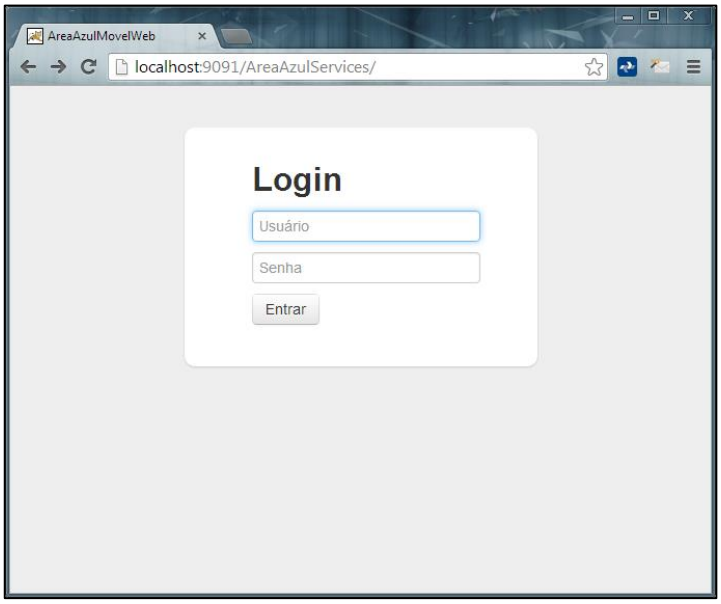
Figura 46 – Emissão de notificação através do aplicativo AreaAzulMovelMonitor

A última opção disponível para o monitor é a consulta de informações detalhadas sobre os veículos. As informações serão retornadas conforme os dados cadastrados no servidor. Para isto, o monitor acessará o módulo de consulta de informações sobre veículos, informará a placa do veículo e pressionará o botão de lupa. O resultado do processamento será exibido abaixo do botão de consulta, conforme Figura 47. Esta funcionalidade atende ao requisito funcional RF16.

Figura 47 – Consulta de informações sobre veículos

Sob a perspectiva do operador, este acessará o sistema *web* através do navegador de internet. Deverá preencher os dados de usuário e senha e confirmar a operação, exibidos na Figura 48. Será redirecionado ao menu principal do sistema, onde poderá criar novos cadastros, efetuar a manutenção dos pontos de estacionamento e dar baixa nas notificações emitidas.

Figura 48 – Interface de *login* do AreaAzulMovelWeb

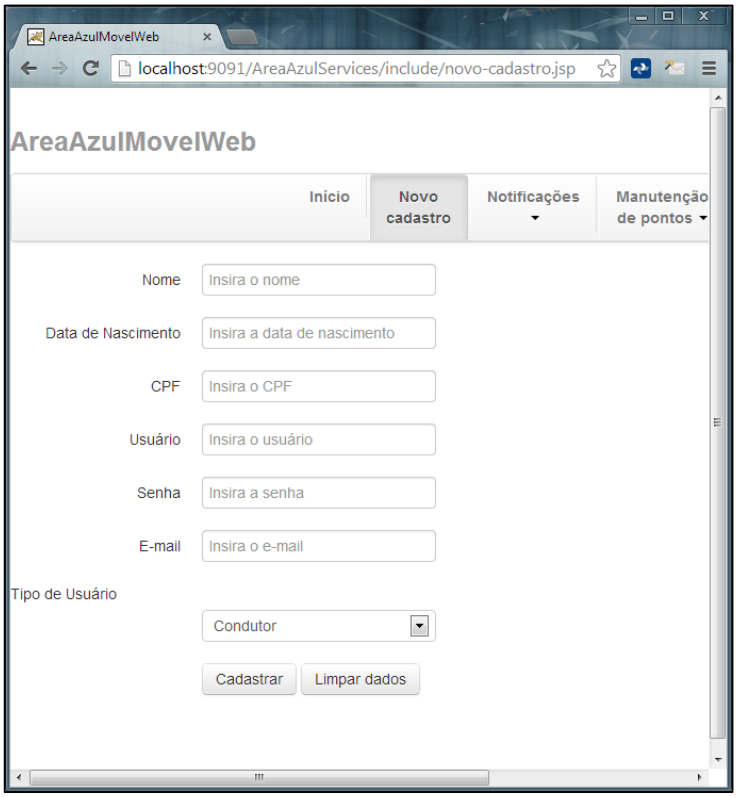


A página de início exibe uma mensagem de boas vindas e informações básicas sobre as funcionalidades da página, conforme apresentado na Figura 49.

Figura 49 – Página início do sistema AreaAzulMovelWeb

Ao clicar na opção `Novo cadastro` o usuário será redirecionado para a página na qual ele poderá efetuar o cadastro de um novo usuário. Apenas através do AreaAzulMovelWeb é que novos usuários Operadores e Monitores poderão ser cadastrados no sistema. A tela de cadastro é exibida na Figura 50.

Figura 50 – Novo cadastro



Ao clicar no menu `Notificações`, o usuário poderá optar por consultar ou dar baixa nas notificações. A Figura 51 exibe a tela de consulta de notificações. O operador deverá informar a placa do veículo e o ID do condutor (informado pelo próprio condutor ao se dirigir para o balcão a fim de consultar ou regularizar as notificações) e confirmar a operação.

Figura 51 – Consulta de notificações

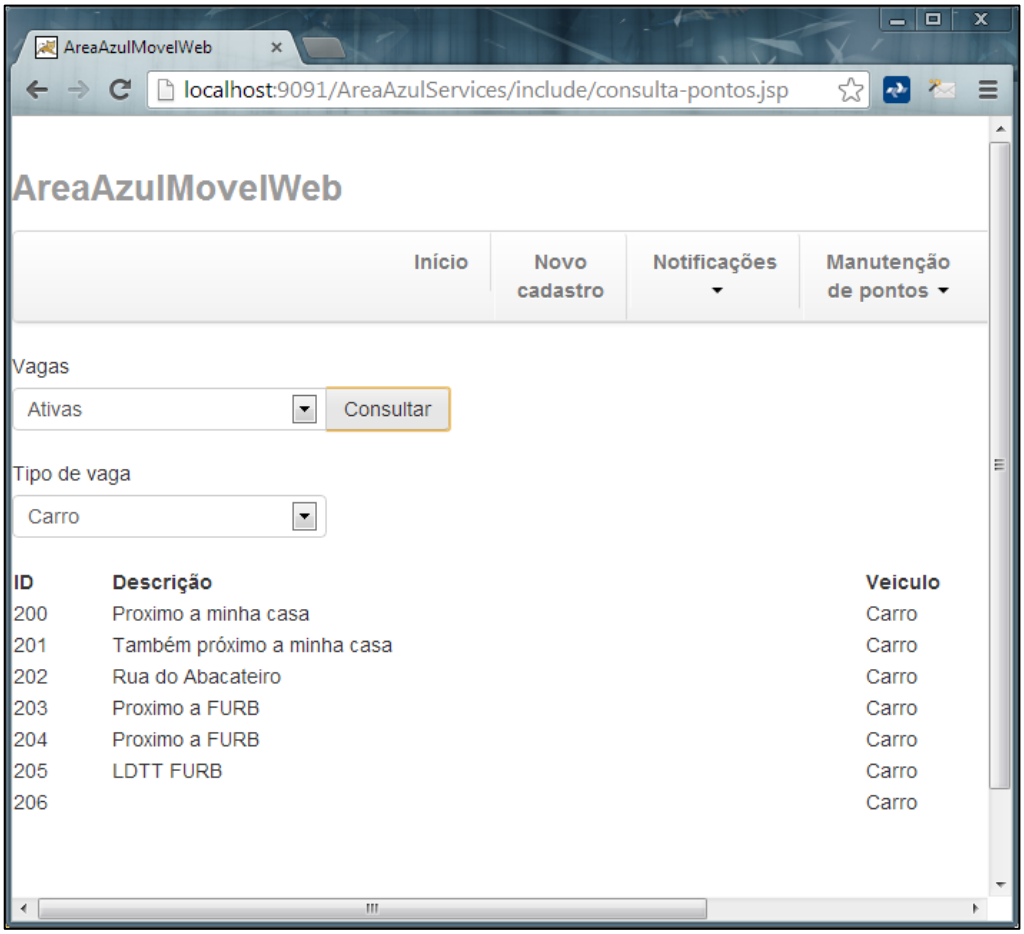
Ao selecionar a opção `Dar baixa`, o operador é direcionado para a página na qual poderá dar baixa nas notificações emitidas. Para isto, o mesmo deverá digitar o ID da

notificação e clicar na opção “Dar baixa”, apresentada na Figura 52.

Figura 52 – Dar baixa em notificação

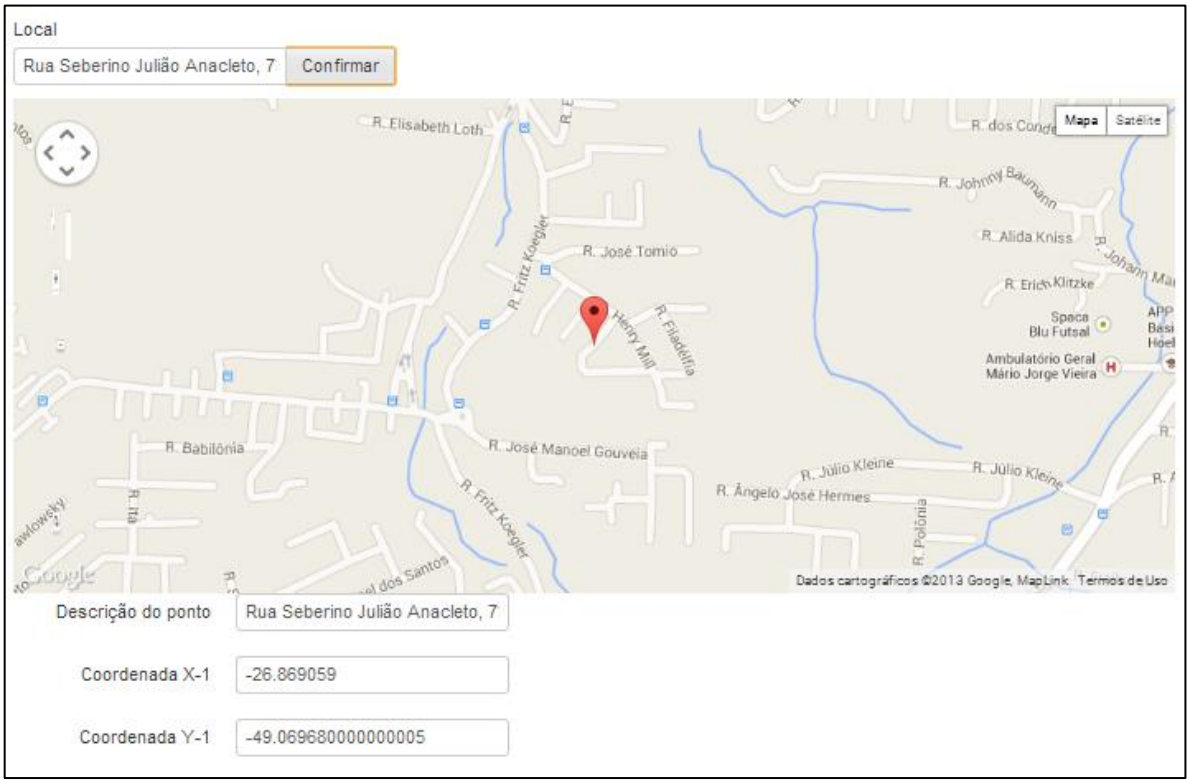
Ao clicar na opção `Manutenção de pontos` o usuário poderá optar por consultar, inserir, ativar ou inativar um ponto de estacionamento. Estas funcionalidades atendem ao requisito funcional RF14. O usuário seleciona a opção `Consultar` e é direcionado à página exibida na Figura 53. Ele poderá optar por visualizar as vagas inativas (que poderão ser ativadas) ou ativas (que podem ser utilizadas para efetuar o registro de estacionamento). Entre as vagas ativas, ele deverá optar pelo tipo de vaga de acordo com o veículo (carro ou moto). Para confirmar a consulta ele clicará sobre o botão `Consultar`.

Figura 53 – Consulta de vagas



Ao selecionar a opção “Inserir”, o usuário será redirecionado à página na qual efetuará o cadastro de um novo ponto de estacionamento. Para um melhor entendimento, esta tela deve ser dividida (logicamente e não fisicamente) em duas partes: superior e inferior. A parte superior, exibida na Figura 54, contém um mapa do Google Maps, com um marcador na cor vermelha. O mapa inicial traz a localidade da Prefeitura Municipal de Blumenau (PMB) como centro de referência. Ainda sim, acima do mapa há uma barra de busca por endereços e um respectivo botão. A parte inferior contém um formulário com os dados a serem preenchidos para o cadastro de um novo ponto de estacionamento.

Figura 54 – Mapa com marcador para inclusão de ponto de estacionamento



Para criar um novo ponto de estacionamento o usuário deverá preencher a descrição do ponto e, no mínimo, o primeiro par de coordenadas XY (latitude e longitude). Há duas formas de se obter as coordenadas XY:

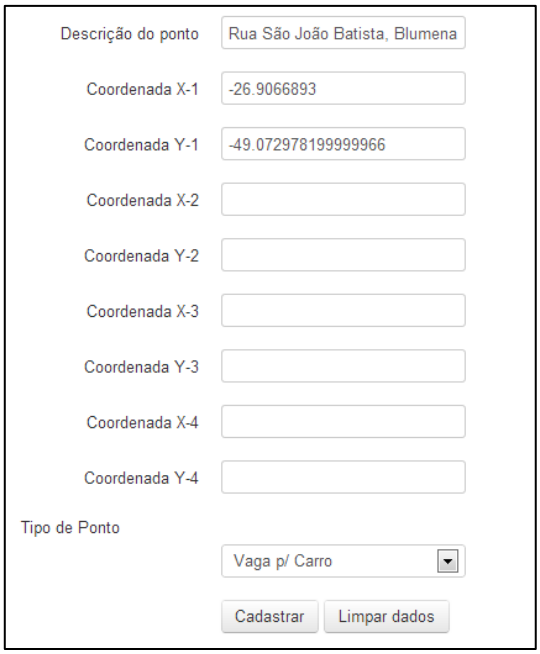
a) arrastar o marcador: o usuário arrasta o marcador diretamente no mapa, deslocando-o até o lugar desejado. Desta forma, o primeiro par de coordenadas será atualizado dinamicamente nos campos `Coordenada X-1` e `Coordenada Y-1`, com a latitude e longitude, respectivamente, porém o campo `Descrição do ponto` manterá o valor já contido no mesmo, ainda que nulo;

b) buscar através do endereço: o usuário digita um endereço de busca e clica no botão `Confirmar`. O marcador é direcionado para o resultado da busca, o campo `Descrição do Ponto` no formulário é preenchido com o endereço utilizado como parâmetro, e os campos `Coordenada X-1` e `Coordenada Y-1` são preenchidos com a latitude e longitude, respectivamente.

Na parte inferior da tela encontra-se o formulário para inserção de pontos. Nele, o usuário informará os campos necessários para a criação de um novo ponto de estacionamento. Caso uma vaga de estacionamento estenda-se por um raio superior a 20 metros, o operador pode utilizar os demais campos de coordenadas para preencher conjuntos de coordenadas próximas, aumentando assim a abrangência de um local de estacionamento. Estes campos

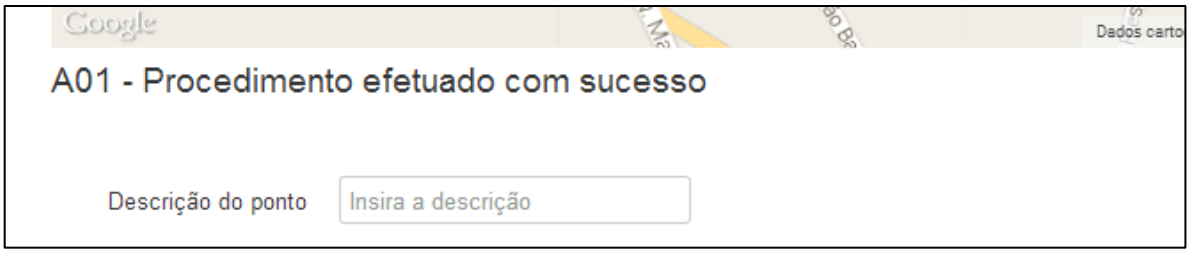
adicionais não são preenchidos automaticamente. Sugere-se que o monitor mova o marcador do mapa para obter todos os pontos necessários e copie manualmente os valores dos campos `Coordenada X-1` e `Coordenada Y-1` para compor os demais campos. Esta opção evita a existência de um número muito grande de locais de estacionamento em uma determinada área. Cada par de coordenadas representa um eixo central, utilizado como referência para cálculo da proximidade do condutor no ato do registro de estacionamento. A Figura 55 apresenta o respectivo formulário. Após o usuário preenchê-lo, ele deverá clicar no botão `Confirmar`, para enviar os dados e cadastrar o ponto, ou no botão `Limpar dados` para apagar o conteúdo dos campos da tela.

Figura 55 – Formulário para inserção de ponto



Após a confirmação do cadastro, o resultado do processamento será exibido em uma mensagem entre o Mapa e o formulário. Esta mensagem indicará sucesso ou erro ao realizar cadastro, bem como validações sobre os campos mínimos obrigatórios. Um exemplo de mensagem de retorno é exibido na Figura 56.

Figura 56 – Resultado do processamento



A última funcionalidade disponível é a ativação ou inativação de pontos. O operador

inativará um ponto quando, fisicamente, o ponto deixar de existir ou for substituído por outro ponto. Sempre que um ponto é inativado não são permitidos novos registros de estacionamento naquela localidade. Sempre que um ponto é ativado ou cadastrado, o mesmo passará a compor a lista de vagas disponíveis para uso dos condutores. Para ativar ou inativar um ponto o operador deverá informar o ID do ponto, obtido através da consulta, selecionar a opção `ativar` ou `inativar` e confirmar a operação. A Figura 57 apresenta a tela de ativação e inativação de pontos.

Figura 57 – Tela de ativação ou inativação de pontos

3.4 RESULTADOS E DISCUSSÃO

O protótipo desenvolvido atendeu às expectativas e requisitos propostos, propiciando flexibilidade para condutores utilizarem a Área Azul através de seus *smartphones* ou *tablets*. A utilização de um mapa do Google Maps ajudou no que diz respeito à interface de usuário, pois exibe os pontos de estacionamento próximos ao condutor e permite que este saiba exatamente onde está depositando seu crédito de estacionamento. O uso da API do PayPal também teve efeito similar, pois permite a aquisição dos créditos com base em uma interface bem construída e operações seguras.

Muitos obstáculos foram encontrados, mas nem todos totalmente transpostos. Um exemplo claro e que pode ser citado, foi a necessidade de alteração do *framework* para a parte *web*. O desenvolvimento iniciou-se com o uso do *framework* Sencha Touch que utiliza linguagem de desenvolvimento EXT JS, mas ao longo do projeto mostrou-se complexo e não totalmente integrado com a tecnologia DWR. Com isto, optou-se pela tecnologia JSP para esta parte do protótipo, que utiliza a linguagem HTML e Javascript (linguagens com relativa facilidade de compreensão) mescladas com código Java, que permite ao desenvolvedor utilizar estruturas já conhecidas (laços de repetição, estruturas condicionais entre outros) na

construção das páginas.

Dos trabalhos correlatos apresentados na seção 2.10, apenas o sistema Vaga AZUL, da empresa Declink, e o Estacionamento Rotativo, da BHTRANS, estão inseridos no mesmo contexto que o protótipo desenvolvido neste trabalho. No Quadro 3 apresenta-se o quadro comparativo entre os sistemas. Com relação às funcionalidades descritas a respeito do sistema Vaga AZUL e Estacionamento Rotativo, algumas são desconhecidas, uma vez que detalhes técnicos não são mencionados nas páginas consultadas.

Quadro 3 – Comparativo entre funcionamento dos sistemas

Sistema / **Funções**	**AreaAzulMovel**	**Vaga AZUL**
1. Uso de plataforma móvel?	SIM	SIM
Através do dispositivo móvel:		
1.1 Permite o registro de estacionamento?	SIM	NÃO
1.2 Permite a consulta de registro de estacionamento?	SIM	SIM
1.3 Permite a emissão de notificação?	SIM	NÃO
1.4 Permite a aquisição de créditos?	SIM	SIM
1.5 Controla as vagas de estacionamento?	SIM	SIM
1.6 Permite a consulta dos dados de um veículo?	SIM	NÃO
1.7 Permite a consulta dos locais de estacionamento?	SIM	NÃO
Fora da plataforma móvel:		
2. Controla as vagas de estacionamento?	NÃO	SIM
3. Pode trabalhar paralelamente a outros sistemas?	PARCIAL (talões)	SIM

4 CONCLUSÕES

Ao término do desenvolvimento todos os objetivos propostos do trabalho foram alcançados. Isto significa que o protótipo desenvolvido atende às necessidades básicas da forma de trabalho e atuação da Área Azul no município de Blumenau.

O grande diferencial do trabalho em questão é o uso da plataforma móvel Android. Desta forma, a sociedade em geral ganha, pois tem a possibilidade de utilizar seu dispositivo móvel "sem sair do lugar" para efetuar uma tarefa que antes demandava conhecimento, para saber onde adquirir os talões da Área Azul, e tempo, para enfrentar o caixa, muitas vezes com fila.

Quem ganha também, a longo prazo, é o meio-ambiente, pois o uso de papel (dos talões de estacionamento) torna-se dispensável. Com isto, o usuário da Área Azul pode efetuar todos os procedimentos necessários para registro de estacionamento, através de seu aparelho, sem a necessidade de deslocar-se a outros pontos e correr o risco de ser notificado.

Os monitores disfrutam de flexibilidade semelhante, já que podem consultar os registros de estacionamento através de seu aparelho e também emitir as notificações.

Os operadores podem efetuar a manutenção dos pontos de estacionamento – inclusão, ativação, inativação e consulta – através de uma interface simples e objetiva. Para facilitar (e garantir) o local onde um ponto está sendo inserido, o operador pode utilizar o mapa embarcado na parte *web* para levantar as coordenadas geográficas exatas.

O desenvolvimento de um protótipo de sistema de informação que atue em uma área pública (estacionamento em locais públicos, por exemplo) exige uma profunda avaliação das funcionalidades e flexibilidades que deve-se alcançar.

Saber integrar as ferramentas e tecnologias para proporcionar uma boa usabilidade é fruto de (muitas) horas de análise e dezenas de rascunhos com o desenho do processo. A curva de aprendizado obtida durante as mais de 400 horas de trabalho mostrou-se bastante elevada e o autor precisou mudar seu paradigma de desenvolvimento para enquadrar-se nas diferentes plataformas envolvidas (servidor, cliente móvel, cliente *web)* inúmeras vezes.

A falta de experiência com o desenvolvimento em um contexto geral dificultou e comprometeu a produtividade da etapa de codificação e exigiu mudanças de última hora que afetaram outras partes do projeto, como a realização de testes e a edição da monografia.

A cada dia novos obstáculos apareceram e mostraram-se reveladores, pois sempre que transpostos aumentaram a força de vontade para conclusão do trabalho. Quando não puderam

ser transpostos e nem mesmo contornados, exigiram a reanálise e até reestruturação da forma construtiva, exigindo nova pesquisa e aprendizado de tecnologias.

Por fim, o desenvolvimento deste trabalhou proporcionou ao autor exponencial crescimento no conhecimento de tecnologias disponíveis no mercado e a correta aplicação das mesmas, e também aguçou o senso crítico sobre os requisitos que um negócio precisa atingir, antes de se discutir plataformas e tecnologias.

4.1 EXTENSÕES

A fim de aprimorar e melhorar a experiência do usuário, sugere-se a remodelagem das telas dos aplicativos móveis, de modo a implementar recursos gráficos mais sofisticados, disponíveis em versões mais recentes da plataforma Android.

No aplicativo AreaAzulMovel, poderia ser implementado o tratamento para tolerâncias de estacionamento (estacionamentos de no máximo 15 minutos), desta forma o condutor não precisaria gastar um crédito de estacionamento para utilizar a vaga.

A utilização da tecnologia *Optical Character Recognition* (OCR) no protótipo AreaAzulMovelMonitor, para as operações que envolvam a digitação da placa do veículo diminuiria a incidência de erros de digitação deste tipo de dado. Adicionalmente, a foto capturada poderia ser enviada com estampa de data, horário e posição geográfica, para o servidor, e anexada ao momento da consulta ou notificação, como prova conclusiva em possíveis contestações efetuadas pelos donos dos veículos.

De forma semelhante, a tecnologia *Near Field Communication* (NFC) poderia ser aplicada. No veículo existiria um *chip* suportado por esta tecnologia com uma respectiva área destinada à identificação. Com a aproximação do dispositivo móvel do monitor – que também deve suportar esta tecnologia – o veículo seria identificado, permitindo assim a verificação do registro de estacionamento. Para isto, o cadastro do veículo no sistema precisaria ser redefinido, de forma que suporte os dados adicionais de identificação do veículo.

Outra sugestão é implementar códigos do tipo *Quick Response Code* (QR *Code*) nas placas sinalizadoras de Área Azul e tratar o reconhecimento das mesmas no protótipo AreaAzulMovel, permitindo assim que o usuário efetue o registro de estacionamento no local desejado quando sua localização não condizer precisamente com o local onde seu carro está estacionado.

Poder-se-ia também implementar um novo módulo do sistema para auditar a baixa das notificações. Desta forma, seria possível rastrear quem efetuou a baixa de uma notificação e

quando. Adicionalmente, o envio de *e-mail* para o dono do veículo ou condutor poderia ser implementado, em ações como a verificação de um veículo, emissão de notificação ou baixa de notificação.

Sugere-se, ainda, o desenvolvimento dos aplicativos móveis para a plataforma iOS, utilizada em *smartphones* e *tablets* da Apple, como o iPhone 5 e o iPad 2.

REFERÊNCIAS

ANDROID DEVELOPERS. **Android, the world's most popular mobile platform**. Mountain View, 2013a. Disponível em: <http://developer.android.com/about/index.html>. Acesso em: 24 maio 2013.

_____. Develop. **Get the Android SDK**. Mountain View, 2013b. Disponível em: <http://developer.android.com/sdk/index.html>. Acesso em: 26 maio 2013.

_____. **Activity**. Mountain View, 2013c. Disponível em: < http://developer.android.com/reference/android/app/Activity.html>. Acesso em: 10 nov. 2013.

BARROS, Thiago. TechTudo. PayPal. **Baixe PayPal, faça pagamentos e transferências com segurança**. São Paulo, 2013. Disponível em: <http://www.techtudo.com.br/tudo-sobre/s/paypal.html>. Acesso em: 5 nov. 2013

BHTRANS. Empresa de Transportes e Trânsito de Belo Horizonte. **Estacionamento Rotativo:** o que é Rotativo. Belo Horizonte, [2013?]. Disponível em: <http://www.bhtrans.pbh.gov.br/portal/page/portal/portalpublico/Tr%C3%A2nsito/rotativo>. Acesso em: 10 abr. 2013.

BOOTSTRAP. Sleek, intuitive, and powerful mobile first fron-end framework for faster and easier web development. **Bootstrap**. San Francisco, 2013. Disponível em: <http://getbootstrap.com/>. Acesso em: 16 nov. 2013.

DECICINO, Ronaldo. **GPS – Sistema de posicionamento global tem diferentes utilidades.** São Paulo, 2009. Disponível em: <http://educacao.uol.com.br/disciplinas/geografia/gps-sistema-de-posicionamento-global-tem-diferentes-utilidades.htm>. Acesso em: 27 maio 2013.

DECLINK. Produto. **Vaga AZUL – Controle de Estacionamento em Logradouro Público**. Rio de Janeiro, [2013?]. Disponível em: <http://site.declink.com.br/celp>. Acesso em: 30 mar. 2013.

DEMARCHI, Felipe. **Migração do sistema Aruana-maleta para a plataforma Android**. 2012. 67 f, il. Trabalho de Conclusão de Curso (Graduação em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau. Blumenau, 2012. Disponível em: <http://www.bc.furb.br/docs/MO/2012/352402_1_1.PDF>. Acesso em: 30 abr. 2013.

DEPARTAMENTO ESTADUAL DE TRÂNSITO. Detran/SC. Estatística. Veículo. **Frota de veículos por município**. Florianópolis, 2013. Disponível em: < <http://consultas.detrannet.sc.gov.br/Estatistica/Veiculos/winVeiculos.asp?lst_municipio=8047&nome_munic=BLUMENAU&lst_ano=2013&lst_mes=0>. Acesso em: 2 nov. 2013.

DRAGO, Daniele; DISPERATI, Attilio A. **Aspectos básicos sobre GPS**. Curitiba: FUPEF, 1996. p. 1-2.

DWR. **DWR**: easy Ajax for Java. Market Harborough, 2013. Disponível em: <http://directwebremoting.org/dwr/introduction/index.html>. Acesso em: 28 oct. 2013.

EMPRESA BRASILEIRA DE CORREIOS E TELÉGRAFOS. Para Você. **Calculador remoto de Preços e Prazos**. Brasília, [2013?]. Disponível em: <http://www.correios.com.br/webservices/>. Acesso em: 6 nov. 2013

FRANCISCO, Wagner de Cerqueira e. **GPS – Sistema de Posicionamento Global**. Aparecida de Goiânia, [2013?]. Disponível em: < http://www.brasilescola.com/geografia/gpssistema-posicionamento-global.htm>. Acesso em: 27 maio 2013.

FREITAS, Eduardo de. **Coordenadas Geográficas.** Aparecida de Goiânia, [2008?]. Disponível em: <http://www.brasilescola.com/geografia/sig.htm>. Acesso em 11 abr. 2013.

GOOGLE DEVELOPERS. Aplicativos móveis. **Criar mapas para aplicativos móveis**. Mountain View, 2013. Disponível em: <https://developers.google.com/maps/mobile-apps?hl=pt-br>. Acesso em: 26 maio 2013.

GOOGLE MAPS. **Bem vindo ao Google Maps**. Mountain View, 2013. Disponível em: <http://support.google.com/maps/bin/answer.py?hl=pt&answer=144352&topic=1687350&ctx=topic>. Acesso em: 10 abr. 2013.

IBM. DeveloperWorks. **Ajax for Java developers:** Ajax with Direct Web Remoting. Foster City, 2005. Disponível em: <http://www.ibm.com/developerworks/library/j-ajax3/>. Acesso em: 3 nov. 2013

ORACLE. Oracle Technology Network. Java. Java EE. **JavaServer Pages Techonology**. Redwood City, [2013?]. Disponível em: <http://www.oracle.com/technetwork/java/javaee/jsp/index.html>. Acesso em: 16 nov. 2013.

PAMPLONA, Vitor Fernando. Web Services. **Construindo, disponibilizando e acessando Web Services via J2SE e J2ME.** São Paulo, 2010. Disponível em: <http://javafree.uol.com.br/artigo/871485/Web-Services-Construindo-disponibilizando-e-acessando-Web-Services-via-J2SE-e-J2ME.html>. Acesso em: 3 nov. 2013.

PAYPAL. Conheça PayPal. **Com PayPal, suas compras ficam bem mais simples e seguras**. San Jose, 2013a. Disponível em: < https://www.paypal.com/br/webapps/mpp/conheca-paypal>. Acesso em: 5 nov. 2013

_____. PayPal Developer. **PayPal Mobiles SDKs**. San Jose, 2013b. Disponível em: <https://developer.paypal.com/webapps/developer/docs/integration/mobile/mobile-sdk-overview/>. Acesso em: 7 nov. 2013

PENA, Rodolfo Alves. **SIG.** Aparecida de Goiânia, [2013?]. Disponível em: <http://www.brasilescola.com/geografia/sig.htm>. Acesso em: 11 abr. 2013.

SENA, Cicero. Aplicativos para dispositivos móveis agilizam o mercado. **Nuvendigital**. Salvador, 2012. Disponível em: <http://nuvendigital.com/aplicativos-para-dispositivos-moveis-agilizam-o-cotidiano/>. Acesso em: 05 abr. 05 2013.

SERVIÇO AUTÔNOMO MUNICIPAL DE TRÂNSITO E TRANSPORTES DE BLUMENAU. Seterb. Prefeitura Municipal de Blumenau. Serviços e Informações. Trânsito. **Área Azul**. Blumenau, 2012. Disponível em: <http://www.blumenau.sc.gov.br/gxpsites/hgxpp001.aspx?1,28,358,O,P,0,MNU;E;98;6;153;2;MNU;,;>. Acesso em: 26 mar. 2013.

_____. Seterb. Prefeitura Municipal de Blumenau. Serviços e Informações. Trânsito. **Área Azul**. Blumenau, 2013. Disponível em: <http://www.blumenau.sc.gov.br/gxpsites/hgxpp001.aspx?1,28,358,O,P,0,MNU;E;98;6;153;2;MNU;,;>. Acesso em: 21 maio 2013.

SILVA, Luciano Alves da. Apostila de Android. **Programando passo a passo: 4ª edição**. Rio de Janeiro, 2011. Disponível em: < https://sites.google.com/site/apostilaandroid/apostila-android-4-edicao.zip?attredirects=0>. Acesso em: 03 abr. 2013.

TANJI, Thiago. Info Abril. Notícias. Mercado. **Android tem crescimento de 79,5% nas vendas**. São Paulo, 2013. Disponível em: <http://info.abril.com.br/noticias/mercado/android-tem-crescimento-de-79-5-nas-vendas-16052013-42.shl>. Acesso em: 25 maio 2013.

TOMTOM. A história do GPS. **Como funciona o GPS?.** Países Baixos, [2013?]a. Disponível em: <http://www.tomtom.com/howdoesitwork/page.php?ID=6&CID=2&Language=17>. Acesso em: 26 maio 2013.

_____. Quem utiliza o GPS. **Como funciona o GPS?.** Países Baixos, [2013?]b. Disponível em: <http://www.tomtom.com/howdoesitwork/page.php?ID=7&CID=2&Language=17>. Acesso em: 26 maio 2013.

W3SCHOOLS. **SOAP Introduction**. Cambridge, 2013. Disponível em: <http://www.w3schools.com/webservices/ws_soap_intro.asp>. Acesso em: 3 nov. 2013.

WORLD WIDE WEB CONSORTIUM. **Web Services Activity Statement.** Cambridge, 2012. Disponível em: <http://www.w3.org/2002/ws/Activity>. Acesso em: 3 nov. 2013.

APÊNDICE A – Descrição dos Casos de Uso

Este Apêndice apresenta, através do Quadro 4, a descrição dos principais casos de uso descritos na seção de especificação deste trabalho.

Quadro 4 – Descrição dos Casos de Uso

UC01 Cadastrar Usuário no Aplicativo
Permite ao usuário realizar o cadastro no aplicativo após o *download* do mesmo, através de seu dispositivo móvel.

Constraints
Pré-condição. O usuário/ condutor deve possuir acesso à Internet.

Pré-condição. O usuário/ condutor deve possuir o aplicativo instalado e funcional em seu dispositivo móvel.

Pós-condição. O usuário/ condutor deverá estar devidamente cadastrado no sistema.

Cenários
Consulta serviço {Principal}.
1. O usuário acessa o aplicativo em seu dispositivo móvel;

2. O sistema solicita a inserção dos dados de *login* no sistema;

3. O usuário que ainda não possui cadastro clica sobre a opção de novo cadastro;

4. O usuário utiliza seus dados de *login* para realizar o acesso ao sistema.

Usuário já cadastrado {Exceção}
No passo 2 do cenário principal, o usuário já possui cadastro no sistema. Ir direto para o passo 4 do cenário principal.

UC06 Registrar Estacionamento
Permite o usuário efetuar o registro de estacionamento em local de estacionamento regulamentado: Área Azul. São realizadas as validações quanto à localização, veículo utilizado e créditos disponíveis.

Constraints
Pré-condição. O usuário/ condutor deve possuir acesso à internet.

Pré-condição. O usuário/ condutor deve possuir o aplicativo instalado e funcional em seu dispositivo móvel.

Pré-condição. O usuário/ condutor deve possuir cadastro no sistema.

Pós-condição. O usuário/ condutor conseguiu registrar seu estacionamento em vaga pública.

Cenários
Registrar estacionamento {Principal}
1. O usuário/ condutor estaciona o veículo no local desejado;

2. O usuário faz *login* no aplicativo;

3. O usuário acessa o menu de registro de estacionamento;

4. O sistema valida a posição do usuário/ condutor para registro de estacionamento;

5. O sistema solicita os dados do veículo que fará o registro no local;

6. O sistema verifica se o usuário possui créditos válidos;

7. O sistema insere os dados do registro no histórico, finaliza o registro do estacionamento e inicia o contador (relógio);

8. Nos últimos 10 minutos válidos do estacionamento o sistema emite um aviso para o usuário sobre a validade do estacionamento;

9. Nos últimos 5 minutos válidos do estacionamento o sistema emite um aviso para o usuário sobre a validade do estacionamento;

10. O registro de estacionamento do usuário expira.

Veículo não associado ao usuário {Alternativo}
No passo 5 do cenário principal, caso o veículo inserido ainda não esteja vinculado ao usuário solicitante, o sistema exibe confirmação para "nova associação de veículo x usuário". Retorna ao passo 6 do cenário principal.

Usuário não possui créditos {Alternativo}
No passo 6 do cenário principal, o usuário não possui créditos para registrar seu estacionamento. Ele será redirecionado ao passo 2 do cenário principal do Caso de Uso UC04.

Usuário descarta o aviso {Alternativo}
Nos passos 8 e/ ou 9 o usuário pode optar por descartar o aviso de término do registro de estacionamento. Retorna então ao passo 10 do cenário principal.

Local para estacionamento não localizado {Exceção}
No passo 4 do cenário principal, o sistema não localiza qualquer vaga no raio de espaço em que o condutor se encontra. O condutor deve se reposicionar ou o processo será abortado.

UC13 – Emitir Notificação
Permite a monitora efetuar notificações de autuação ("Amarelinhas") para os veículos estacionados de forma irregular na Área Azul, seja por não possuir registro de estacionamento ou pelo mesmo ter expirado.

Constraints
Pré-condição. O usuário/ monitora deve possuir acesso à Internet.

Pré-condição. O usuário/ monitora deve possuir o aplicativo instalado e funcional em seu dispositivo móvel.

Pré-condição. O usuário/ monitora deve possuir o aplicativo instalado e funcional em seu dispositivo móvel.

Pré-condição. O usuário deve estar em uma sessão válida.

Pós-condição. A monitora emitiu a notificação para o veículo.

Cenários
Emissão da notificação {Principal}
1. O usuário acessa o módulo de consulta de registros;

2. O sistema verifica a posição da monitora (usuário) e exibe os pontos de Área Azul mais próximos;

3. O usuário seleciona o ponto de Área Azul que está efetuando a verificação;

4. O usuário insere a placa do veículo que quer consultar;

5. O sistema exibe informações sobre o veículo e informa se o mesmo está irregular na vaga ou não;

6. O usuário faz a verificação visual em busca de cartão de Estacionamento Área Azul preenchido

7. O usuário solicita a emissão de notificação para o veículo irregular/ inexistente;

8. A emissão da notificação é registrada e exibida mensagem de sucesso para o usuário.

Veículo com situação regular {Alternativo}
No passo 5 do cenário principal, caso o veículo esteja com situação regular na vaga, o sistema não permite a emissão de notificação. Exibe a mensagem para o usuário e volta para a tela de consulta de placas.

Veículo irregular com notificação já emitida {Alternativo}
No passo 5 do cenário principal, caso o veículo esteja com situação irregular na vaga, porém já foi emitida uma notificação para o mesmo naquele período, o sistema não permite a emissão de notificação. Exibe a mensagem para o usuário e volta para a tela de consulta de placas.

APÊNDICE B – Descrição do Dicionário de Dados

Este Apêndice apresenta a descrição das tabelas do banco de dados apresentadas na seção de especificação deste trabalho. Nota-se divergências nos tipos de dados do diagrama de entidade e relacionamento da Figura **7** em relação aos tipos de dados utilizados fisicamente nas tabelas, devido a inexistência de certos tipos de dados específicos na ferramenta utilizada para construção do diagrama. Os tipos de dados utilizados nos atributos são:

a) *integer*: para variáveis numéricas inteiras;
b) *int*: para variáveis numéricas inteiras utilizadas em relacionamentos;
c) boolean: para variáveis booleana (verdadeiro e falso);
d) varchar: para variáveis textuais/ descritivas;
e) *serial*: para variáveis numéricas inteiras auto incrementáveis;
f) *timestamp*: para variáveis do tipo data e hora;
g) *float8*: para variáveis numéricas flutuantes (coordenadas geográficas);
h) *numeric*: para variáveis numéricas inteiras grandes;
i) *decimal*: para variáveis numéricas decimais.

Todas as tabelas possuem ao menos um atributo identificado como chave primária, ou *Primary Key* (PK). As tabelas envolvidas em relacionamentos possuem chaves estrangeiras, ou *Foreign Key* (FK).

No Quadro 5 apresenta-se a tabela PERFIL. Nela serão armazenados os tipos de perfis (Condutor, Monitor, Operador).

Quadro 5 – Tabela PERFIL

Entidade: PERFIL					
Descrição: armazena os códigos de perfis disponíveis					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
ID	Sim	Não	int		Código do perfil
DESCRICAO	Não	Não	varchar	30	Descrição do perfil

No Quadro 6 apresenta-se a tabela PESSOA, na qual todas os usuários do sistema serão armazenados.

Quadro 6 – Tabela PESSOA

Entidade: PESSOA					
Descrição: armazena as pessoas (usuários) do sistema					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**

ID	Sim	Não	*serial*		Código da pessoa
CPFCNPJ	Sim	Não	*numeric*		CPF da pessoa
DTBIRTHDAY	Não	Não	*timestamp*		Data de nascimento
DTLASTLOGIN	Não	Não	*timestamp*		Data de último *login*
NAME	Não	Não	varchar	255	Nome
EMAIL	Não	Não	varchar	60	*E-mail*
USERNAME	Sim	Não	varchar	30	Nome de usuário
PASSWORD	Não	Não	varchar	50	Senha
CREDITOS	Não	Não	*decimal*		Créditos do usuário

No Quadro 7 apresenta-se a tabela PESSOA_PERFIL. Nela serão armazenados os relacionamentos entre os usuários e respectivas permissões, que concede acesso aos mesmos a determinados módulos do sistema.

Quadro 7 – Tabela PESSOA_PERFIL

Entidade: PESSOA_PERFIL					
Descrição: armazena as relações entre as pessoas e perfis de acesso					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
IDPESSOA	Sim	Sim	int		Código da pessoa
IDPERFIL	Sim	Sim	int		Código do perfil

No Quadro 8 apresenta-se a tabela VEICULO, na qual serão armazenados os veículos utilizados nos eventos de estacionamento.

Quadro 8 – Tabela VEICULO

Entidade: VEICULO					
Descrição: armazena os veículos utilizados nos registros					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
PLACA	Sim	Não	varchar	8	Placa
NMPROPRIETARIO	Não	Não	varchar	255	Nome do proprietário
RENAVAM	Não	Não	*numeric*		Número do Renavam
MODELO	Não	Não	varchar	15	Modelo do veículo
MARCA	Não	Não	varchar	15	Marca do veículo
ANO	Não	Não	int		Ano do veículo

No Quadro 9 apresenta-se a tabela CONDUTOR_VEICULO. Esta tabela possui uma ligação entre as tabelas VEICULO e PESSOA, representando assim um condutor (pessoa que conduz um veículo).

Quadro 9 – Tabela CONDUTOR_VEICULO

Entidade: CONDUTOR_VEICULO

Descrição: armazena as relações entre pessoas e veículos					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
PLACA	Sim	Sim	varchar	8	Placa
IDPESSOA	Sim	Sim	int		Id da pessoa

No Quadro 10 apresenta-se a tabela TIPO_PONTO. Nela serão armazenados os tipos básicos de pontos de acordo com sua finalidade: para carros e para motos.

Quadro 10 – Tabela TIPO_PONTO

Entidade: TIPO_PONTO					
Descrição: armazena os tipos de ponto					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
ID	Sim	Não	int		Código do tipo do ponto
DESCRICAO	Não	Não	varchar	30	Descrição do tipo do ponto

No Quadro 11 apresenta-se a tabela ponto, na qual os pontos de estacionamento serão cadastrados. Ela possui quatro pares de coordenadas geográficas, para que os grandes pontos de estacionamento tenham mais de um centro lógico (coordenada central utilizada para aferir a distância entre o condutor e o ponto de estacionamento).

Quadro 11 – Tabela PONTO

Entidade: PONTO					
Descrição: armazena os pontos de estacionamento (locais)					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
ID	Sim	Não	*serial*		Número sequencial do ponto
CORDSUPESQ_X	Não	Não	float8		Latitude 1
CORDSUPESQ_Y	Não	Não	float8		Longitude 1
CORDSUPDIR_X	Não	Não	float8		Latitude 2
CORDSUPDIR_Y	Não	Não	float8		Longitude 2
CORDINFESQ_X	Não	Não	float8		Latitude 3
CORDINFESQ_Y	Não	Não	float8		Longitude 3
CORDINFDIR_X	Não	Não	float8		Latitude 4
CORDINFDIR_Y	Não	Não	float8		Longitude 4
ATIVO	Não	Não	boolean		Indica se o ponto está ativo
DTCADASTRO	Não	Não	*timestamp*		Data de cadastro
DSPONTO	Não	Não	varchar	255	Descrição do ponto
IDTIPOPONTO	Não	Sim	int		Tipo de ponto (carro ou moto)

No Quadro 12 apresenta-se a tabela EVENTO_ESTACIONAMENTO. Todo e qualquer evento ou ação que envolva um veículo e um ponto de estacionamento irá compor um registro nesta tabela. Ela identifica os envolvidos no evento: o monitor (quando houver), o

condutor (quando houver), o local de estacionamento e o veículo.

Quadro 12 – Tabela EVENTO_ESTACIONAMENTO

Entidade: EVENTO_ESTACIONAMENTO					
Descrição: armazena todos os eventos relacionados às vagas de estacionamento					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
ID	Sim	Não	*serial*		Número sequencial do evento
IDMONITOR	Não	Sim	int		ID do monitor
IDCONDUTOR	Não	Sim	int		ID do condutor
PLACA	Não	Sim	varchar	8	Placa do veículo
IDPONTO	Não	Sim	*integer*		ID do ponto
DT_EVENTO	Não	Não	*timestamp*		Data e hora do evento
ATIVO	Não	Não	boolean		Indica se o evento está ativo

No Quadro 13 apresenta-se a tabela TIPO_EVENTO. Nela serão armazenados os tipos de eventos primários (registro de estacionamento, verificação/ consulta e notificação).

Quadro 13 – Tabela TIPO_PONTO

Entidade: TIPO_EVENTO					
Descrição: armazena os tipos de registros (detalhes de um evento)					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
ID	Sim	Não	int		Código do tipo de registro
DESCRICAO	Não	Não	varchar	30	Descrição do registro

No Quadro 14 apresenta-se a tabela DETALHE_REGISTRO. Sempre que um evento de estacionamento é criado é gerado um registro nesta tabela com a especificidade do evento criado. Caso haja um evento de estacionamento ativo para o veículo e ponto em questão, apenas será criado um novo registro na tabela DETALHE_REGISTRO. A relação de foreign key (FK) entre esta tabela e a tabela TIPO_PONTO identifica qual o tipo do detalhe efetuado e em soma ao atributo SEQ_EVENTO, pode-se determinar, cronologicamente, quais foram as ações tomadas sobre cada evento de estacionamento.

Quadro 14 – Tabela DETALHE_REGISTRO

Entidade: DETALHE_REGISTRO					
Descrição: armazena os detalhes de eventos (informações mais detalhadas)					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
ID	Sim	Não	*serial*		Número sequencial do registro
IDTIPOEVENTO	Não	Sim	*integer*		Tipo do detalhe do evento
IDEVENTO	Não	Sim	int		ID do evento pai
SEQ_EVENTO	Sim	Não	int		Número de sequência do registro
DETALHE	Não	Não	varchar	255	Descrição do evento

DT_DETALHE	Não	Não	*timestamp*		Data de criação do registro
ATIVO	Não	Não	boolean		Indica se o detalhe está ativo

No Quadro 15 apresenta-se a tabela NOTIFICACAO. Sempre que um evento do tipo "Notificação" for gerado na tabela DETALHE_REGISTRO automaticamente é gerado um registro na tabela NOTIFICACAO, para controle detalhado das notificações emitidas.

Quadro 15 – Tabela NOTIFICACAO

Entidade: NOTIFICACAO					
Descrição: armazena as notificações de estacionamento emitidas					
Campo	**PK?**	**FK?**	**Tipo**	**Tamanho**	**Descrição**
ID	Sim	Não	*serial*		Número sequencial da notificação
IDREGISTRO	Não	Sim	int		ID do registro-pai
PENDENTE	Não	Não	boolean		Indica de a notificação está pendente