Thiago José Silveira

Sistema de controle de acesso automotivo

São José – SC

Novembro / 2014

Thiago José Silveira

Sistema de controle de acesso automotivo

Monografia apresentada à Coordenação do Curso Superior de Tecnologia em Sistemas de Telecomunicações do Instituto Federal de Santa Catarina para a obtenção do diploma de Tecnólogo em Sistemas de Telecomunicações.

Orientador:
Prof. Eraldo Silveira e Silva, Dr.

CURSO SUPERIOR DE TECNOLOGIA EM SISTEMAS DE TELECOMUNICAÇÕES
INSTITUTO FEDERAL DE SANTA CATARINA

São José – SC

Novembro / 2014

Monografia sob o título *"Sistema de controle de acesso automotivo"*, defendida por Thiago José Silveira e aprovada em 23 de fevereio de 2014, em São José, Santa Catarina, pela banca examinadora assim constituída:

Prof. Eraldo Silveira e Silva, Dr.
Orientador

Prof. Marcio Henrique Doniak, M. Eng.
IFSC

Prof. Tiago Semprebom, Dr.
IFSC

Algo só é impossível até que alguém duvide e resolva provar ao contrário.

Albert Einstein

Agradecimentos

Dedico meus sinceros agradecimentos à todos os familiares que deram forças para continuar essa jornada árdua, em especial a minha avó, Alvina Faustino Pereira que faleceu durante a conclusão deste trabalho, aos colegas que por muitas vezes me ajudaram a entender assuntos que pouco dominava e que com trabalho em grupo, conseguimos superar o desafio, aos colegas de trabalho que deram um suporte extra e a todos os professores envolvidos, que dedicam grande parte de suas vidas em passar conhecimento aos alunos.

Resumo

Sistemas de alarme e segurança são largamente empregados em automóveis. Normalmente o objetivo destes sistemas é detectar intrusões e disparar algum tipo de sinalização. Este trabalho propõe um sistema de controle automotivo com características diferenciadas daqueles normalmente encontrados no mercado, trazendo mais segurança e controle na utilização para os proprietários de veículos. A ideia central é que o proprietário de um ou mais veículos possa autorizar pessoas e definir períodos de uso do mesmo, utilizando um **subsistema de gerenciamento de usuários** com interface web. Um **subsistema portátil de identificação** é usado para receber o perfil do usuário e para interagir com um **subsistema embarcado no automóvel** de forma a permitir dar acesso e ligar o veículo. Através de uma rede celular de dados, o subsistema embarcado pode ainda interagir com o subsistema de gerenciamento de usuários para procurar dados adicionais. Este último, realiza ainda as tarefas normais de um sistema de segurança automotiva O sistema inicialmente concebido prevê também uma interface com um sistema de autoridade de trânsito, de forma a incluir na autorização de condução a habilitação legal de direção. O sistema foi parcialmente implementado usando uma aplicação web com PHP/Apache, uma placa Raspberry Pi a ser embarcado no veículo e um dispositivo celular com Android que é usado como dispositivo portátil de identificação. A interação entre o dispositivo portátil e os demais sistemas é realizado através de uma interface Bluetooth.

Abstract

Alarm and security systems are widely adopted in vehicles. Usually the aim of these systems is to detect intrusions and trigger some sort of signaling. This work proposes an automotive control system with different characteristics than those usually found in the market, adding more security and control for vehicle owners. The central idea is that the owner of vehicles may authorize people to drive them in specific time intervals, using a web interface. A portable identification subsystem is used to receive the drivers profile and to interact with an embedded subsystem in the vehicle to allow access and to start the vehicle. Using a cellular data network, the embedded subsystem may also interact with the user management subsystem to search for additional data. The proposed systems still performs the regular tasks of an automotive safety system. The system was conceived to provide an interface with a transit authority system and to allow the use of a electronic driving permission in the future. The system was partially implemented using a web application with PHP / Apache, a Raspberry Pi card to be embedded in the vehicle and a mobile device with Android that is used as a portable identification device. The interaction between the handheld and the other systems is performed via a Bluetooth interface.

Sumário

Lista de Figuras

Lista de Tabelas

1 Introdução

Desde muito tempo os fabricantes e a sociedade, vem buscando formas de aumentar a segurança dos veículos, dificultando a ação de ladrões de automóveis, agindo principalmente na forma de detectaçao da intrusão ao veículo.

Hoje os fabricantes utilizam vários métodos para inibir a ação de pessoas não autorizadas, mas poucas trazem controle para o proprietário do veículo para que ele defina quem está autorizado ou não a dirigir o seu veículo.

Fabricantes como a Renault já possuem nas suas linhas comerciais de veículos um sofisticado sistema de chave com RFID. É possível abrir a porta de um veículo pela simples aproximação do mesmo. O veículo pode ser ligado sem inserção mecânica da chave no sistema de ignição.

Entretanto, na visão do autor deste trabalho, algumas características adicionais ainda podem ser inseridas em sistemas de segurança veicular. Por exemplo, a possibilidade de um dono de veículo definir quais condutores estão aptos a dirigir o veículo em específico. Poderia ser o caso de uma mãe ou pai definir quais filhos ou parentes podem dirigir seu veículo e em que períodos. Em adição, este sistema também poderia comunicar-se com um sistema de autoridade de trânsito no sentido de identificar se os condutores estão habilitados ou não a dirigir.

Outras características poderiam ser incorporadas a estes sistemas. O controle de acesso a uma garagem poderia também estar integrado a este sistema de autorizaçao e controle de acesso. Em outra situação mais avançada, a própria carteira de motorista poderia servir como dispositivo portátil de acesso a um veículo.

Um sistema com as características citadas poderia ser utilizado por empresas para controle de frotas e, em caso mais amplo, abre a possibilidade de o sistema ser utilizado para controle governamental.

É reconhecido que esta área é fortemente explorada pelos grandes fabricantes de carros, porém, entende-se que um sistema desta natureza possa também ser incorporado na frota já

existente. É dentro deste contexto que se insere este trabalho.

1.1 Motivação

Uma das motivações para o desenvolvimento deste trabalho foram as reportagens vinculadas na TV, que mostravam acidentes de trânsito ocasionados por condutores não habilitados ou que perderam do direito de conduzir veículos. Com este problema em mente, pensou-se em formas de o veículo identificar individualmente os condutores, o que abre precedente para impossibilitar condutores não habilitados de conduzir veículos.

Os relatos de amigos quem tem os seus veículos compartilhados entre os familiares e que reclamavam da falta de controle sobre o uso de seu veículo, também motivou o autor no sentido de criar um sistema de baixo custo, com a possibilidade de controlar quais os familiares podem conduzir determinados veículos.

Por fim, este sistema também pode trazer mais uma barreira contra o furto de veículo, já que o sistema pode bloquear o uso do veículo, apenas entrando em seu cadastro e bloquear o uso.

1.2 Objetivos do Trabalho e Visão Inicial da Estrutura Proposta

O trabalho proposto objetiva o desenvolvimento de um sistema de controle de acesso veicular que permita criar perfis de uso de veículos, possibilitando que um proprietário do veículo determine os condutores de seu veículo e incorporando inclusive restrições de horário e data de uso. As permissões de uso são configuradas em um dispositivo portátil e em um sistema embarcado no veículo. O sistema deve prever uma interface adicional com uma autoridade de trânsito no sentido de verificar a habilitação dos condutores.

De forma específica o sistema deve permitir:

1. Gerenciar através de uma aplicação web, o cadastro de veículos, usuários e proprietários;
2. Configurar um dispositivo portátil a ser usado pelos usuários condutores de forma a ter acesso a um determinado veículo;
3. Interagir com um sistema de autoridade de trânsito no sentido de obter informações a respeito dos condutores;

4. Executar as tarefas normais de controle e segurança automotiva (detecçao de intrusão) através de um subsistema embarcado no veículo.

Para atingir os objetivos colocados são propostos três subsistemas: (i) um dispositivo portátil de identificação, a ser portado pelo usuário e permitir o acesso a um veículo, (ii) um subsistema de cadastro de veículo e usuários (gerenciamento de usuários))e (iii) um sistema embarcado no veículo para controle do acesso. A Figura 1.1 ilustra o sistema proposto.

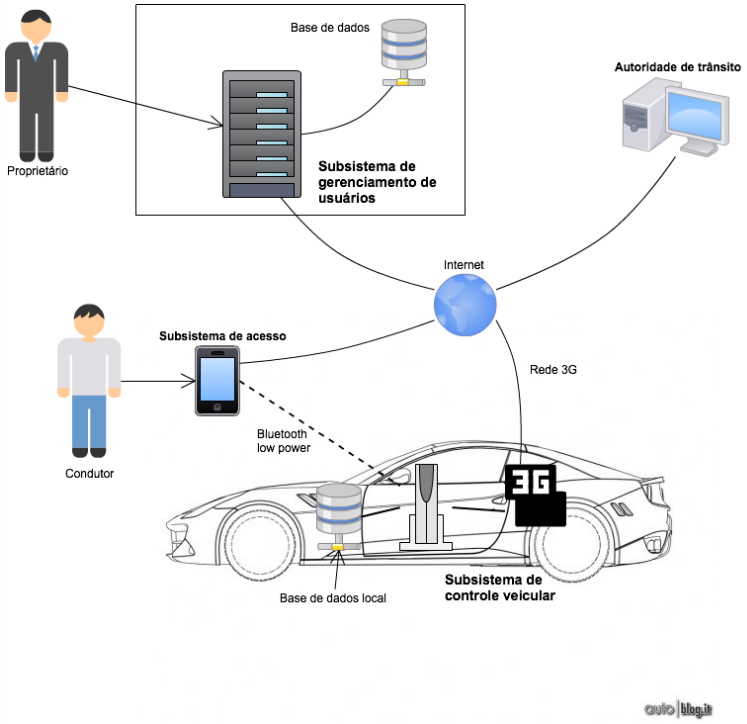

Figura 1.1: Visão do Sistema Proposto

O subsistema de gerenciamento de usuários pode interagir com a autoridade de trânsito no sentido de verificar habilitações de condutores e regularidade de veículos. O proprietário define através deste subsistema os usuários de seus veículos e limitações do acesso. Um base de dados deve ser usada como apoio ao armazenamento. O dispositivo portátil de identificação recebe o perfil de uso através de uma interface *Bluetooth Low Power*. Ainda através desta interface, o dispositivo portátil interage com o subsistema embarcado no automóvel para que este forneça acesso ao veículo e a sua condução. Através da rede 3G o susbistema embarcado pode interagir com o subsistema de gerenciamento de usuário.

1.3 Organização do texto

O texto está organizado da seguinte forma: No capítulo 2 é apresentada a fundamentação teórica. No capítulo 3 é apresenta uma visão geral do sistema desenvolvido seguindo uma sequência que reflete a metodologia utilizada. No capítulo 4 é mostrada como foi relizada a implementação do sistema, bem como os testes realizados. Por fim, no capítulo 5 são apresentadas as conclusões sobre este trabalho.

2 *Fundamentação Teórica*

Este capítulo inicialmente realiza um panorama de sistemas de controle de acesso que se aproximam ao sistema proposto. Na sequência serão apresentadas tecnologias e ferramentas utilizadas no desenvolvimento do sistema.

Conforme citado na introdução, o sistema proposto apresenta três subsistemas: (i) um dispositivo portátil de identificação, a ser portado pelo usuário e permitir o acesso a um veículo, (ii) um subsistema de cadastro de veículo e e usuários e (iii) um sistema embarcado no veículo para controle do acesso. Seguindo esta ordem, será realizada uma visão geral das tecnologias utilizadas no subsistema que corresponde ao dispositivo portátil de identificação. Inclui-se a interface bluetooth, o sistema operacional *android* e o App Inventor, um sistema de desenvolvimento de aplicações para o Android. A placa microprocessadora *Raspberry Pi* é descrita pelo fato de ser utilizada no subsistema embarcado no veículo. Finalmente, são brevemente revistas as tecnologias usadas na implementação do subsistema de gerenciamento de cadastro de veículos e usuários. Em particular, discute-se a abordagem utilizada no desenvolvimento da aplicaçao web utilizado e a tecnologia de banco de dados.

2.1 Visão Geral de Sistemas de Identificação/segurança Automotivos Existentes

Muitos estudos e aplicações vem sendo realizados na área de identificação e segurança automotiva. Muitos carros desta montadora já dispõe de chaves baseadas em RFID que permitem abrir um veículo pela simples aproximação e ligá-lo estando com a chave (cartão) no bolso do condutor (RENAULT, 2012). Apesar de ser um sistema sofisticado, o mesmo não possui as características pretendidas neste trabalho.

Em El Salvador está sendo implantado um sistema de identificação de condutores, utilizando um Smart Card (cartão com um circuito integrado que possibilita várias aplicações), no lugar da carteira de motorista convencional (SERTRACEN, 2013a). Neste sistema o agente

de trânsito, através de um equipamento consegue ter a leitura e confiança de que os dados de identificação são autênticos e estão atualizados. Através deste sistema conseguiu-se um aumento na arrecadação de impostos, diminuição da criminalidade e melhora nos hábitos de condução (SERTRACEN, 2013b).

Em (AOYAMA, 1998), foi publicado um estudo sobre acidentes de trânsito onde está descrito o uso de um sistema de identificação usando *Integrated Circuit (IC) Card* (semelhante ao *Smart Card*). Neste sistema é necessário inserir o cartão no veiculo, que por sua vez identifica o condutor e armazena todos os dados de condução do veículo no cartão, posteriormente esses dados são verificados no ato da renovação da carteira.

No escopo da segurança contra roubo de automóveis, hoje no mercado temos inúmeros alarmes e rastreadores, vale citar uma proposta interessante, de dois alunos da Universidade Federal do Paraná (GOMES, 2012), que desenvolveram parte de um projeto de alarme inteligente, que utiliza tecnologia de baixo custo para informar ao proprietário via *Short Message Service (SMS)*, quando o seu carro foi violado e possibilita o rastreio do veiculo via *Global Positioning System (GPS)*.

Várias empresas hoje no mercado oferecem sistemas de rastreio via SMS e ou *General Packet Radio Service (GPRS)*, alguns destes serviços oferecem o bloqueio do veículo a distância a pedido do proprietário. Como exemplo de serviço oferecido, pelas empresas Simtrack [1], Positron[2], KGK[3] e inúmeras outras que disponibilizam os serviços de rastreamento e bloqueio remoto do veículo.

2.2 Tecnologia Bluetooth

No sistema proposto neste trabalho, a comunicaçao entre o dispositivo portátil de identificação e os demais subsistemas é realizada através da tecnologia Bluetooth (BLUETOOTH, 2014b). A tecnologia Bluetooth é hoje uma das tecnologias de comunicação sem fio, mais difundidas do mundo, que conta com mais de 20 mil companhias associadas. A versão Bluetooth 4.0 com suporte a baixo consumo de energia (BLE) foi liberada em 2010.

Na Bluetooth clássica são previstos modo de transmissão DR (*data rate*) e EDR (*enhanced data rate*) com taxas de transmissão de 1Mbps e 3 Mbps respectivamente. Na versão 3.0 foi incluído uma opção de alta velocidade (até 24Mbps). A camada de enlace prevê conexões

[1] http://www.simtrack.com.br
[2] http://www.positron.com.br/rastreamento/tecnologia
[3] http://www.kgk-global.com/

asíncronas, para dados em rajadas, e síncronas para dados cadenciados, tal como a voz (TELECO, 2014).

A ideia inicial neste trabalho foi de utilizar a Bluetooth de baixa energia (BLE). A BLE foi concebida para uso em sensores de baixíssimo consumo e cuja finalidade é comunicar estruturas de dados de pequeno tamanho. Ela se utiliza da faixa de 2.4Ghz, transmissão a 1Mbps e alcance até 50m. Estas características fazem da BLE uma das principais tecnologias da Internet das Coisas. Deve-se ressaltar que a BLE pode conviver (em dispositivos *smart*) com a Bluetooth convencional e que pode ser usada para transferência de voz e imagem.

Como a Bluetooth é altamente disponível em tablets e celulares, avaliou-se que esta tecnologia seria mais apropriada para o dispositivo portátil de identificação proposto. Ela permite a reutilização telefones celulares para a finalidade de liberaçao dos automóveis e, se necessário, pode-se construir a baixo custo um dispositivo específico.

Nas versões da Bluetooth a partir da 4.0 as características da Bluetooth clássica podem conviver com a de baixa energia conforme mostrado na Figura 2.1.

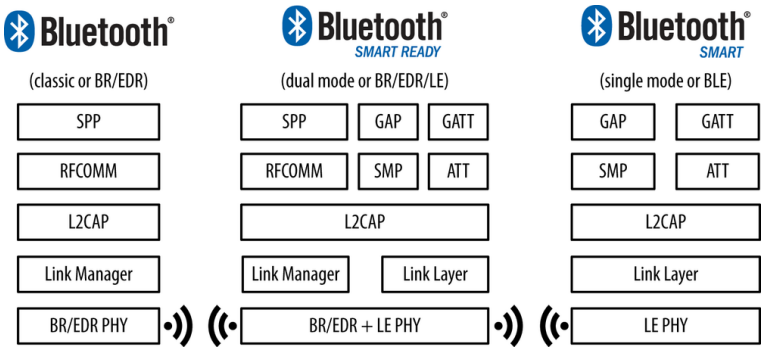

Figura 2.1: Bluetooth em Modo Dual: clássica e baixa energia (SAFARI BOOKS ON LINE, 2015)

Um dispositivo pode conter todas as camadas necessárias a Bluetooth clássica e BLE tal como é mostrada na camada do centro da Figura (funcionamento dual mode). Um celular com interface dual mode pode se conectar a um dispositivo clássico, por exemplo um fone de ouvido, e também a um dispositivo BLE, por exemplo, um sensor de batimento cardíaco.

É importante observar que a camada física da Bluetooth se utiliza do salto de frequência (frequency hop) segundo um padrão pré-definido a partir de um mestre. A sequência é adaptada para evitar canais em utilização por outras tecnologias o que traz uma grande robustez (SAFARI BOOKS ON LINE, 2015).

Na sequência serão fornecidos mais alguns detalhes sobre a BLE dado que trata-se de uma evolução recente da tecnologia.

2.2.1 Visão Geral do Funcionamento e da Arquitetura

O padrão BLE da Bluetooth se utiliza de dispositivos mestre e escravos para a construção de piconets. Uma piconet é uma rede formada por até 8 dispositivos, sendo 1 mestre e os demais escravos (TELECO, 2014). O funcionamento do BLE pode ser resumido a três fases:

1. Fase de varredura (estado *scanning*) e advertência (estado *advertsisng*): um dispositivo mestre varre canais de controle esperando por advertências de outros dispositivos. Estas advertências podem trazer dados ou pedidos de conexão para formar piconets;

2. Fase de estabelecimento de conexão, com a troca de capacidades: nesta fase o mestre pode descobrir os serviços ofertados pelo escravo;

3. Fase de leitura/escrita de atributos. Os dados a serem transmitidos/lidos são vistos como atributos que o sensor (visto como um servidor) mantêm atualizados e fornece caso seja solcitado.

Para auxiliar na compreensão da arquitetura de protocolo, seguindo a visão apresentada por (SIQUEIRA, 2006), pode-se dividí-la em três grupos lógicos: transporte, *middleware* e aplicação. A camada de aplicação está associada as aplicações propriamente ditas, podendo ser aplicações legadas ou aplicações orientadas a Bluetooth. Na sequência são vistas as camada de transporte e o middleware.

2.2.2 A camada de Transporte

A camada de transporte inclui o conjunto de protocolos responsável por gerenciar links físicos e lógicos, além de localizar outros dispositivos. Nota-se que o termo transporte não está associado a camada de transporte do modelo OSI mas sim as camadas física e enlace. Dentro dessa camada de transporte do Bluetooth, estão inclusas os blocos de Radio Frequência (RF), *Baseband*, *Link Manager* e *Logical Link Control and Adaptation Protocol (L2CAP)*.

O bloco RF é responsável por transmitir e receber dados de um canal físico e permite que o bloco baseband controle o seu tempo e portadora. Este bloco também passa a modulação do canal para banda base. Para a transmissão é utilizada a faixa de frequência de 2.4GHz, dividido em 40 canais de 2MHz de banda cada, diferente dos 79 canais e 1MHz de banda cada da versão anterior, também reserva 3 canais 36, 37 e 38, usados para descobrir outros dispositivos. O restante dos canais é usado para transferência de dados (UFRJ, 2012), na Figura 2.2 é mostrado

como é organizada a transmissão. É usado a abordagem de salto de frequência entre canais para a determinaçao de um canal de menor ruído.

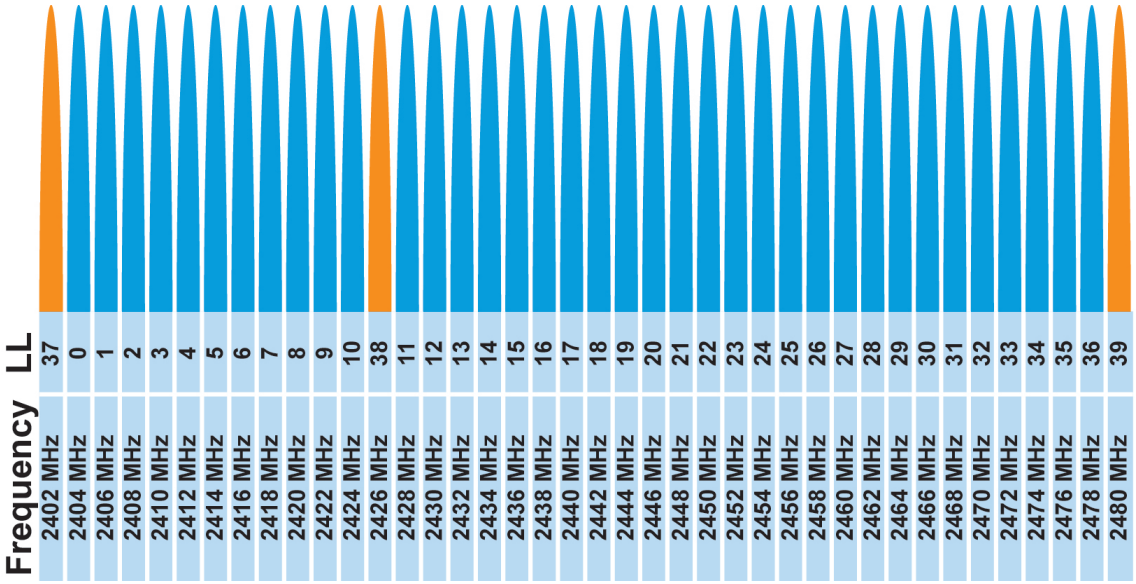

Figura 2.2: Modelo de transmissão (CONNECTBLUE, 2014a)

O *Link Controller* é responsável por codificar e decodificar os dados recebidos da camada superior/inferior, informar se o pacote foi recebido, adicionar parâmetros para o funcionamento da camada abaixo e controlar o estado da conexão. Os estados possíveis (UFRJ, 2012) (CONNECTBLUE, 2014b) são:

- *Standby*: Ocorre quando o dispositivo não recebe e nem envia pacotes;
- *Advertising*: Esse estado possibilita enviar pacotes por cada *advertising channel* e pode responder a requisições de outros dispositivos;
- *Scanning*: Nesse estado o dispositivo fica escutando transmissões *advertising* e se necessário pode responder solicitando mais informações;
- *Initiating*: O dispositivo entra nessa estado, quando tem a intenção de formar uma conexão;
- *Connection*: Após a troca de estados entre *Advertising* e *Initiating*, o dispositivo entra no estado *Connection* que após estabelecido, passa a transferir dados através dos *data channels*.

Após formada uma conexão, os dispositivos assumem cada um o seu papel de mestre, dispositivo que entra no estado de *Initiating*, e escravo, dispositivo que estava no estado *Advertising*. Com o objetivo de poupar energia, o dispositivo escravo fica em modo *sleep*, já o mestre

tem a tarefa de acordar o dispositivo escravo em intervalo de tempo, pré determinado (UFRJ, 2012). Através do *Advertising* é que será proposto enviar dados sem a necessidade de formar uma conexão.

O *Baseband Resource Manager* gerencia todo o acesso ao radio. Ele negocia os tempos de acesso entre as entidades e o meio físico, assim como a qualidade de serviço para as suas entidades (BLUETOOTH, 2010).

O *Link Manager* é responsável por criar, modificar ou liberar ligações lógicas, assim como atualizações de parâmetros das ligações físicas (BLUETOOTH, 2010). O protocolo L2CAP é um protocolo simplificado responsável pelo encapsulamento das mensagens de camadas superiores. Ele é o elo de ligação entre o transporte e o middleware conforme pode-se observar na Figura 2.1.

2.2.3 Middleware

A camada de protocolos *Middeware*, é composta por protocolos de terceiros, padrões industrias e padrões proprietários desenvolvidos pelo próprio *Special Interest Group (SIG)*, isso proporciona uma maior integração com aplicações já existentes (SIQUEIRA, 2006). Foram propostos na versão BLE 4.0, vários protocolos, dentre eles é destacado o protocolo *Generic Attribute Profile (GATT)* e o *Attribute Protocol (ATT)*. Em uma possível integração com o mundo IP, estes protocolos seriam substituídos pelo TCP/IP ou UDP/IP, embora não exista uma correspondência direta entre eles.

O GATT é uma camada acima do ATT que estabelece operações comuns e um *framework* para os dados trasportados e armazenados pelo ATT. É obrigatória a utilização do GATT e ATT para implementar os perfis Bluetooth Low Energy (LE) (BLUETOOTH, 2014a).

Algumas vantagens podem ser citadas, com o uso dessa arquitetura:

- Desenvolvimento e implementação facilitados;
- O ATT foi otimizado para rodar em dispositivos de baixo consumo de energia, pois usa o menor número de bytes possível, e as implementações podem utilizar estruturas de tamanho fixo na memória para fazer Protocol Data Units (PDUs);
- Os protocolos agora tem simplicidade, oferecendo facilidades em sua implementação e diminuindo as chances de provocar erros no microcontrolador;

- Pode existir alguns perfis[4] GATT[5] que não são os ideais, mas pode haver uma segunda conexão com L2CAP em paralelo com o canal de ATT, que por sua vez implementa um protocolo específico do perfil.

Os serviços do GATT são uma coleção de informações e comportamentos associados para realizar uma função característica de um dispositivo ou parte de um dispositivo especifico (BLUETOOTH, 2014a).

As funcionalidades dos serviços do GATT, funcionam atribuindo um *Universally Unique Identifier (UUID)* (0x2800) para todos os atributos que seguem e pertencem a esse serviço, até que o campo UUID contenha novamente o valor 0x2800 (EPX, 2014).

Um exemplo de serviço GATT poderia ser acionamento de LEDs de uma placa específica. Supondo que esta placa tenha um LED vermelho e outro VERDE, então cada LED seria uma atributo, cada uma com um UUID. O GATT permite definir estas características bem como as operações que podem ser realizadas sobre cada um deles. As operações poderiam ser ligar/-desligar os LEDs de forma indepedente. Cabe ao protocolo ATT encaminhar o transporte de valores quando solcitados.

Para o caso específico deste trabalho poderia ser criado um perfil para liberação de um automóvel que poderia incluir vários atributos.

2.3 Sistema Android e o desenvolvimento de aplicações com App Inventor

A implementaçao do dispositivo portátil de identificaçao utilizou-se de um telefone celular com sistema operacional Android. O sistema operacional Android é um dos sistemas mais difundidos do mundo (IDGNOW, 2014), o que torna o desenvolvimento para o sistema proposto, bastante vantajoso, já que atinge grande parcela do mercado.

O **Android** foi desenvolvido pelo Google em conjunto com a *Open handset alliance*. O Android é a primeira plataforma aberta, concebida para abranger dispositivos móveis (GOOGLE BLOG, 2007). Esta plataforma conta com um sistema operacional baseado no Linux e com uma inteface de manipulação direta (OPEN HANDSET ALLIANCE, 2015).

Uma das características mais importantes do Android é o gerenciamento de memória vol-

[4]Um perfil contém vários serviços que por sua vez, têm várias características (ELECTRONICS WEEKLY, 203)

[5]Perfis GATT são um conjunto de vários outros perfis específicos ou de serviços (ELECTRONICS WEEKLY, 203)

tado a economia de energia. Uma característica que contribui para isto é o uso da máquina virtual Dalvik otimizada. Cada aplicação Android se executa sobre uma máquina virtual. Cada máquina virtual se executa em um processo do kernel Linux. A Figura 2.3 mostra a arquitetura do Android.

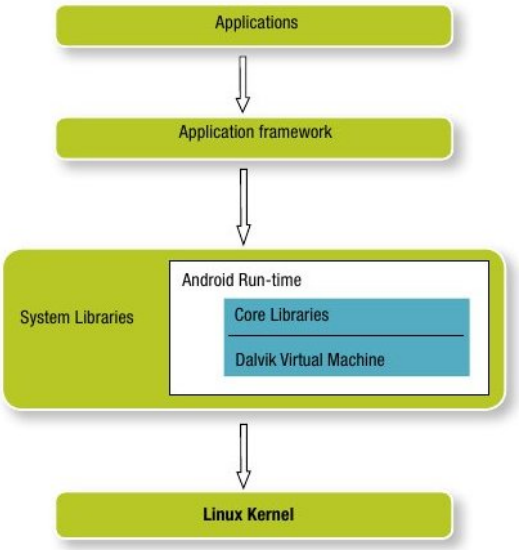

Figura 2.3: Arquitetura do Android (OPEN SOURCE FOR U, 2015)

As aplicações Android são desenvolvidas normalmente em Java. O código Java é convertido para o *bytecode* da máquina virtual Dalvik. Para este trabalho a aplicação desenvolvida no dispositivo móvel foi implementada com o App Inventor desenvolvida pelo o Instituto de Tecnologia de Massachusetts (MIT). O App Inventor usa programação em blocos gráficos, o que permite que pessoas com baixo domínio na linguagem de programação, possa fazer pequenos aplicativos de forma rápida e fácil (MIT, 2014). O código App Inventor é compilado diretamente para o código da máquina virtual Dalvik. O App Inventor disponibiliza uma série de componentes que permitem facilmente acessar o hardware do sistema. Em particular, para este projeto foram utilizados os componentes para acesso ao dispositivo Bluetooth do celular.

O MIT disponibiliza um servidor Web, no qual o usuário pode se logar e fazer a sua aplicação, sem a necessidade de instalar qualquer programa em seu computador, bastando apenas arrastar os blocos de programação e encaixá-los. Após o programa ser montado, o sistema compila a aplicação e o disponibiliza para download.

2.4 Raspberry Pi com Sistema Operacional Linux

Por questões de facilidade de desenvolvimento de protótipo, optou-se pelo uso de placa Raspberry Pi na implementação do subsistema a ser embarcado no automóvel.

Raspberry Pi é um computador de pequeno porte desenvolvido pela **Raspberry Pi Foudation** do Reino Unido. Ele conta como características principais, ter um tamanho reduzido (85.60mm x 56mm x 21mm), gerenciado pelo processador ARM de 700Mhz, suporte a câmera, USB, HDMI, VGA, Serial (I2C) e varias I/O digitais, pode ser alimentado com uma fonte de 5V e 700mA para funcionamento básico ou até 1000mA para o uso completo (RASPBERRY PI FOUDATION, 2014c) como pode ser visto na Figura 2.4. Com Raspberry Pi é possível fazer de forma básica, tudo que um computador completo pode fazer. Para este trabalho ele será utilizado como uma plataforma de fácil programação, com o intuito de gerenciar a comunicação entre os dispositivos Bluetooth, central de alarme do veículo e dados de cadastros na nuvem.

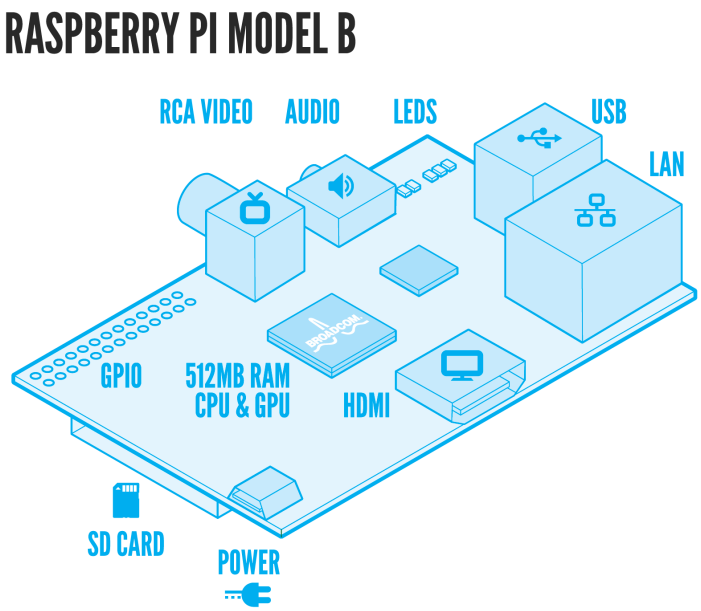

Figura 2.4: Diagrama básico Raspberry Pi (RASPBERRY PI FOUDATION, 2014a)

2.4.1 Unidade de Processamento Raspberry Pi

A unidade de processamento é um Broadcom BCM2835. O seu encapsulamento conta com um processador ARM1176JZFS, uma GPU, um controlador de memória RAM, controlador de memória SD, áudio, USB, SPI, GPIO, I2C, UART0 e câmera (RASPBERRY PI FOUDATION, 2014b), (PETERVIS, 2014), como pode ser visto na Figura. 2.5;

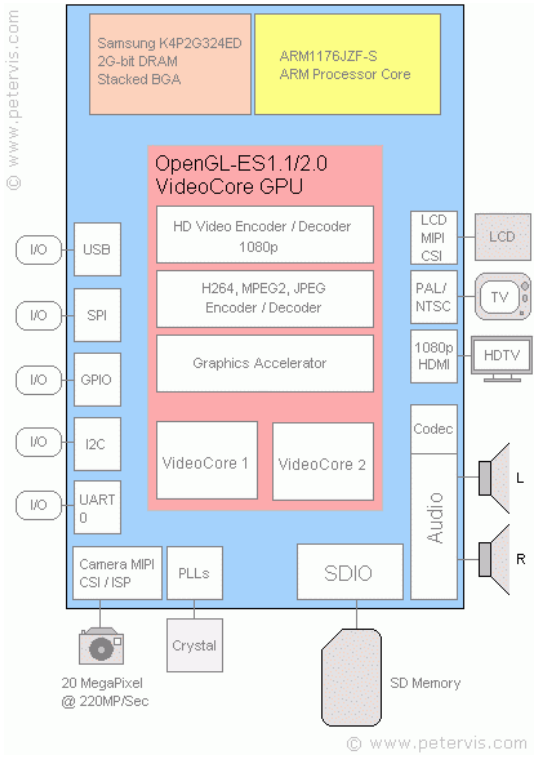

Figura 2.5: Diagrama BCM2835 (PETERVIS, 2014)

A linha de processadores ARM1176 é destinada para uso em Smart Phones, TV Digital, leitores de livros digitais, entre outros, seu clock pode chegar até 1GHz, com tensões de *overdrive* (ARM, 2014). O processador tem as seguintes características:

- Sigle Core.
- Gerenciador de memória interno.
- Baixa latência nas interrupções.
- Suporte a vários sistemas operacionais, dentre eles várias versões de Linux.

Nas Figuras 2.6 e 2.7 é mostrado um diagrama que exemplifica o funcionamento da comunicação interna e externa do processador.

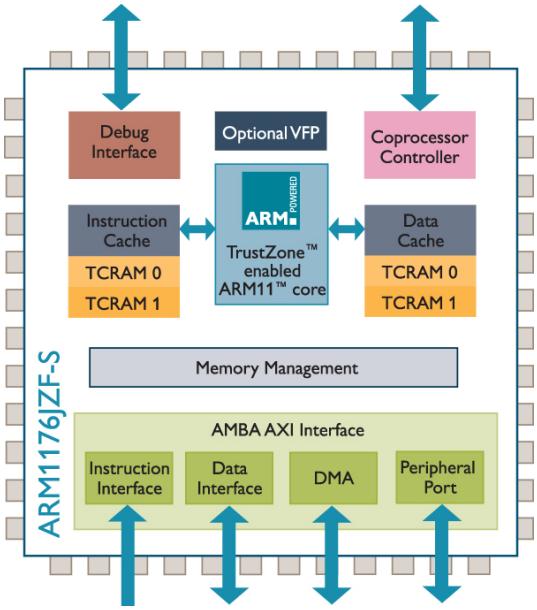

Figura 2.6: Diagrama ARM1176JZF (ARM, 2014)

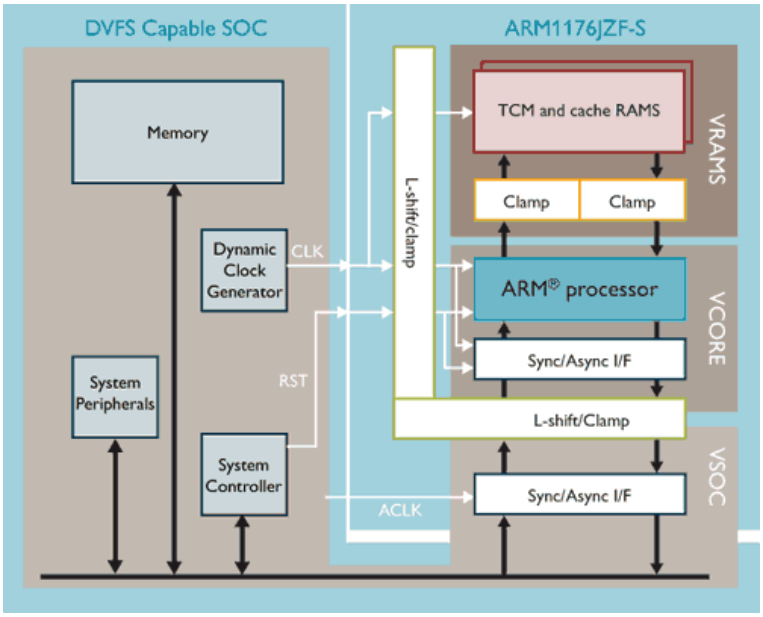

Figura 2.7: Diagrama barramento ARM1176JZF (ARM, 2014)

2.5 Tecnologias Usadas na Aplicação Web e no Armazenamento de Dados

Através do servidor hospedado de forma segura, será ofertado os serviços de cadastro, consulta e edição dos cartões de acesso, o servidor contará com as seguintes características:

- Sistema operacional Linux como plataforma;
- Aplicação PHP para comunicação entre os subsistema de controle veícular e a base de dados;
- Servidor Apache para acesso dos usuários;
- Banco de dados MySQL;
- Sistema de comunicação, via post http, usando o formato de dados JSON.

Esta estrutura servirá como base para o acesso dos usuários ao sistema, oferecendo a informação necessária para a gerencia.

2.5.1 JSON

JavaScript Object Notation (JSON) é uma notação de objetos, para troca de dados. É de fácil leitura e escrita para humanos e de fácil interpretação por máquinas. Baseado em um subconjunto da linguagem de programação JavaScript, JSON é um formato de texto completamente independente de linguagem, pois usa convensões que são comuns em grande parte das linguagens de programação.

No JSON, os dados são apresentados da seguinte forma:

O objeto JSON é um conjunto de pares desordenado, com o nome/valor. Um objeto JSON começa com chave de abertura ”{”e termina com chave de fechamento ”}”, cada nome é seguido por ”:”e os pares nome/valor, são separados por ”,”. como mostra a Figura 2.8.

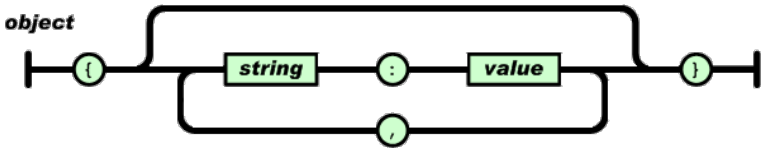

Figura 2.8: Diagrama do objeto JSON. Fonte (JSON, 2015)

No caso de uso de vetor, eles começam com conchete de abertura ”[”e terminam com conchete de fechamento ”]”, seus valores são separados por virgula ”,”, como é demonstrado na Figura 2.9.

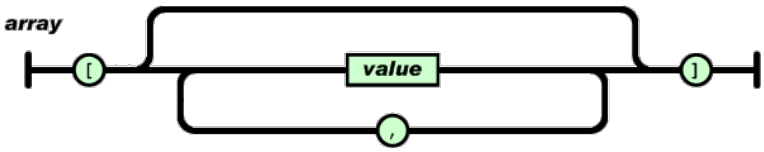

Figura 2.9: Diagrama do vetor JSON. Fonte (JSON, 2015)

Por ser amplamente usado e ser compátivel com as principais linguagens de programação, o JSON será utilizado na comunicação entre o subsistema veicular e o subsistema de gerenciamento de usuários.

2.5.2 PHP

No servidor web foi utilizado a linguagem de programação PHP com a estrutura Model-View-Control (MVC). Está linguagem é amplamente utilizada, segundo (TIOBE, 2015) e a 7 linguagem mais utilizada no mundo.

O MVC é um modelo de arquitetura de software, no qual, o desenvolvimento da aplicação é separada em três módulos: Classes (model), exibição (view) e controlador (controller). O seu funcionamento é exemplificado na Figura 2.10, o usuário faz a requisição ao **controller**, que por sua vez, manipula as informações e envia ao *model* para processamento, ao fim do processamento as informações vão para o *view*, que exibe os dados ao usuário.

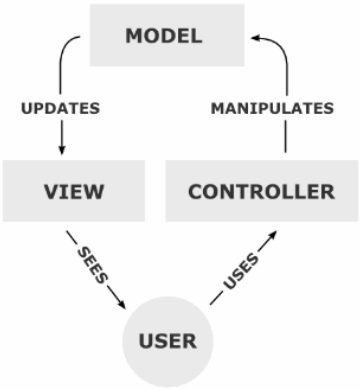

Figura 2.10: Diagrama explicativo do MVC. Fonte (SITEPOINT, 2015)

Este tipo de arquitetura deixa o código mais simples e organizado, facilitando a manutenção do código.

2.6 Conclusão

Com o conhecimento adquirido através dos estudos das tecnologias estudadas neste capítulo, foi possível implementar o sistema de forma adequada, além de trazer uma melhor compreensão do funcionamento do sistema para o leitor.

Nos próximos Capítulos, serão apresentados os subsistemas desenvolvidos para este trabalho.

3 *Sistema de Controle de Acesso Automotivo*

Este capítulo apresenta uma visão geral do sistema proposto seguindo uma sequência que reflete a metodologia utilizada no seu desenvolvimento. Inicialmente apresenta-se uma visão geral do sistema de controle de acesso automotivo, enfatizando os aspectos inovadores da proposta e uma primeira visão dos subsistemas envolvidos.

Na sequência são discutidos os requisitos e a especificação do sistema através do apresentação do principais casos de uso. O projeto é então discutido, explorando-se diagramas de classes identificadas no sistema, ou seja, uma visão estrutural estática. A dinâmica do sistema é explorada através dos principais diagramas de sequência. Não pretende-se cobrir todo o projeto mas somente àquelas partes que melhor ilustram o processo de desenvolvimento e que ajudam no entendimento do mesmo.

Finalmente, a organização da base de dados é apresentada na forma das tabelas e relacionamentos. A implementação e testes realizados serão discutidos no capítulo seguinte.

3.1 Visão geral do sistema e seus componentes

O sistema coloca no veículo mais uma camada de segurança contra furtos: além da chave para ligá-lo, o condutor deve estar cadastrado e autorizado para iniciar o seu funcionamento. Um dispositivo portátil, que pode ser um celular dotado de Bluetooth, atua como uma segunda chave. O sistema traz portanto o controle individual de cada condutor, possibilitando que o proprietário possa acompanhar o histórico de cada condutor e bloqueá-lo quando necessário.

A concepção original do sistema prevê ainda uma interface com a autoridade do trânsito de maneira que autorizações de uso incluam também restrições desta autoridade. O protótipo implementado não incorporou esta facilidade.

O sistema concebido apresenta 3 subsistemas:

- **Subsistema de acesso** ou **dispositivo portátil de identificação**: Este subsistema é responsável por se conectar e passar o identificador via bluetooth ao subsistema de controle veicular.

- **Subsistema de controle veicular** ou **subsistema embarcado**: Informa ao veículo se ele pode ser ligado, verifica se o identificador é válido, se o codutor está habilitado e busca as informações no subsistema de gerenciamento de usuários, através da Internet.

- **Subsistema de gerenciamento de usuários**: É uma aplicação web, que possibilita o cadastramento dos veículos, proprietários, identificadores, condutores e fornecee também o controle para habilitar ou não um condutor.

O sistema foi pensado para que se faça o mínimo de intervenções ou configurações para o uso. Os cadastros de veículos, proprietários e condutores, são feitos através do subsistema de controle de gerenciamento de usuários. Este subsistema fornece acesso individual através de identificação de usuário e senha, e de acordo com as credenciais do usuário, o subsistema fornece os privilégios necessários, os quais serão discutidos nas seções a seguir.

O subsistema de controle veicular é feita através de um dispositivo embarcado no veículo, com acesso a internet via 3G. Ele busca todos perfis de usuários e armazena em banco de dados próprio para consulta rápida dos condutores. As informações são armazenadas por um período pré-definindo pelo o subsistema de gerenciamento de usuários.

Através do subsistema de acesso é possível enviar o identificador do usuário, já pré-definido pelo subsistema de gerenciamento de usuários, para o subsistema instalado no veículo, que por sua vez processa as informações e envia o sinal para o veiculo habilitar a ignição.

O diagrama da Figura 3.1 proporciona uma visão geral do sistema e se constitui em uma abstração da Figura 1.1.

O sistema proposto pode potencialmente ser operado por uma empresa. Neste sentido, um administrador (do lado da empresa) deve operar o subsistema de gerenciamento de usuários de forma a cadastrar proprietários e veículos. Cada proprietário deve também acessar o sistema para cadastrar e habilitar condutores.

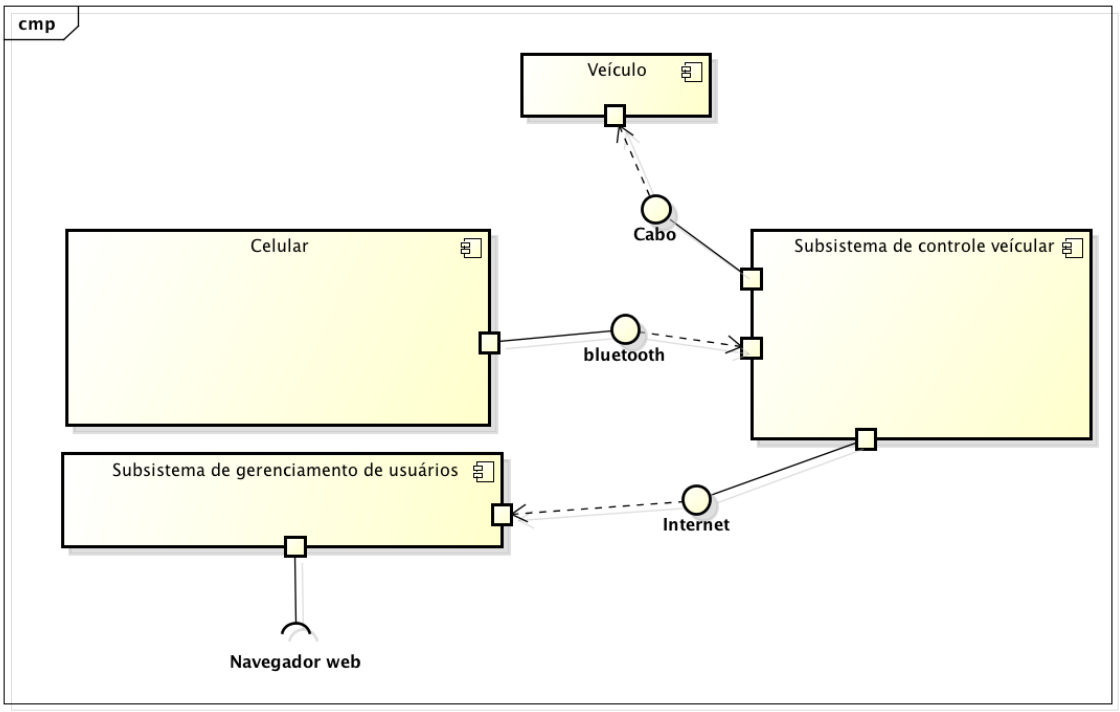

Figura 3.1: Componentes do Sistema.

3.2 Análise de requisitos e especificação do sistema

Nesta seção são apresentados os requisitos, atores e casos de usos identificados no sistema sendo então explorados com algum detalhe os principais casos de usos julgados importantes para uma visão do comportamento funcional do sistema. A notação *Unified Modeling Language (UML)* será amplamente usada.

3.2.1 Atores e requisitos funcionais

Os atores são os agentes externos que interagem com o sistema proposto. É importante a visão clara de quem são estes atores para proceder o levantamento de requisitos e os casos de uso. Tendo como referência a definição inicial do sistema realizada na sessão anterior, pode-se idenficar os seguintes atores no sistema: o administrador, os proprietários, os condutores e, no futuro, a autoridade de trânsito. O admistrador e o proprietário são especializações de um ator usuário. Temporizadores podem ser considerados também como atores. Neste sentido, no subsistema embarcado no veículo um ator temporizador deve atuar no sentido de iniciar uma renovação de credenciais.

Os principais requisitos funcionais previstos são mostradas na Tabela 3.1. Será desconsiderado a participação da autoridade de trânsito nesta primeira versão do sistema.

#	Descrição
RF1	Usuário efetua login no sistema.
RF2	O administrador mantém os veículos.
RF3	O administrador cadastra os identificadores.
RF4	O administrador mantém os proprietários.
RF5	O proprietário mantém condutores.
RF6	Usuário efetua o logoff do sistema.
RF7	Um condutor com permissão deve ser capaz de acessar/ligar o veículo.

Tabela 3.1: Requisitos funcionais.

O termo 'mantém' é usado para indicar a possibilidade de criar, editar e remover um dado elemento do sistema.

3.2.2 Casos de uso

Nesta subseção são descritos os principais casos de uso do sistema. Um caso de uso descreve uma interação de um mais atores com o sistema de modo a realizar uma determinada atividade. Os casos de usos podem servir como uma especificação funcional do sistema mostrando o comportamento do sistema do ponto de vista externo, sem se ater a como o sistema realiza internamento uma dada tarefa ou atividade.

Na Figura 3.2 é mostrado um diagrama UML que mostra casos de uso associados com os atores admiministrador e proprietário. Cada figura elipsoidal reflete um caso de uso. Por exemplo, o login implica uma sequência de interações entre um ator para obter acesso ao sistema através do fornecimento de credenciais. Na sequência são mostrados alguns dos casos de uso que foram elaborados. O login é mostrado na Tabela 3.2, a manutenção de cadastro de veículos

#	**Descrição**
Caso de Uso	Login.
Descrição	O usuário informa o seu login e senha e o sistema verifica se os dados estão corretos. Caso positivo o sistema carrega o perfil do usuário.
Atores	Administrador, Proprietário.
Precondições	O usuário deve está cadastrado para poder efetuar o login.
Fluxo principal	1. Usuário acessa a tela de login. 2. Usuário preenche o formulário. 3. O sistemma verifica se existe o usuário e se o identifcador de usuário e a senha estão corretos. 4. O usuário é liberado para acessar o sistema, de acordo com os seus privilégios de seu cadastro.
Fluxo de exceção	2.1 Preenche o usuário e senha com dados incorretos. 2.2 O sistema apresenta mensagem de erro.

Tabela 3.2: Efetuando login no subsitema de gerenciamento de usuários.

na Tabela 3.3, a manutenção do cadastro de identificadores na Tabela 3.4, a manutenção do cadastro de proprietários na Tabela 3.5 e o acesso ao veículo na Tabela 3.6.

Na metodologia seguida, os casos de uso serão a base para a identificação de classes no sistema e para a construção dos diagramas de classe e de sequência, como será visto na seção seguinte.

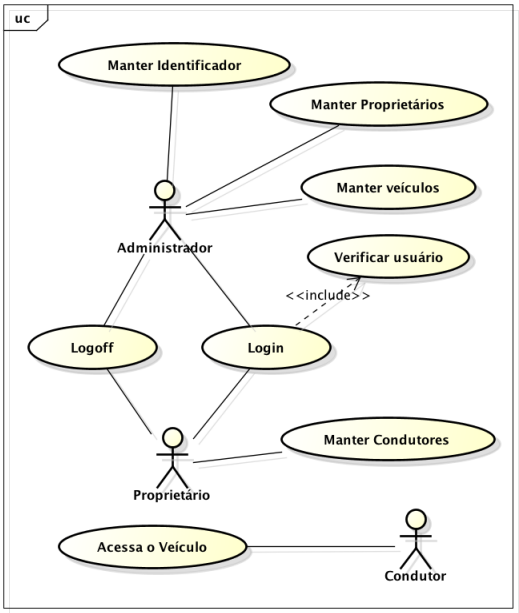

Figura 3.2: Casos de uso.

#	**Descrição**
Caso de Uso	Manter veículo.
Descrição	O administrador mantém o cadastro de veículos.
Atores	Administrador.
Fluxo principal	1. O Administrador clica em cadastro de veículo. 2. Acessa o formuário de cadastro e o preenche. 3. Confirma o cadastro clicando no botão salvar. 4. Exibe mensagem de êxito.
Fluxo de exceção	3.1 Preenche um veículo já existente. 3.2 O sistema apresenta mensagem de erro.
Cenário alternativo	1 Administrador quer editar um veículo. 2 O administrador entra no formuário de busca. 3 Digita o CPF do proprietário. 4 Seleciona o veículo na lista. 5 O sistema exibe o formulário de edicão. 6 O administrador efetua as edições necessárias. 7 Clica no botão confirmar. 8 O sistema exibe mensagem de êxito.

Tabela 3.3: Mantendo os dados dos veículos.

#	**Descrição**
Caso de Uso	Manter identificador.
Descrição	O administrador mantém o cadastro dos identificadores.
Atores	Administrador.
Fluxo principal	1. O Administrador clica em cadastro de identificador. 2. Acessa o formuário de cadastro e o preenche. 3. Confirma o cadastro clicando no botão salvar. 4. Exibe mensagem de êxito.
Fluxo de exceção	3.1 Preenche um veículo já existente. 3.2 O sistema apresenta mensagem de erro.

Tabela 3.4: Cadastrando identificadores.

#	**Descrição**
Caso de Uso	Mantém proprietários.
Descrição	O administrador mantém o cadastro de propritários.
Atores	Administrador.
Fluxo principal	1. O Administrador clica em cadastro de condutor. 2. Acessa o formuário de cadastro e o preenche. 3. Confirma o cadastro clicando no botão salvar. 4. Exibe mensagem de êxito. 5. Clica em atribuir categoria. 6. Atribui uma categoria e confirma. 7. Exibe mensagem de êxito. 8. Clica em atribuir veículo. 9. Atribui e confirma. 10. Exibe mensagem de êxito.
Fluxo de exceção	3.1 O proprietário já existente. 3.2 O sistema apresenta mensagem de erro.
Fluxo alternativo	1. O administrador edita o proprietário. 2. Clica em buscar veículo. 3. Digita a placa do veículo do proprietário. 4. Escolhe o proprietário. 5. Edita o formuário. 6. Confima as alterações. 7. O sistema apresenta mensagem de êxito.

Tabela 3.5: Manter proprietários.

#	**Descrição**
Caso de Uso	Acesso ao veículo.
Descrição	Um condutor com permissão deve ser capaz de acessar/ligar o veículo.
Atores	Condutor.
Fluxo principal	1. O Condutor passa o identificador, via Bluetooth, para o subsistema de controle veicular. 2. O subsistema de controle veicular, faz as verificações necessárias. 3. Em caso de liberado, o sistema mostra mensagem de OK no display e libera o veículo para uso.
Fluxo de exceção	3.1 O condutor identificado não está liberado. 3.2 O subsistema mostra no display, que o usuário não está liberado.
Fluxo de exceção	3.1 O condutor identificado não foi encontrado no banco local. 3.2 O subsistema busca as informações no subsistema de gerenciamento de usuários. 3.3 O subsistema atualiza a sua base de dados. 3.4 O subsistema retorna ao item 2 do fluxo principal.

Tabela 3.6: Manter proprietários.

3.3 Visão de classes e diagramas de sequência do sistema

A partir dos casos de usos levantados foram identificadas classes do sistema, as interligações entre estas classes, na forma de diagramas de classes e a interação entre objetos instanciados a partir destas classes, na forma de diagramas de sequência. A dinâmica de um caso de uso pode ser mapaeada em um ou mais diagrama de sequência. Nesta seção são apresentados alguns diagramas de classe e de sequência que são considerados mais relevantes para o entendimento do sistema. No anexo B todos diagramas são mostrados.

3.3.1 Diagramas de classe do subsistema veicular

O papel do subsistema veicular é basicamente receber um identificador do susbistema acesso e liberar o veículo caso o condutor asssociado com o identificador seja autorizado para uso do veículo.

A Figura 3.3 mostra um diagrama de classes do subsistema veicular. São as classes necessárias a implementação dos casos de uso associados a este subsistema. Na prática, o diagrama de classes (visão estrutural estática) é construído em paralelo com o diagrama de sequência (visão dinâmica), o qual é mostrado na Figura 3.5.

- Classe EnviaDados

  Classe pertencente ao pacote comunicação. Ela é responsável por fazer o post http para o servidor do subsistema de gerenciamento de usuários e contém as seguintes propriedades:

  url: Endereço do subsistema de gerenciamento de usuários;

  parametros: Dados a serem enviados;

  USER_AGENT: Propriedade estática que recebe qual navegador que o Java vai informar ao servidor.

  A classe tem apenas um método, *execute()* que envia o post para o servidor configurado.

- Classe Condutor

  Esta classe pertence ao pacote model e tem a função de gerenciar os dados do condutor, as propriedades da classe são:

  id: Identificador único;

  nome: Nome completo;

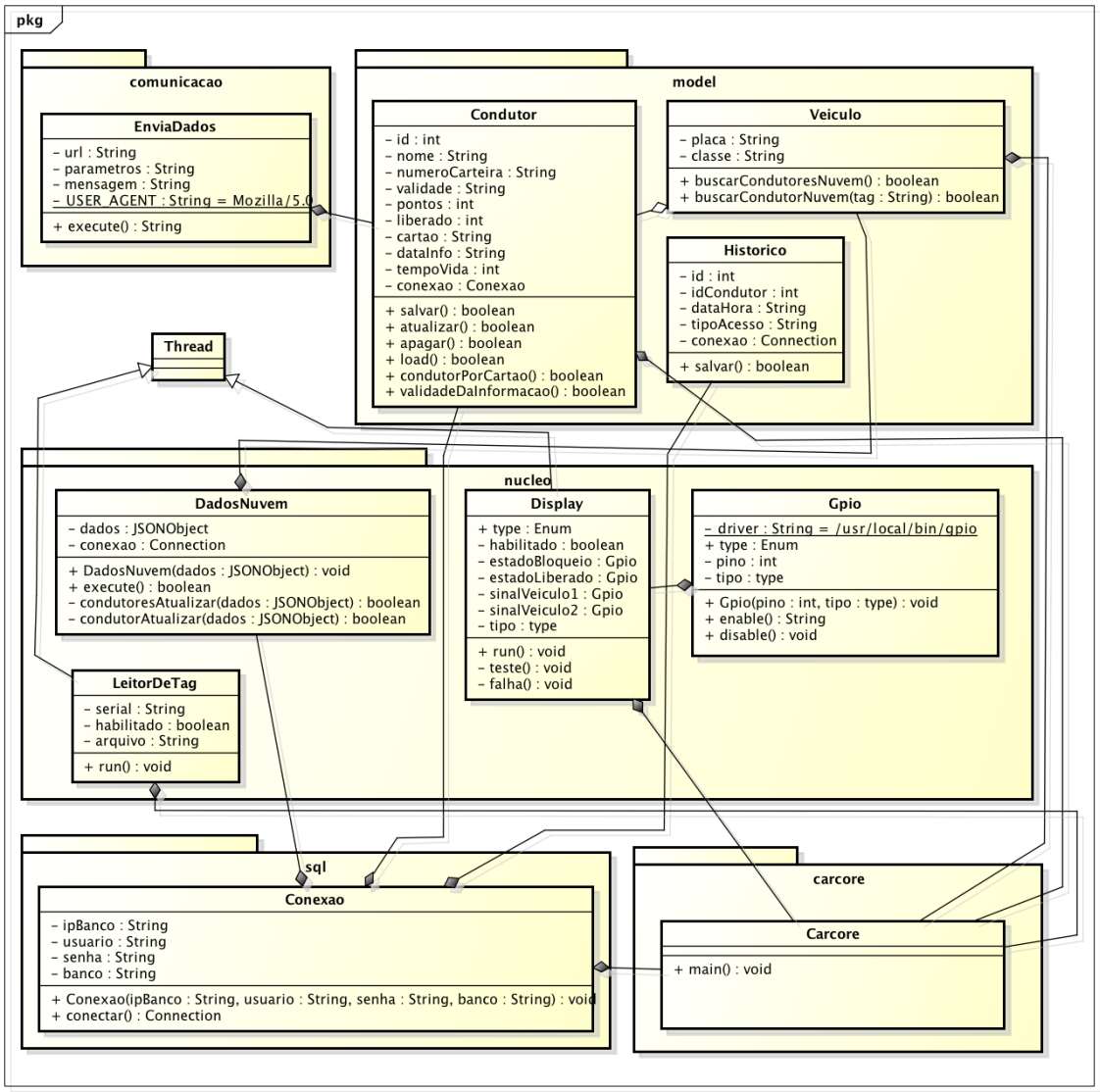

Figura 3.3: Diagrama de classe do Subsistema Veicular.

numeroCarteira: Número da carteira de motorista;

validade: Data de validade da carteira de motorista;

pontos: Pontos de infração;

liberado: Informa qual a situação do motorista, (1 para liberado);

cartao: Identificador do condutor;

dataInfo: Data do dia que foi baixada a informação do servidor;

tempoDeVida: Tempo máximo que essa informação é válida, sem atualizar;

conexao: Conexão com o banco de dados.

A classe conta também com 6 métodos públicos, são eles:

salvar: Salva os dados no banco de dados;

atualizar: Atualiza os dados já salvos no banco de dados;

apagar: Apaga o condutor do banco de dados;

load: Carrega o condutor a partir do banco de dados;

condutorPorCartao: Carrega o condutor do banco, apenas com o numero do identificador;

validadeDaInformacao: Verifica se a informação carregada, ainda está dentro do prazo de validade.

Todos os métodos retornam true ou false.

- Classe veiculo

Esta classe gerencia os dados do veículo e faz consultas no subsistema de gerenciamento de usuários. Eela possui apenas 2 propriedades:

placa: Placa do veículo;

classe: Categoria do veiculo (DETRAN - PR, 2014).

A classe contam também com 2 métodos públicos, são eles:

buscaCondutoresNuvem: Baixa todos os condutores do veículo no subsistema de gerenciamento de usuários;

buscaCondutorNuvem: Pega os dados de um condutor no subsistema de gerenciamento de usuários. Este método exige que informe o identificador referente.

- Classe Histórico

  Esta classe armazena todos os dados de acesso ao veículo, ela conta com 5 propriedades:

  id: Identificador único;

  idCondutor: Identificador do condutor;

  dataHora: Data e hora do ocorrido;

  tipoAcesso: Informação descritiva;

  conexao: Conexão com o banco de dados.

  A classe apresenta apenas um método, *salvar()*, que apenas inclui os dados no banco de dados.

- Classe DadosNuvem

  É a classe que pega a resposta do subsistema de gerencia de usuários, processa e armazena os dados no banco de dados. A classe conta com as seguintes propriedade.

  dados: Essa propriedade espera um JSONObject com um código e as informações a serem processadas;

  conexao: Conexão com o banco de dados.

  Esta classe contém 2 métodos públicos e 2 privados, são eles:

  DadosNuvem: O construtor da classe exige que seja passado os dados na criação do objeto;

  execute: É o método que processa as informações;

  condutoresAtualizar: É um método privado que pega um vetor de condutores e armazena no banco de dados;

  condutorAtualizar: Método privado que pega o condutor passado, e armazena no banco de dados.

- Classe Display

  A Classe Display gerencia as informações que serão apresentadas fisicamente ao condutor através de sinais luminosos, ela estende thread pois é executada independe do programa principal, ela tem 1 propriedade pública e 6 privadas, são elas:

  type: É um Enum que facilita na implementação, pré definindo as opções disponíveis;

habilitado: Em true mantem a thread em execução, quando alterado para false a thread é encerrada;

estadoBloqueio: Gerencia o pino 3 I/O do Raspberry, está configurado como saída, no protótipo este pino acende o LED vermelho;

estadoLiberado: Gerencia o pino 0 I/O do Raspberry, está configurado como saída, no protótipo este pino acende o LED verde;

sinalVeiculo1: Gerencia o pino 4 I/O do Raspberry, está configurado como saída, no protótipo este pino acende o LED verde;

sinalVeiculo2: Gerencia o pino 5 I/O do Raspberry, está configurado como saída, no protótipo não há uso.

Existe apena 1 método público que inicia a thread e 2 privados:

teste: Aciona de forma intermitente os pinos 3, 0, 4 e 5.

falha: Aciona de forma intermitente os pinos 3 e 0.

- Classe Gpio

Esta classe utiliza o driver encontrado em (GORDONS PROJECTS, 2013) para acionar os pinos I/O do Raspberry e seu funcionamento é através de 4 propriedades:

driver: É uma propriedade estática que contém o caminho do driver I/O;

type: Enum para facilitar a programação;

pino: Qual pino será configurado;

tipo: Informa se o pino é de entrada ou saída.

A classe conta com 3 métodos:

Gpio: Construtor que exige o pino e o tipo;

enable: Coloca um sinal de 3.3V na saída, quando configurado como *out* ou lê o pino quando configurado como *in*;

disable: Coloca o pino com 0V.

- Classe LeitorDeTag

Esta classe estende uma classe thread. No programa ela é executada de forma independente. As suas propriedades são:

serial: Propriedade que armazena a tag;

habilitado: Quando *true* mantém o processo em executando;

arquivo: Serve para informar o caminho do arquivo a ser lido.

A classe contém apenas um método *run()* para iniciar a execução da thread.

3.3.2 Diagramas de sequência do subsistema veicular

A seguir, na Figura 3.4, é mostrada uma visão geral da sequência de funcionamento do sistema, onde o condutor através do subsistema de acesso, solicita o envio do identificador, via bluetooth para o subsistema de controle veicular, que por sua vez, caso não ache o identificador na sua base de dados, busca as informações no subsistema de gerenciamento de usuarios.

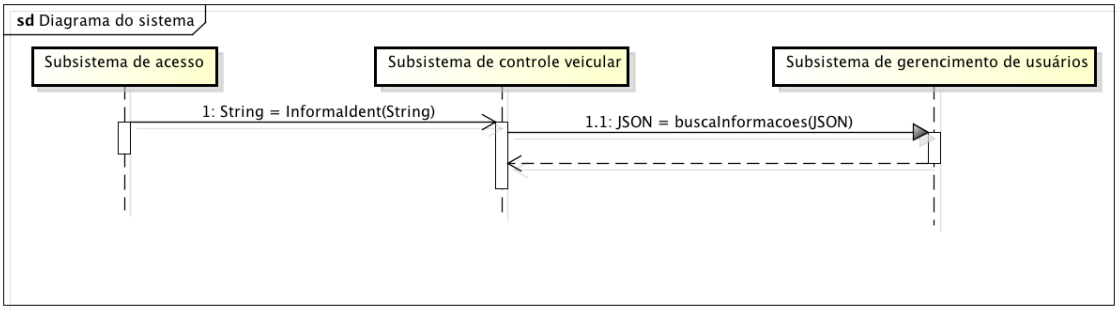

Figura 3.4: Diagrama de sequência do subsistema veicular, geral

Na Figura 3.5 é mostrado um diagrama de sequência com detalhamento dos objetos envolvidos instanciados a partir das classes mostradas anteriormente. O subsistema veicular interage com o celular (subsistema de acesso) através de uma interface Bluetooth. A classe responsável por esta comunicação é a class_LeitorDeTag. Ela é derivada de uma classe thread portanto se executa em pseudoparalelismo ao sistema. Esta thread fica em loop aguardando o envio do identificador via Bluetooth, ao receber a informação ela repassa para a thread principal, class_CarCore que verifica se o identificador é válido e está liberado. Após as devidas verificações a thread principal solicita a liberação do veículo para a thread class_Display, que sinaliza ao condutor e ao veículo a situação do condutor.

No anexo B são mostrados outros diagramas de sequência do subsistema veicular.

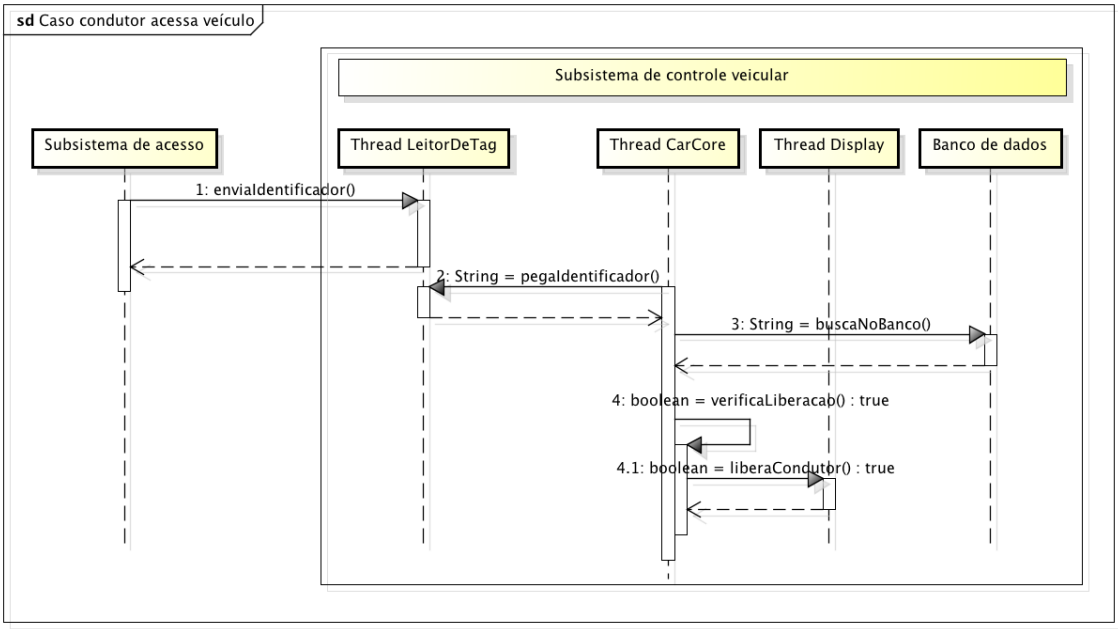

Figura 3.5: Diagrama de sequencia do Subsistema Veicular. Fonte: autor

3.3.3 Digrama de classe do subsistema de gerenciamento de usuários

A seguir é mostrado na Figura 3.6 os diagramas de classe do subsistema de gerenciamento de usuários.

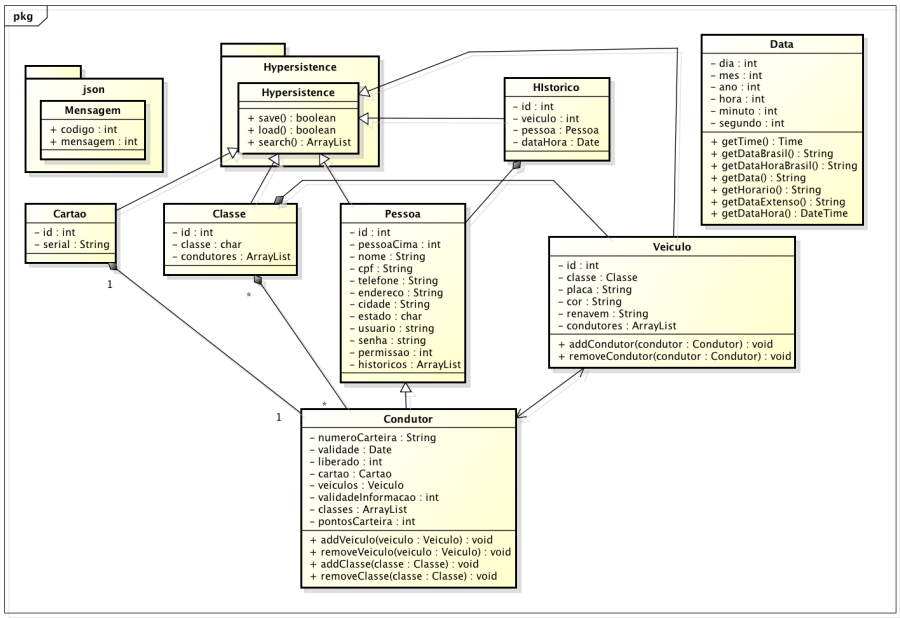

Figura 3.6: Diagrama de Classe

Pode-se identificar neste diagrama as seguintes classes:

- Mensagem: É a classe responsável por fazer a comunicação entre o módulo web e o

módulo do veículo, nela vai o código da mensagem e a mensagem a ser transmitida;

- **Hypersistence**: Classe que abstrai funcionalidade do banco de dados como, save, load, search (FORNARI, 2014b);
- **Data**: Classe com uma série de ferramentas para manipular datas;
- **Cartao**: A classe cartão armazena o numero da tag/serial de acesso, que será informado ao módulo do veículo, que irá fazer a consulta ao módulo web;
- **Classe**: Armazena os dados da categoria a qual o condutor e o veículo pertence (A, B, C, D ou E), é uma classe muitos para muitos, um classe pode ter muitos condutores assim como um condutor pode ter muitas classes e muitos para um, uma classe pode ter muitos veículos;
- **Historico**: A classe histórico serve para armazenar dados de acesso das pessoas que acessam o sistema, é uma classe muitos para um, isto é, uma pessoa pode ter muitos históricos;
- **Pessoa**: A classe pessoa é responsável por identificar, autenticar e dar os privilégios de acesso;
- **Condutor**: A classe condutor, estende pessoa, faz a identificação do condutor para o veiculo, ela é responsável por passar as informações necessárias para o módulo veicular;
- **Veiculo**: É a classe que identifica individualmente cada veículo, é uma classe muitos para muitos com o condutor, isto é, um veículo pode ter muitos condutores assim como um condutor pode ter muitos veículos.

3.3.4 Diagramas de sequência do subsistema veicular

Na Figura 3.7 é mostrado uma visão geral do funcionamento do subsistema de gerênciamento de usuários.

O usuário efetua o *login* no sistema, que por sua vez, faz uma verificação junto ao banco de dados, checando se o usuário e senha estão corretos, em caso se positivo, ele cria uma sessão entre o servidor e o nevegador do cliente, armazenando os dados do perfil do usuário.

Estando devidademnte autorizado, o usuário procede com o cadastro das informações que o sistema disponibiliza, de acordo com os privilégios do usuário e após os dados serem confirmados, o sistema armazena as informações em banco de dados.

O usuário pode a qualquer momento solicitar *logoff* do sistema destruindo a sessão entre o nevegador e o servidor, retornando a tela de *login*.

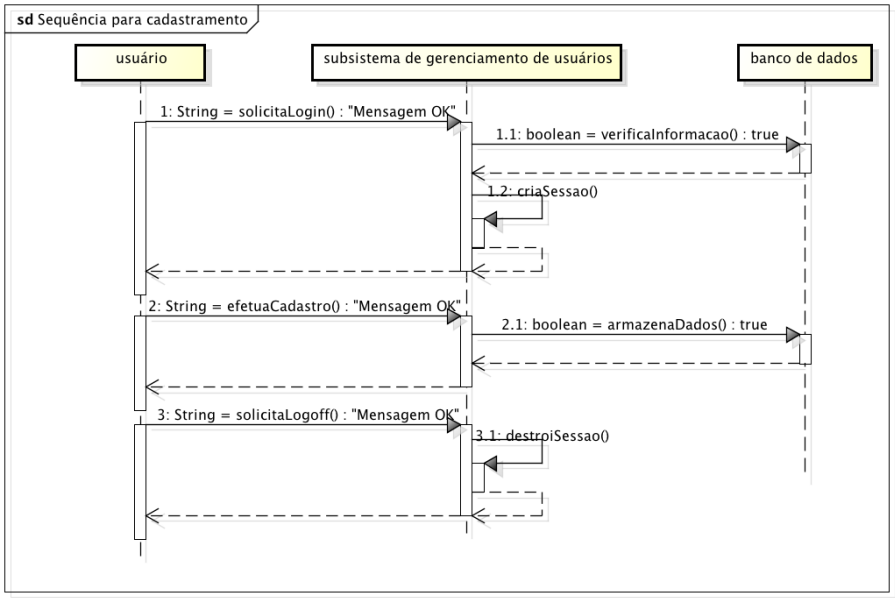

Figura 3.7: Diagrama de sequência do subsistema de gerenciamento de usuários.

3.3.5 Diagramas de atividades do subsistema de gerenciamento de usuários

Para um melhor planejamento do funcionamento do código escrito, foram desenhados alguns diagramas de atividade. Estes diagramas podem ser consultados no Anexo D.

3.4 Estrutura da Base de Dados

3.4.1 Diagrama de banco do subsistema de gerenciamento de usuários

Na Figura 3.8 é mostrado o diagrama do subsistema de gerenciamento de usuários. Na sequência será apresentado o detalhamento das tabelas que compõem o banco.

Pessoa

A tabela pessoa, armazena os dados de identificação e de acesso aos usuários, ela referencia ela mesma para criar sub-pessoas, como um cascata de pessoas, formando uma hierarquia de usuários. Esta tabela tem uma ligação de um para muitos com o histórico para armazenar os dados de acesso e uma ligação com o condutor de um para um.

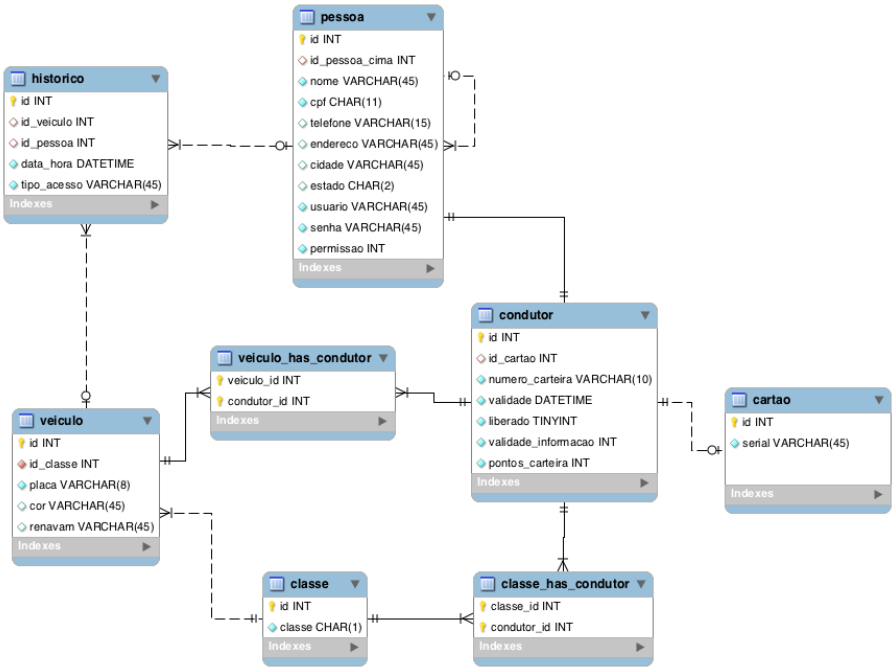

Figura 3.8: Diagrama de banco de dados do subsistema de gerenciamento.

Condutor

Esta tabela armazena os dados do condutor, para acesso do módulo veicular, esta tabela tem uma ligação com pessoa, um para um, uma ligação com o veículo de muitos para muitos, uma ligação com a tabela classe de muitos para muitos e uma ligação com o cartão de um para um onde, esta ligação pode ser nula.

Cartão

A tabela cartão, armazena os números de série dos indentificadores que serão utilizados no módulo veicular, esta tabela tem uma legação um para um com o condutor, onde esta ligação pode ser nula, isto é, um condutor pode não ter um identificador.

Classe

A tabela classe (categoria da carteira) tem valores fixados em A, B, C, D e E, esta tabela tem ligação muitos para muitos com a tabela condutor, e muitos para um com veiculo.

Histórico

Tabela que guarda as informações de acesso ao subsistema de gerenciamento de usuários.

Veículo

Esta tabela armazena os dados de todos os veículos cadastros, a tabela faz um relacionamento muitos para muito com o condutor e muitos para uma classe.

3.4.2 Diagrama de banco do subsistema veicular

Na Figura 3.9 é apresentado o banco de dados do subsistema veicular. Na sequência será descrito com detalhes as tabelas que compõem o banco de dados.

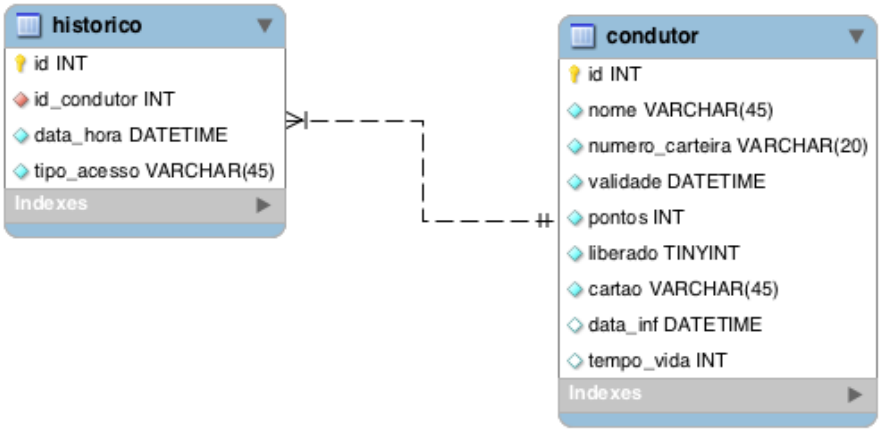

Figura 3.9: Diagrama de banco de dados do subsistema veicular. Fonte: autor

Condutor

Esta tabela armazena os dados dos condutores, habilitados ou não, para o veículo em específico, uma característica que difere esta tabela **condutor** para a tabela **condutor** do **subsistema de gerenciamento de usuários**, são os campos adicionais:

- **data_inf** que guarda a data que foi pega a informação do **subsistema de gerenciamento de usuários**.
- **tempoDeVida** que armazena o tempo máximo que esta informação pode ficar sem ser atualizada.

Histórico

Tabela que guarda as informações de acesso ao subsistema veicular.

3.5 Conclusões

Neste capítulo foi realizada uma visão geral do sistema proposto e do projeto realizado para sua implementação. Foram escolhidos alguns diagramas para ilustrar o projeto sem a intenção de abranger todo o desenvolvimento realizado. O projeto do subsistema do dispositivo portátil de identificação não foi explorado porque a sua implementação foi simplificada, dado o tempo restrito para a finalização do trabalho. No capítulo seguinte será mostrada a implementação do sistema.

Deve-se salientar que o sistema foi pensado para que seja acrescentado de forma fácil novas funcionalidade, por este motivo foi colocado em prática as melhores práticas de programação, tomando o cuidado de manter o código organizado e modular.

4 *Implementação e Teste do Sistema*

Neste capítulo são apresentados alguns aspectos da implementação do sistema proposto. Inicialmente é explorado o subsistema de acesso (dispositivo portátil de identificação). Na sequência é apresentada a implementação do subsistema de gerenciamento de usuários e depois é apresentado o subsistema veicular. Os testes são então apresentados e os resultados discutidos.

4.1 Subsistema de Acesso

No subsistema de acesso (celular) foi utilizado o App Inventor[1] para programar o software que envia o identificador para o subsistema de controle veicular. Conforme discutido no Capítulo 2, o App Inventor se utiliza de uma linguagem em blocos gráficos que permite o rápido desenvolvimento de uma aplicação Android. A Figura 4.1, mostra a interface da aplicação também desenvolvida com o App Inventor.

A implementação do subsistema de acesso se dá através dos scripts das Figuras 4.2, 4.3, 4.4, 4.5. Note-se que não houve um projeto formal para este subsistema dado as limitações de tempo e a simplicidade do mesmo.

A Figura 4.2 mostra o script que faz teste de conexão com o subsistema de controle veicular. Caso o botão **Conectar** seja pressionado, a aplicação tenta parear com o dispositivo. Em caso de sucesso, a aplicação se conecta e mostra mensagem de **conectado**. Caso contrário mostra a mensagem de **erro**.

Na Figura 4.3 é mostrado o script que envia o identificador para o subsistema de controle veicular. Ao pressionar o botão **Autorizar**, a aplicação tenta parear com o subsistema de controle veicular. Em caso de sucesso, ele tenta se conectar, verificando se a conexão foi efetuada com sucesso. Caso não consiga mostra uma mensagem de **erro** e, em caso positivo, mostra a mensagem de **conectado** e verifica se tudo ocorreu certo com a conexão. Se houve sucesso na operação mostra a mensagem de pareado e envia o identificador, em caso de erro mostra a

[1] http://tinyurl.com/mvpovp8

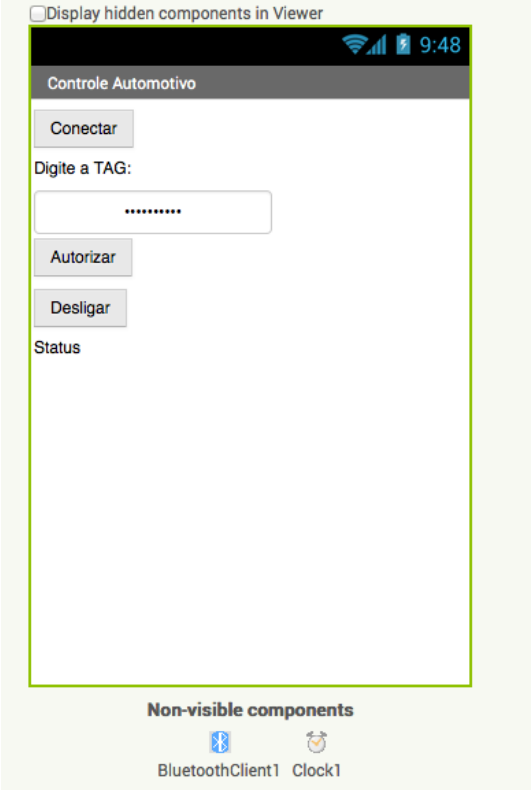

Figura 4.1: Diagrama do app inventor, tela do programa.

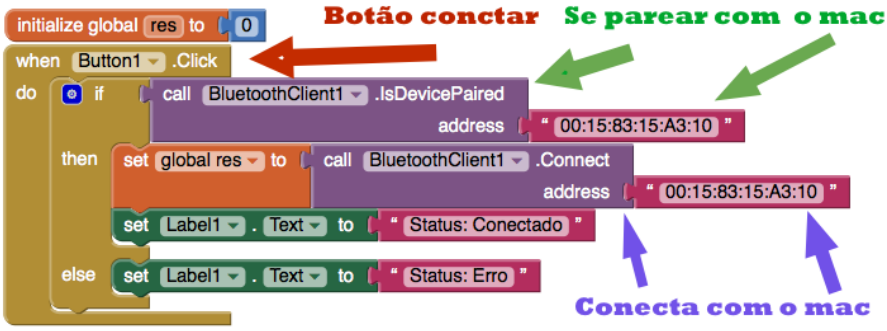

Figura 4.2: Diagrama do app inventor, bloco 1.

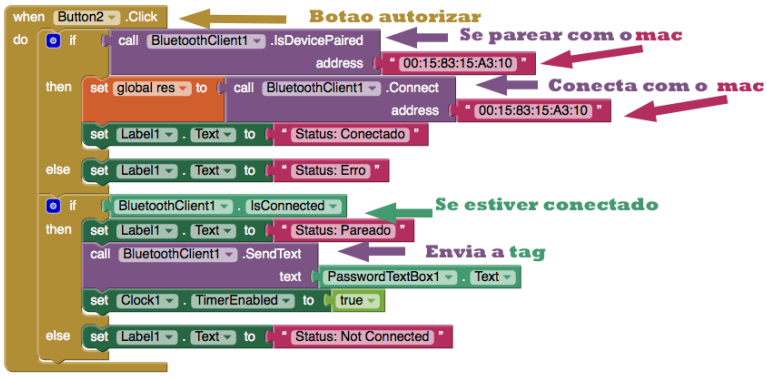

Figura 4.3: Diagrama do app inventor, bloco 2.

mensagem de **Not Connected**.

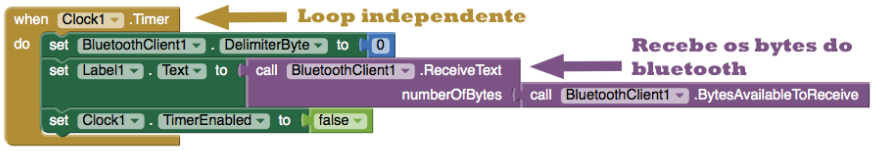

Figura 4.4: Diagrama do app inventor, bloco 3.

Os blocos da Figura 4.4 mostram um componente que faz um loop independente. Este loop faz a leitura dos dados que chegam através do bluetooth, e mostra no *label* **status**, os dados que chegam.

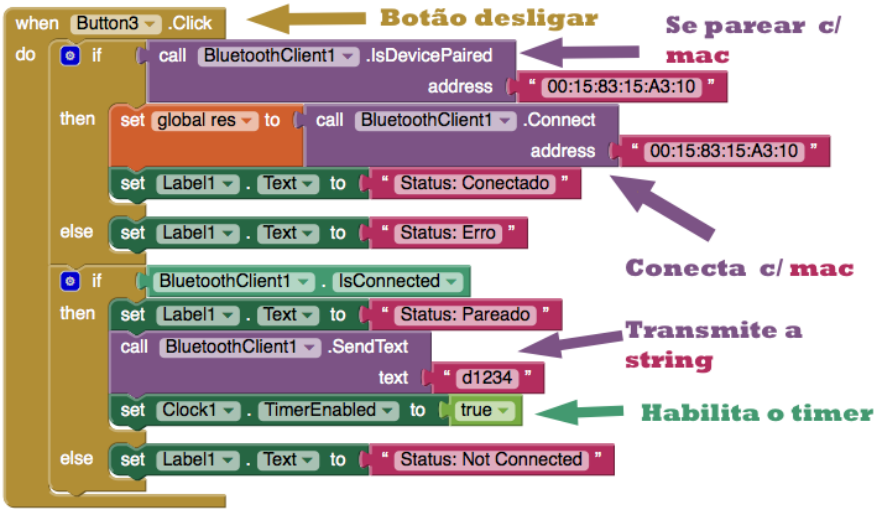

Figura 4.5: Diagrama do app inventor, bloco 4.

A Figura 4.5 mostra o *script* responsável por enviar a mensagem de desligamento do subsistema de controle veicular. Ele é praticamente o mesmo script da Figura 4.3, com a diferença de que ao invés de enviar o identificador, envia uma *string* com um código específico para esta função.

O subsistema está disponível no link http://tinyurl.com/mvpovp8 ao através do *QR-Code* 4.6.

Figura 4.6: QR-Code da aplicação.

4.2 Implementação do subsistema de gerenciamento de usuários

Este subsistema funciona em plataforma Linux e seu hardware pode ser qualquer computador X86 ou X64, tornando simples a sua implantação. Todo o subsistema de Gerenciamento de Usuários foi desenvolvido em PHP seguindo o projeto apresentado parcialmente no Capítulo 3. O código PHP foi depositado em um repositório *GNU Interactive Tools (GIT)*. O programa PHP é acionado por um servidor Web, no caso o Apache. Toda a comunicação com o susistema de gerenciamento de usuário é realizada através de método POST do HTTP, seguindo o modelo web tradicional. Os detalhes da implantação são mostrados a seguir.

4.2.1 Sistema operacional

O sistema operacional usado, foi o Debian 7, disponível no site oficial *https://www.debian.org*. A versão de instalação escolhida foi a pequena, possibilitando a escolha da menor quantidade de pacotes possível e sem interface gráfica, para reduzir ao máximo os recursos necessários.

4.2.2 Programas instalados

Os programas necessários para o funcionamento do subsistema são:

- Apache 2.
- PHP 5.
- MySQL.
- PhpMyAdmin

- Git.

Um pequeno tutorial mostrando a instalação das aplicações, é apresentado no Anexo E.3

4.3 Testes de integração do sistema

Os testes foram efetuados em ambiente controlado. O celular usado foi um Motorola Moto G de primeira geração com Android 4.4.4. O Raspberry Pi utilizado foi o padrão *model A* ligado a rede por cabo. O subsistema de gerenciamento de usuários foi colocado em uma máquina virtual na mesma faixa de IP do Raspberry Pi. A estrutura pode ser vista na Figura 4.7. Foi aguardado que todos os sistemas fossem carregados por completo. A ordem de inicialização não influência no funcionamento.

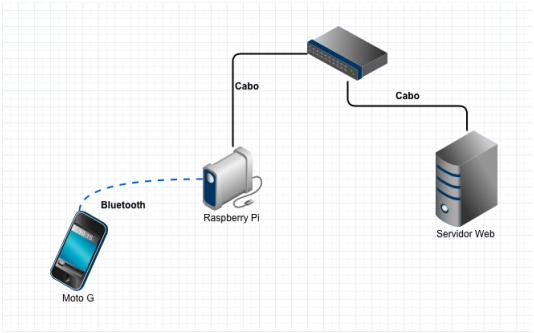

Figura 4.7: Estrutura de testes.

4.3.1 Teste de conectividade com o bluetooth

Este teste foi efetuado, seguindo os passos necessários para parear com o subsistema de controle veicular e fazer a primeira comunicação via bluetooth.

1. Buscando dispositivos: O subsistema apareceu como a identificação, a placa do veículo, portanto funcionou corretamente, como mostra a Figura no anexo G.1;

2. Pareando: Ao parear o dispositivo solicitou o PIN para o pareamento, depois de colocar o PIN correto ele pareou normalmente, como mostra a Figura no anexo G.2.

O próximo teste é feito no programa desenvolvido no App Inventor.

1. Abrindo o programa: Ao abrir o programa ele apresenta uma mensagem de erro, **Error 515: Not connected to a Bluetooth device**. Não foi explorado a causa deste erro, pois

não influência no uso, apenas atrapalha a sua usabilidade inicial, como mostra a Figura no anexo G.3;

2. Clicando em Conectar: Ao clicar em conectar o dispositivo demorou ”um pouco para”responder, mas apresentou a mensagem de conectado corretamente, como mostra a Figura no anexo G.4;

3. Clicando em Autorizar: Após clicar no botão, o programa enviou corretamente o identificador até o subsistema de controle veicular, como mostra a Figura no anexo G.5.

4. Clicando em Desligar: Ao clicar em desligar, tudo ocorreu conforme o esperado, exatamente como o item anterior;

4.3.2 Testes de comunicação do subsistema de controle veicular

Os testes aqui apresentados, são da perspectiva do subsistema de controle veicular, no qual faz o entreposto entre o subsistema de acesso e o subsistema de gerenciamento de usuários.

O teste a seguir foi efetuado com o banco de dados local vazio. É enviado o identificador via bluetooth para este subsistema que irá procurar os dados no subsistema de gerenciamento de usuários, que por sua vez irá devolver a informação que será cadastrada no banco de dados local.

1. Conexão e recebimento do identificador: Neste momento não houve nenhum problema de conexão e recebimento do identificador como mostra a Figura 4.8;

2. Leitura e busca do identificador: A leitura do identificador foi efetuada com sucesso. Como o identificador não se encontrava no banco local, o sistema fez uma busca no subsistema de gerenciamento de usuários, através de um **post http**, com as informações necessárias no formato **JSON** (JSON, 2015);

3. Resposta do subsistema de gerenciamento de usuários: A resposta do subsistema veio de forma correta. O seu conteúdo também veio no formato correto, como mostra a Figura 4.8;

4. Inserção dos dados no banco local: A inserção foi feita de forma correta sem apresentar erros.

5. Verificação do condutor: Nesta etapa ocorreu um erro, o sistema não verificou novamente se o condutor estava habilitado, deixando o veiculo bloqueado para uso, sendo necessário enviar o identificador novamente;

```
Preparando para leitura..          Iniciando os drivers bluetooth
accepted connection from 34:BB:26:25:FD:ED          Conexão bluetooth efetuada
received [123456789]          Recebido o identificador
Lido: 123456789          Lido pelo subsistema
Procurando condutor
Não encontrado o cartao.          Identificador não encontrado no banco local

Enviando POST para a url: http://192.168.254.114/json          Enviado ao servidor um post http
Paramentros do post: dados={"codigo":1,"classe":"B","placa":"LWX-7546"}          Com o conteúdo
Resposta do servidor: 200          Resposta OK          Conteúdo
{"codigo":200,"mensagem":[{"nome":"Thiago Jos\u00e9","nCarteira":"622588158","validade":"2017-07-09 00:00:00","ponto
s":"0","id":"1","liberado":"1","cartao":"123456789","tempoVida":"10"},{"nome":"Diego Silveira","nCarteira":"62258815
7","validade":"2018-07-09 00:00:00","pontos":"0","id":"4","liberado":"1","cartao":"87654321","tempoVida":"10"}]}
```

Figura 4.8: Leitura e busca do identificador.

O próximo teste foi efetuado com o condutor já cadastrado no banco local.

Como é possível verificar na Figura 4.9, a leitura e a verificação do identificador no banco, foram feitas de forma correta e funcionaram perfeitamente. As luzes indicativas também funcionaram de forma correta, como mostra a Figura 4.10.

```
Preparando para leitura..
accepted connection from 34:BB:26:25:FD:ED
received [123456789]
Lido: 123456789
Procurando condutor
Dias de validade: 0
Condutor Habilitado
Veiculo Habilitado = ligado
```

Figura 4.9: Leitura e busca do identificador no banco local.

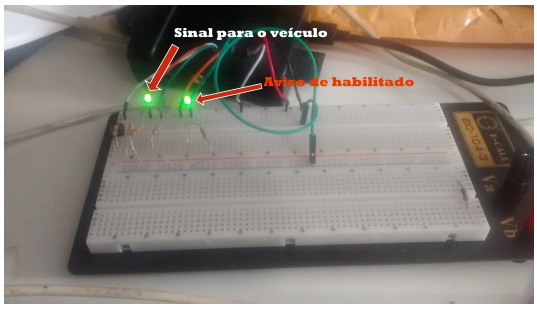

Figura 4.10: Display, indicador de habilitado.

No próximo teste foi verificado o comportamento do sistema, para um condutor não habilitado.

Como é possível ver na Figura 4.11, o sistema fez a leitura do identificador e a verificação no banco de forma correta e informou através das luzes indicadoras, que o condutor não está habilitado, como mostra a Figura 4.12.

O próximo teste é no mecanismo que efetua o desligamento do veículo, através de um identificador reservado. Neste caso foi utilizado o identificador **d1234**, como mostra a Figura 4.13. Tudo ocorreu conforma esperado e o sinal mostrado na foto 4.10 foi desativado.

```
Preparando para leitura..
accepted connection from 34:BB:26:25:FD:ED
received [123456789]
Lido: 123456789
Procurando condutor
Dias de validade: 0
Condutor Desabilitado
Veiculo Habilitado = ligado
Veiculo desligado
```

Figura 4.11: Condutor não habilitado.

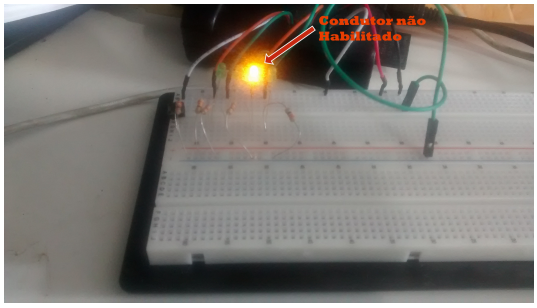

Figura 4.12: Display, indicador de não habilitado.

```
Preparando para leitura..
accepted connection from 34:BB:26:25:FD:ED
received [d1234]
Lido: d1234
Veiculo desligado
Aguardando
```

Figura 4.13: Desligando o veículo através do bluetooth.

Por último, foi efetuado o teste de quando o veículo recebe o sinal proveniente do veículo, neste caso simulado por um botão, como mostra a foto da Figura 4.15, solicitando o desligamento. Como é possível ver na Figura 4.14 tudo ocorreu conforme o esperado.

```
/usr/local/bin/gpio mode 8 out
/usr/local/bin/gpio read 8
lido: 1
Entrada = 1        <- Sinal = 1
 = 1
/usr/local/bin/gpio mode 8 out
/usr/local/bin/gpio read 8
lido: 0
Entrada = 0        <- Sinal = 0
Veiculo desligado
Aguardando
```

Figura 4.14: Desligando o veículo através de sinal externo.

Problema conhecido

Por algum motivo desconhecido ainda, quando se faz a leitura de um identificador bloqueado e logo após se faz uma leitura de um identificador liberado, o sistema para de ler os identificadores, problema este que não ocorre, quando se lê apenas identificadores liberados.

Figura 4.15: Desligando o veículo através de sinal externo, display.

4.4 Conclusões

No subsistema de controle veicular, o Bluetooth funcionou da forma esperada, mesmo havendo a necessidade incorporar o código ao programa principal, não houve travamentos ou congelamentos durantes os testes e é possível implementar outros recursos, como display de LCD, avisos sonoros e outras funcionalidades.

O subsistema de acesso requer o desenvolvimento de uma aplicação dedicada a função, para deixá-lo com mais funcionalidades.

O subsistema de controle veicular, pode ser aprimorado na leitura dos identificadores via bluetooth, deixando de utilizar o programa feito em C para fazer esta tarefa. Também é possível utilizar a biblioteca Java **pi4j** (PI4J.COM/, 2015), para manipular diretamente as entradas e saídas da placa, melhorando assim a performance e recursos disponíveis.

5 *Conclusões*

Neste capítulo são apresentadas as conclusões do trabalho. Inicialmente é feita uma análise sobre os objetivos do trabalho. Posteriormente são apresentadas as conclusões finais e perspectivas de trabalhos futuros.

5.1 Análise de objetivos do trabalho

- O gerenciamento feito através da aplicação web, funcionou de forma satisfatória, sem erro, problemas ou inconsistência em seu funcionamento. É necessário neste sistema fazer ajustes nas validações das informações inseridas.
- O objetivo de configurar um sistema portátil, foi cumprido parcialmente, infelizmente foi possível apenas implementar a função de enviar identificadores.
- Infelizmente não foi possível implementar uma interface que permita a autoridade de trânsito, interagir e obter informações.
- Não foi possível também implementar um mecanismo de detecção de intrusão, apesar de o sistema está preparado para faze-lo.

5.2 Conclusões finais

Os objetivos iniciais do trabalho foram atingidos. O conceito de uma plataforma de controle automotivo se mostrou muito interessante, devido a grande versatilidade dos módulos é possível implementar inúmeras funcionalidades.

Os testes mostraram uma certa dificuldade no uso do bluetooth, com o Raspberry Pi, seria interessante verificar a possibilidade do uso de outras tecnologias de acesso, como: **RFID** e **Smart Card**.

A plataforma se mostrou bastante promissora. Para um protótipo totalmente funcional, é necessário apenas fazer alguns ajustes no subsistema controle veicular, desenvolver uma aplicação mais amigável para celulares e implementar o uso da rede celular, para a comunicação com a internet.

No futuro seria possível através da experiência adquirida com o com esta plataforma, criar um sistema nacional de controle de condutores, no qual seria possível o governo criar carteiras de motoristas digitais. Abrindo possibilidade para que desta forma, seja possível fazer um controle mais efetivo, inibindo que condutores com carteira de motorista vencida ou suspensa, possam conduzir seus veículos.

5.2.1 Trabalhos futuros

São vários os trabalhos que podem se seguir, alguns deles são:

1. Desenvolver e analisar outras formas de identificação, como **RFID** e **Smart Card**;
2. Utilizar o bluetooth não somente para envio do identificador, mas também para recebimento de dados do veículo, como: Dados do cadastro, velocidade, rastreio, informações de uso entre outros;
3. Utilizar a Bluetooth BLE, um dos objetivos iniciais que foi abandonado em função do tempo dispoível;
4. Em uma ideia mais ampla, é possível usar este sistema para controle governamental, podendo através do uso do bluetooth, o agente de trânsito se conectar ao sistema automotivo e puxar os dados do veículo e condutor. Também é possível fazer com que a plataforma se comunique com o DETRAN, possibilitando o bloqueio do acesso ao condutor, caso sua carteira esteja suspensa.

ANEXO A – Casos de uso

A seguir serão apresentados os casos de uso que não foram inclusos no texto principal.

Na Tabela A.1 é apresentado o caso de uso do subsistema de gerenciamento de usuários, no qual o proprietário mantém os condutores.

Na Tabela A.2 é apresentado o caso de uso do sistema de gerenciamento de usuários, no qual o usuário solicita a desconexão com o subsistema.

#	**Descrição**
Caso de Uso	Manter condutores.
Descrição	O proprietário mantém os condutores.
Atores	Proprietário.
Fluxo principal	1. O proprietário clica em cadastro de condutor. 2. Acessa o formulário de cadastro e o preenche. 3. Confirma o cadastro clicando no botão salvar. 4. O sistema coloca no campo pessoa acima o usuário que está efetuando o cadastro. 5. Exibe mensagem de êxito. 6. Clica em atribuir categoria. 7. Atribui uma categoria e confirma. 8. Exibe mensagem de êxito. 9. Clica em atribuir veículo. 10. Atribui e confirma. 11. Exibe mensagem de êxito.
Fluxo de exceção	3.1 O proprietário já existente. 3.2 O sistema apresenta mensagem de erro.
Fluxo alternativo	1. O proprietário edita o condutor. 2. Clica em buscar veículo. 3. Digita a placa do veículo do proprietário. 4. O sistema exibe apenas os condutores abaixo do proprietário que está efetuando a busca. 5. Escolhe o proprietário. 6. Edita o formulário. 7. Confirma as alterações. 8. O sistema apresenta mensagem de êxito.

Tabela A.1: Manter condutores.

#	**Descrição**
Caso de Uso	Logoff.
Descrição	O usuário solicita a desconexão do sistema.
Atores	Administrador, Proprietário.
Precondições	O usuário deve está *logado* para efetuar o *logoff*.
Fluxo principal	1. O usuário clica no botão sair. 2. O sistema apaga as sessões do servidor. 3. O sistema redireciona para a tela de login.

Tabela A.2: Efetuando login no subsistema de gerenciamento de usuários.

ANEXO B – Diagramas de sequência do Sistema

A seguir serão apresentados os diagramas de sequência, que não foram inclusos no texto principal.

O diagrama B.1, apresenta uma sequência na qual o identificador se encontra no banco de dados local do subsistema.

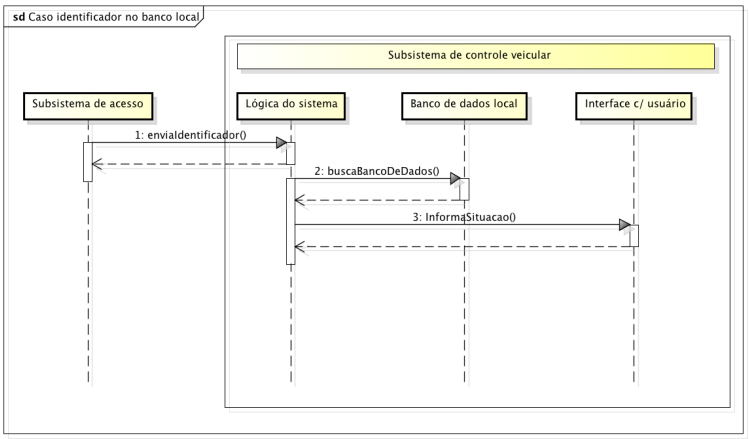

Figura B.1: Diagrama de sequência do sistema, caso identificador no banco.

Na Figura B.2 é apresentado a sequência executada, quando o identificado não se encontra no banco de dados local do subsistema.

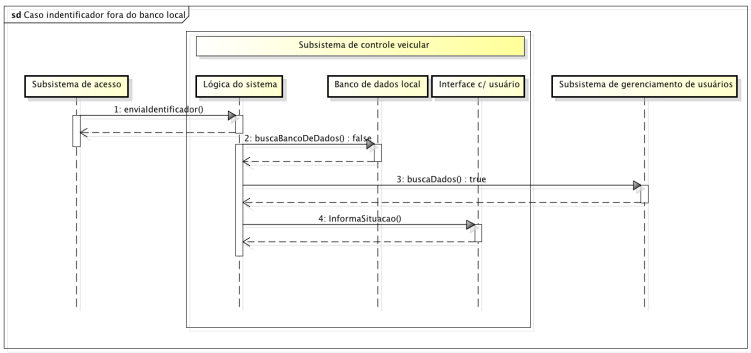

Figura B.2: Diagrama de sequência do sistema, caso identificador no banco externo.

Outros diagramas, testes e conceitos, podem ser vistos no endereço *http://tinyurl.com/ndc86j5*.

ANEXO C – Estrutura de arquivos do subsistema de gerenciamento de usuários

Os arquivos estão distribuídos como é mostrado C.1:

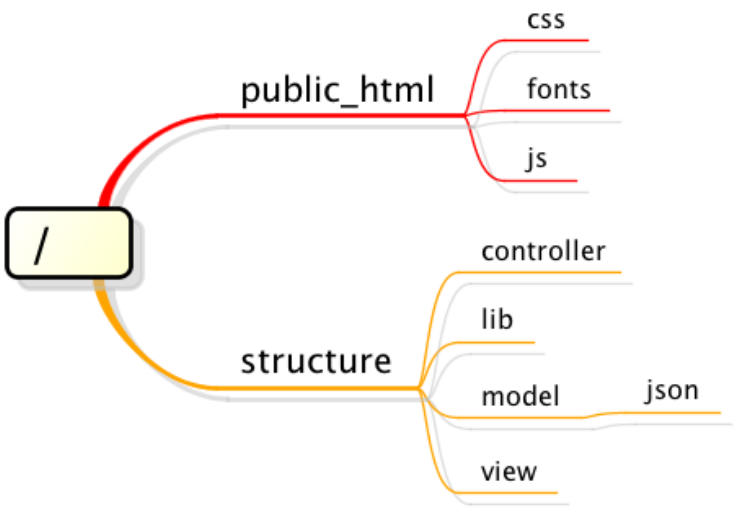

Figura C.1: Estrutura dos arquivos

1. public_html: Raiz dos arquivos públicos, que pedem ser acessados diretamente pela URL;

   css: Arquivos do CSS;

   fonts: Fontes customizadas do Bootstrap;

   js: Java script do Bootstrap e de outras funções do sistema;

2. structure: Estrutura do programa, essa parte é inacessível pelo navegador;

   controller: PHP responsável por controlar as views;

   lib: Bibliotecas do sistema;

   model: Classes do sistema;

   view: HTMLs do sistema.

ANEXO D – Diagramas de atividades do subsistema de gerenciamento de usuários

Estes diagramas facilitam a compreensão do funcionamento dos algoritmos presentes no subsistema.

D.1 Login

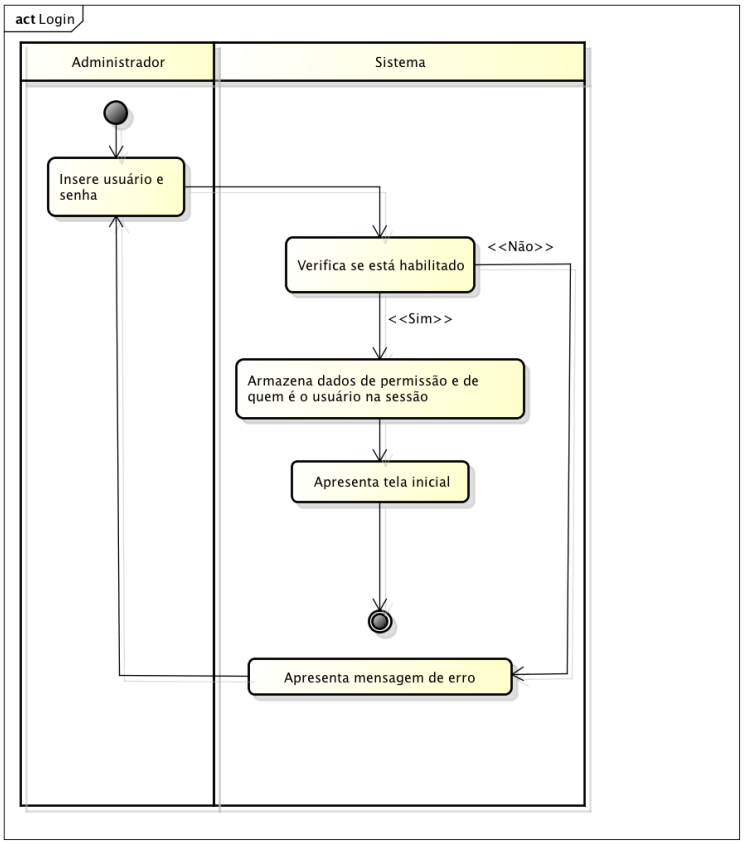

Figura D.1: Diagrama de Atividade - Login

O usuário insere o seu usuário e senha e clica no botão entrar, o sistema verifica se o usuário se senha estão corretos e se está habilitado, caso não, apresenta mensagem de erro, caso sim

armazena os dados de permissão na sessão e mostra a tela inicial do sistema, Figura D.1.

D.2 Cadastro de Veículos

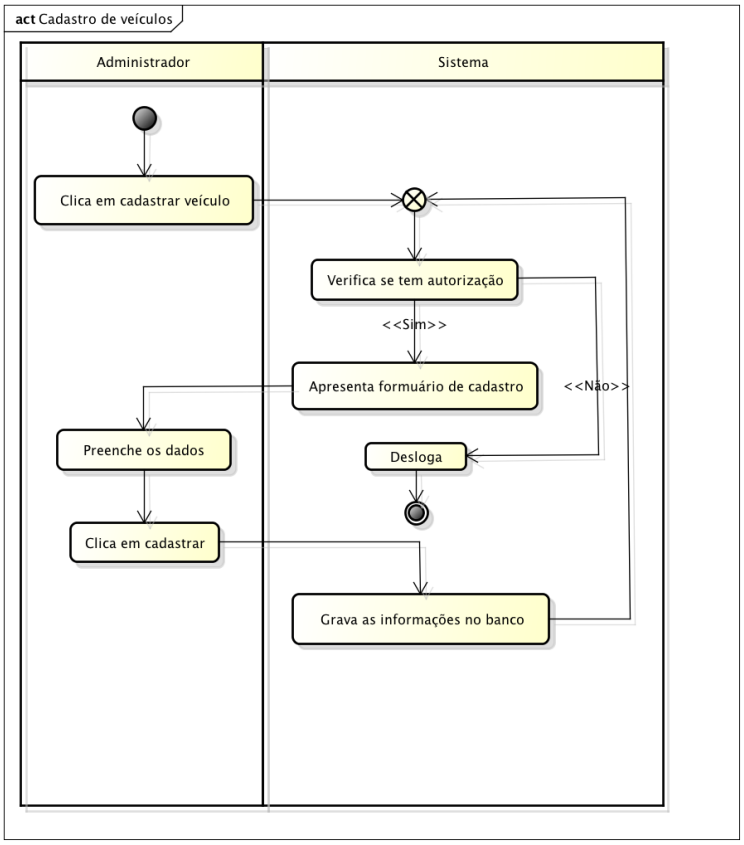

Figura D.2: Diagrama de Atividade - Cadastro de veículos

Ao clicar em cadastro de veículo, o sistema verifica se aquele usuário tem autorização para acessar essa página, caso não os sistema desloga o usuário, caso sim apresenta o formulário de cadastro de veículos, após preenchido corretamente o formulário, ao usuário clicar em cadastrar, o sistema pega as informações a armazena no banco de dados, Figura D.2.

D.3 Cadastro de Condutor

Ao acessar o link Cadastrar Condutor, o sistema verificará se o usuário está autorizado, caso não o sistema desloga o usuário, caso sim é apresentado o formulário de cadastro de condutor, após o formulário ser preenchido corretamente, o usuário deve clicar no botão cadastrar, o sistema verifica se o usuário é administrador, caso sim ele salva os dados no banco, caso não, ele adiciona o usuário que está efetuando o cadastro como usuário superior ao que está sendo

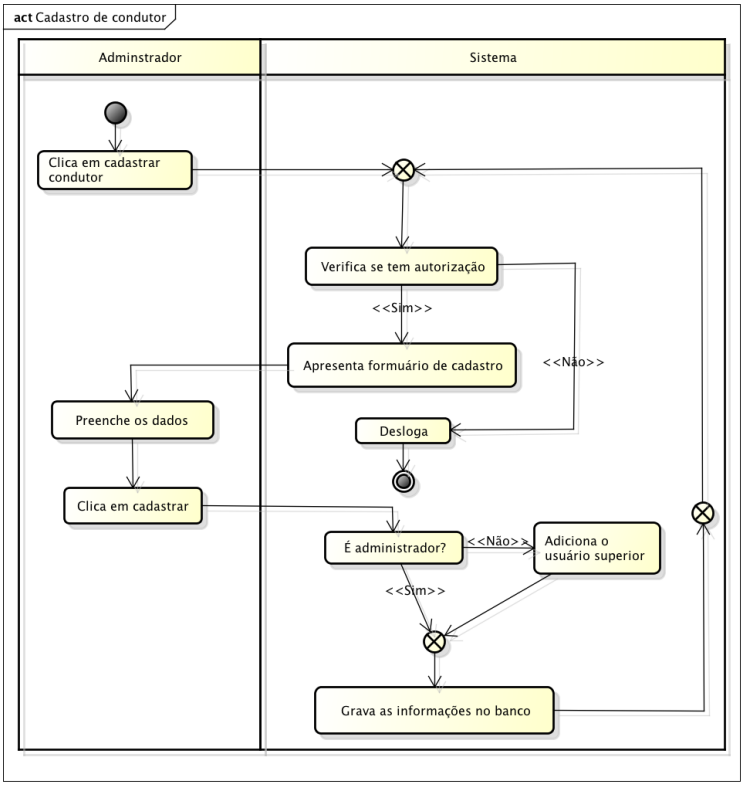

Figura D.3: Diagrama de Atividade - Cadastro de condutor

cadastrado, desta forma é atrelado quem é o condutor primário e o condutor secundário, após esse processo os dados são salvos em banco, Figura D.3.

D.4 Consultar dados do usuário

Ao clicar em consultar dados do usuário, o sistema verifica se o usuário tem autorização, caso não o usuário é deslogado, caso sim, apresenta o formulário de consulta, no qual digita o CPF, ao clicar em buscar o sistema consulta os dados no banco e exibe uma lista de veículos atribuída a esse usuário. Também é exibido as opções de retirar o veículo desse usuário e de editar o veículo, Figura D.4.

D.5 Atribuir classe ao usuário

Acessando a página de atribuição de classe ao usuário, o sistema pede o CPF do condutor e qual classe deseja adicionar, ao confirmar os dados, o sistema verifica se o usuário é administrador ou se é supervisor do usuário que está sendo alterado, caso sim, ele aceita o cadastro, grava as informações no banco de dados, caso não apresenta mensagem de erro e retorna ao inicio,

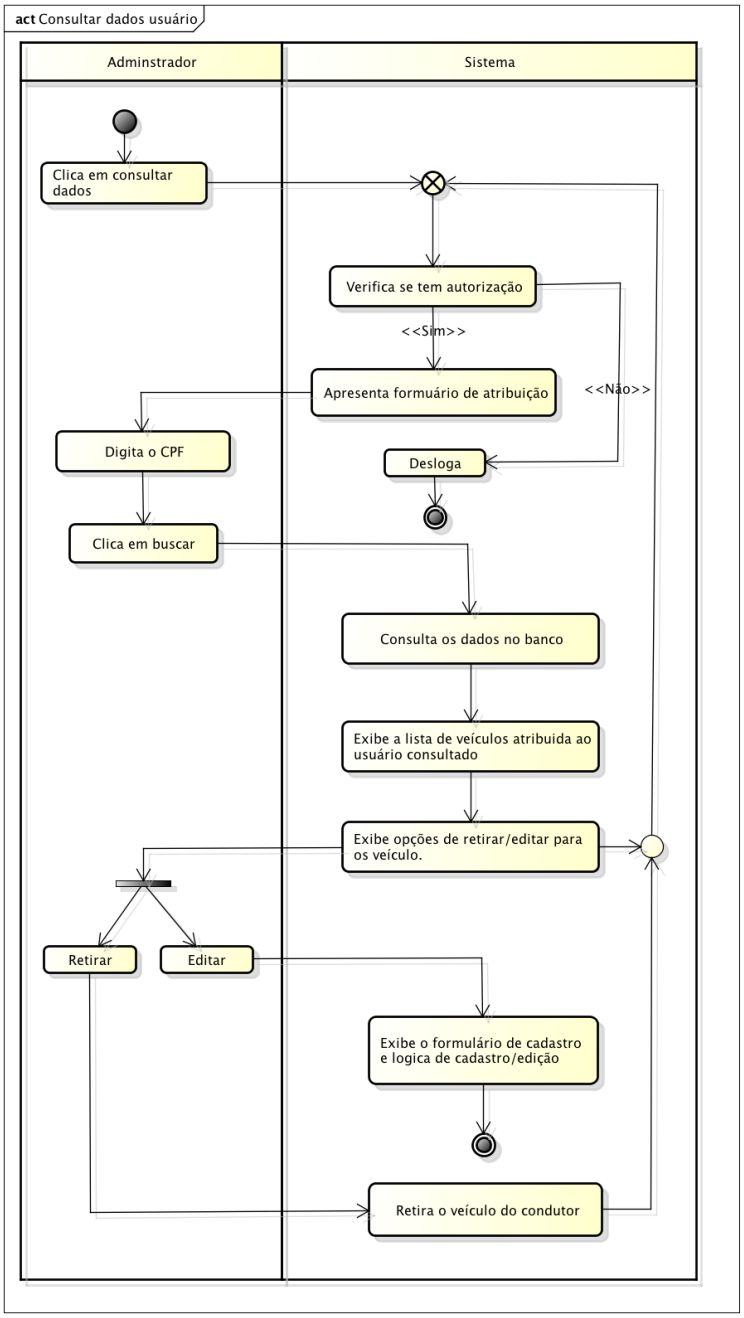

Figura D.4: Diagrama de Atividade - Consultar dados do usuário

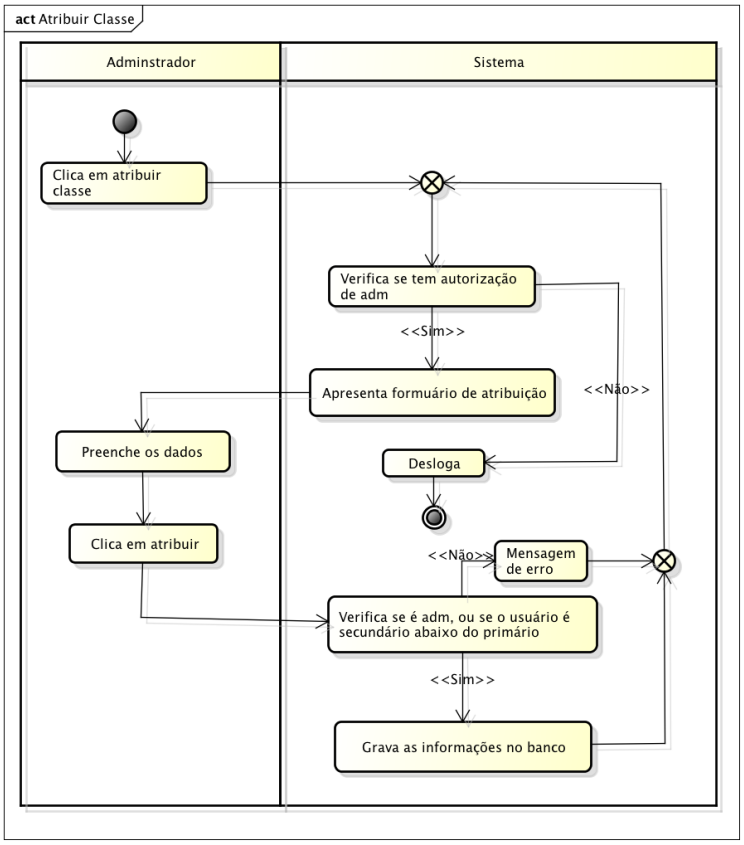

Figura D.5: Diagrama de Atividade - Atribuir classe ao usuário

Figura D.5.

D.6 Atribuir condutor a um veículo

Ao acessar o link de atribuição de condutor a veículo, o sistema verifica se está autorizado a fazer o procedimento, caso sim, é exibido o formulário de atribuição, após o usuário preencher as informações requeridas e clicar em atribuir, o sistema verifica se o usuário é administrador do sistema ou responsável pelo veículo a ser atribuído o condutor, caso sim, os dados são gravados no banco de dados, caso não uma mensagem de erro é apresentada e o programa volta para o inicio, Figura D.6.

D.7 Atribuir veículo a um condutor

Contém praticamente a mesma lógica que o Atribuir condutor a um veículo D.6, Figura D.7.

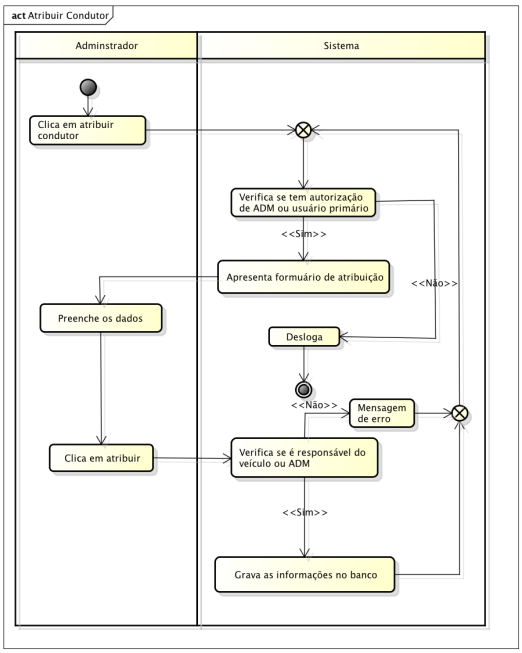

Figura D.6: Diagrama de Atividade - Atribuir condutor a um veículo

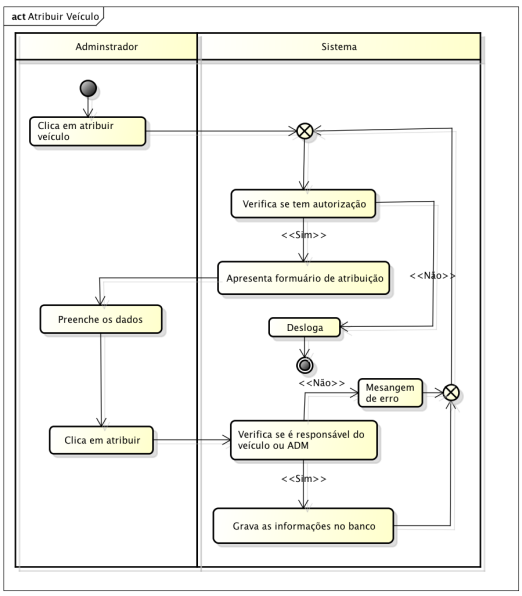

Figura D.7: Diagrama de Atividade - Atribuir veículo a um condutor

D.8 Cadastro do identificador

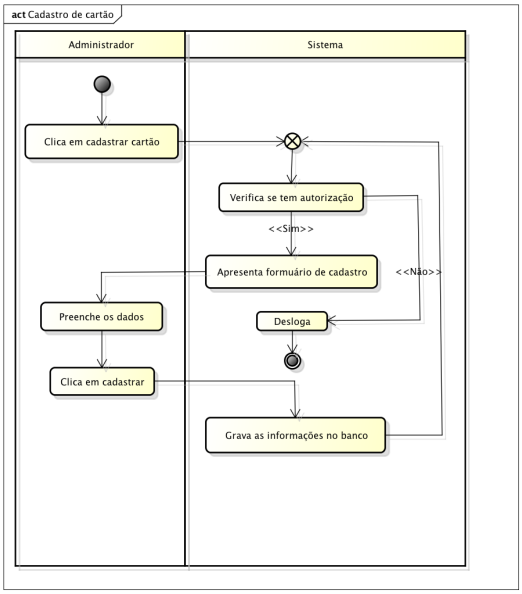

Figura D.8: Diagrama de Atividade - Atribuir veículo a um condutor

Ao cadastrar um novo identificador o sistema verifica se o usuário tem privilégios para isto e apresenta o formulário de cadastro, após clicar em cadastrar o sistema grava no banco de dados o numero de serie digitado Figura D.9.

D.9 Cadastro do identificador

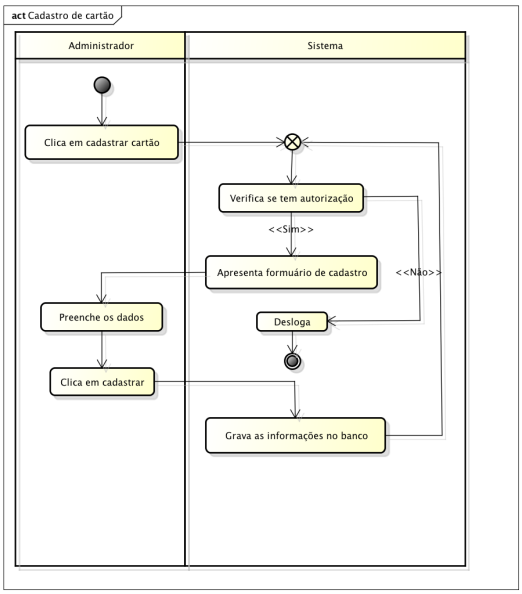

Figura D.9: Diagrama de Atividade - Atribuir veículo a um condutor

Ao cadastrar um novo identificador o sistema verifica se o usuário tem privilégios para isto e apresenta o formulário de cadastro, após clicar em cadastrar o sistema grava no banco de dados o numero de serie digitado Figura D.9.

ANEXO E – Instalando o firmware do Raspberry no SD card

Nesta seção, será descrito como instalar o Raspbian no SD card.

E.1 Formatando cartão SD

Para formatar o SD é necessário ter instalado o programa **SDFormatter**, disponível no endereço *https://www.sdcard.org/downloads/formatter_4/*. A Figura E.1 mostra a tela do programa SDFormatter. Para efetuar a formatação é necessário seguir os seguintes passos:

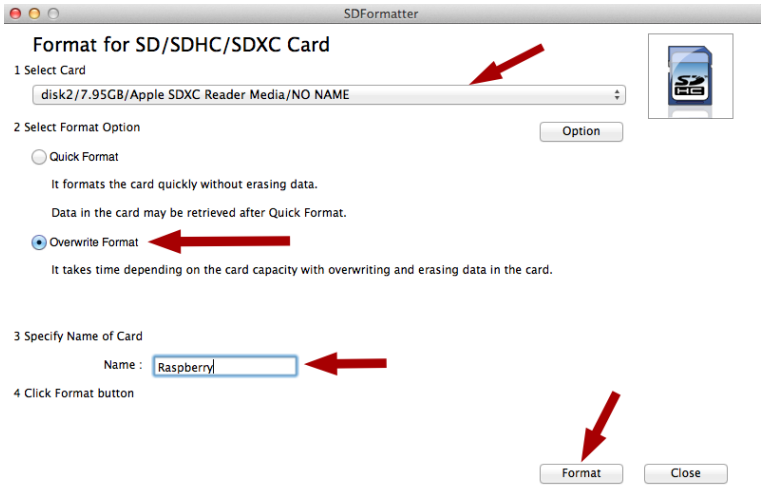

Figura E.1: Tela do SDFormatter

1. Selecione o cartão.
2. Selecione o tipo de formatação.
3. Coloque o nome do cartão.
4. Clique em *Format* e aguarde.

E.2 Instalando Raspbian usando o Mac OSX

Faça o download da imagem no endereço *http://www.raspberrypi.org/downloads/* depois faça o download do programa **SD card setup – Raspberry Pi – Mac** no endereço *https://alltheware.wordpre way-sd-card-setup/*, execute e de os privilégios de administrador, selecione a imagem do Raspbian, logo após será apresentada a tela de seleção do SD Card E.2, marque a opção correta e clique em **OK**.

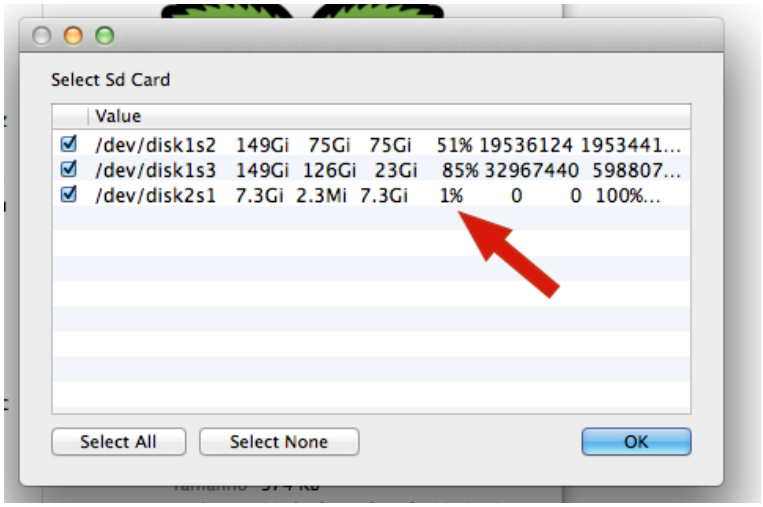

Figura E.2: Tela do card setup

Um símbolo de engrenagem assim ficará rodando até o término da instalação E.3.

Figura E.3: Instalação em curso

E.3 Implementação do subsistema de gerenciamento de usuários

Neste Capítulo será apresentado um pequeno tutorial de instalação, do subsistema.

E.3.1 Apache 2

A instalação do apache 2 é simples, sendo apenas necessário efetuar o comando de instalação, como mostra a Figura E.4.

```
Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.
root@TCC:~# apt-get install apache2
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os pacotes extra a seguir serão instalados:
  apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common libapr1
  libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap ssl-cert
Pacotes sugeridos:
  apache2-doc apache2-suexec apache2-suexec-custom openssl-blacklist
Os NOVOS pacotes a seguir serão instalados:
  apache2 apache2-mpm-worker apache2-utils apache2.2-bin apache2.2-common
  libapr1 libaprutil1 libaprutil1-dbd-sqlite3 libaprutil1-ldap ssl-cert
0 pacotes atualizados, 10 pacotes novos instalados, 0 a serem removidos e 0 não
atualizados.
É preciso baixar 1.494 kB de arquivos.
Depois desta operação, 5.122 kB adicionais de espaço em disco serão usados.
Você quer continuar [S/n]? S_
```

Figura E.4: Instalação do apache 2.

E.3.2 PHP 5

No PHP 5 é apenas necessário instalá-lo conforme a Figura E.5.

```
root@TCC:~# apt-get install php5
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os pacotes extra a seguir serão instalados:
  apache2-mpm-prefork libapache2-mod-php5 libonig2 libqdbm14 php5-cli
  php5-common
Pacotes sugeridos:
  php-pear
Os pacotes a seguir serão REMOVIDOS:
  apache2-mpm-worker
Os NOVOS pacotes a seguir serão instalados:
  apache2-mpm-prefork libapache2-mod-php5 libonig2 libqdbm14 php5 php5-cli
  php5-common
0 pacotes atualizados, 7 pacotes novos instalados, 1 a serem removidos e 0 não a
tualizados.
É preciso baixar 6.212 kB de arquivos.
Depois desta operação, 18,7 MB adicionais de espaço em disco serão usados.
Você quer continuar [S/n]? _
```

Figura E.5: Instalação do PHP 5.

E.3.3 MySQL e PHPMyAdmin

O MySQL e PHPMyAdmin são instaldos conforme mostrado nas Figuras E.6 e E.7.

E.3.4 Git

A instalação do Git é necessária para baixar do repositório o subsistema de gerenciamento de usuários. Da mesma forma que foi feito com as aplicações anteriores, é necessário apenas efetuar uma instalação simples, mostrado na Figura E.8.

```
root@TCC:~# apt-get install mysql-server
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os pacotes extra a seguir serão instalados:
  libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18
  mysql-client-5.5 mysql-common mysql-server-5.5 mysql-server-core-5.5
Pacotes sugeridos:
  libipc-sharedcache-perl libterm-readkey-perl tinyca
Os NOVOS pacotes a seguir serão instalados:
  libaio1 libdbd-mysql-perl libdbi-perl libhtml-template-perl libmysqlclient18
  mysql-client-5.5 mysql-common mysql-server mysql-server-5.5
  mysql-server-core-5.5
0 pacotes atualizados, 10 pacotes novos instalados, 0 a serem removidos e 0 não
atualizados.
É preciso baixar 8.859 kB de arquivos.
Depois desta operação, 92,1 MB adicionais de espaço em disco serão usados.
Você quer continuar [S/n]? _
```

Figura E.6: Instalação do MySQL Server.

```
root@TCC:~# apt-get install phpmyadmin
Lendo listas de pacotes... Pronto
Construindo árvore de dependências
Lendo informação de estado... Pronto
Os pacotes extra a seguir serão instalados:
  dbconfig-common fontconfig-config libfontconfig1 libgd2-xpm libjpeg8
  libltdl7 libmcrypt4 libpng12-0 libxpm4 php5-gd php5-mcrypt php5-mysql
  ttf-dejavu-core
Pacotes sugeridos:
  libgd-tools libmcrypt-dev mcrypt
Os NOVOS pacotes a seguir serão instalados:
  dbconfig-common fontconfig-config libfontconfig1 libgd2-xpm libjpeg8
  libltdl7 libmcrypt4 libpng12-0 libxpm4 php5-gd php5-mcrypt php5-mysql
  phpmyadmin ttf-dejavu-core
0 pacotes atualizados, 14 pacotes novos instalados, 0 a serem removidos e 0 não
atualizados.
É preciso baixar 8.709 kB de arquivos.
Depois desta operação, 22,7 MB adicionais de espaço em disco serão usados.
Você quer continuar [S/n]? _
```

Figura E.7: Instalação do PHPMyAdmin.

```
~# apt-get install git
```

Figura E.8: Instalação do Git.

E.3.5 Implantando o Subsistema

Para o sistema funcionar corretamente é necessário alguns procedimentos de configuração, descritos a seguir.

Baixando a aplicação

Para implantar o subsistema de gerenciamento de usuários, a aplicação deve ser baixada do repositório do *GitHub* em uma pasta pessoal, com o seguinte comando:

git clone https://github.com/BlindNomad/Sistema-de-Identificacao-Veicular.git

Preparando o servidor web

Entra-se na pasta **./Sistema-de-Identificacao-Veicular/web application** e copia-se o seu conteúdo para **/var/www/**. É necessário que o modo *rewrite* do apache esteja ativo para o funcionamento correto do *framework* **HyperMVC** (FORNARI, 2014a) conforme é mostrado na Figura E.9. Edita-se o arquivo */etc/apache2/sites-available/default* com as configurações como mostra a Figura E.10 e reinicia-se o apache.

```
root@TCC:~# a2enmod rewrite
Enabling module rewrite.
To activate the new configuration, you need to run:
  service apache2 restart
```

Figura E.9: Ativando rewite.

Implantando o banco de dados

O administrador deve abrir a página do phpMyAdmin. Entra-se com o usuário e senha previamente configurado, e clica-se no botão **Importar** e logo em seguida no botão **Selecionar arquivo...**, como mostra a Figura E.11.

O arquivo **bancoDeDados.sql** que está dentro do diretório **web application** deve ser selecionado. Clique em executar no fim da página, após este procedimento o banco de dados estará operando, faltando apenas adicionar as categorias na tabela **classe**, procedimento este será descrito no anexo F.

```
<VirtualHost *:80>
        ServerAdmin webmaster@localhost

        DocumentRoot /var/www/public_html
        <Directory />
                Options FollowSymLinks
                AllowOverride All
        </Directory>
        <Directory /var/www/>
                Options Indexes FollowSymLinks MultiViews
                AllowOverride All
                Order allow,deny
                allow from all
        </Directory>

        ScriptAlias /cgi-bin/ /usr/lib/cgi-bin/
        <Directory "/usr/lib/cgi-bin">
                AllowOverride All
                Options +ExecCGI -MultiViews +SymLinksIfOwnerMatch
                Order allow,deny
                Allow from all
        </Directory>

        ErrorLog ${APACHE_LOG_DIR}/error.log

        # Possible values include: debug, info, notice, warn, error, crit,
        # alert, emerg.
        LogLevel warn

        CustomLog ${APACHE_LOG_DIR}/access.log combined
</VirtualHost>
```

Figura E.10: Editando o arquivo default do Apache.

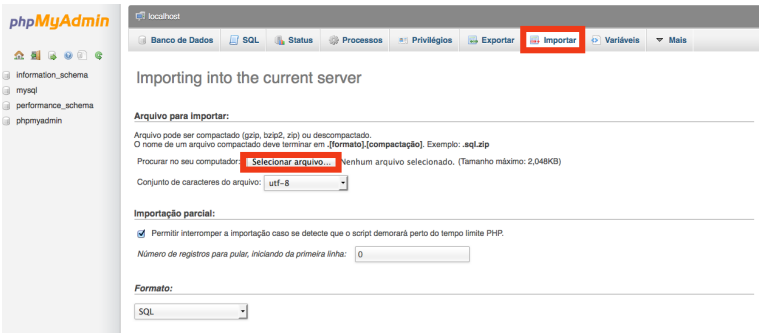

Figura E.11: Usando o phpMyAdmin.

Configurando o usuário ADM

No primeiro acesso, como não existem usuários cadastrados, o sistema entra sem a necessidade de digitar usuário e senha, conforme a Figura E.12.

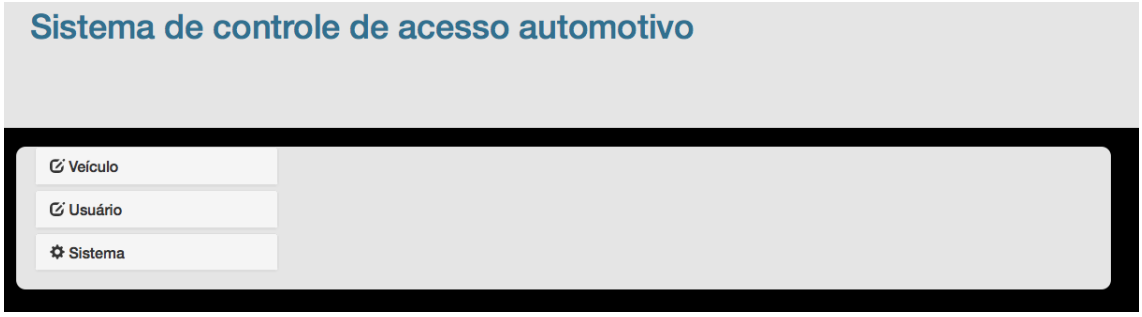

Figura E.12: Página inicial, subsistema de gerenciamento de usuários.

Para cadastrar o usuário, abre-se o menu **Usuário** e depois clica-se em cadastro. Preenche-se então os dados, como mostra a Figura E.13. Clica-se então em **Cadastrar**.

Figura E.13: Página de cadastro de usuários.

Depois que o usuário for cadastrado deve-se ir ao menu **Sistema** e clicar em **Sair**. Uma tela de *login* será mostrada se tudo ocorrer corretamente. Para concluir será necessário entrar novamente no banco de dados, através do **phpMyAdmin** para alterar os privilégios do usuário para administrador. Edita-se a tabela **pessoa**, colocando o número 3 na coluna **permissao**, como mostra a Figura E.14 e clica-se em executar.

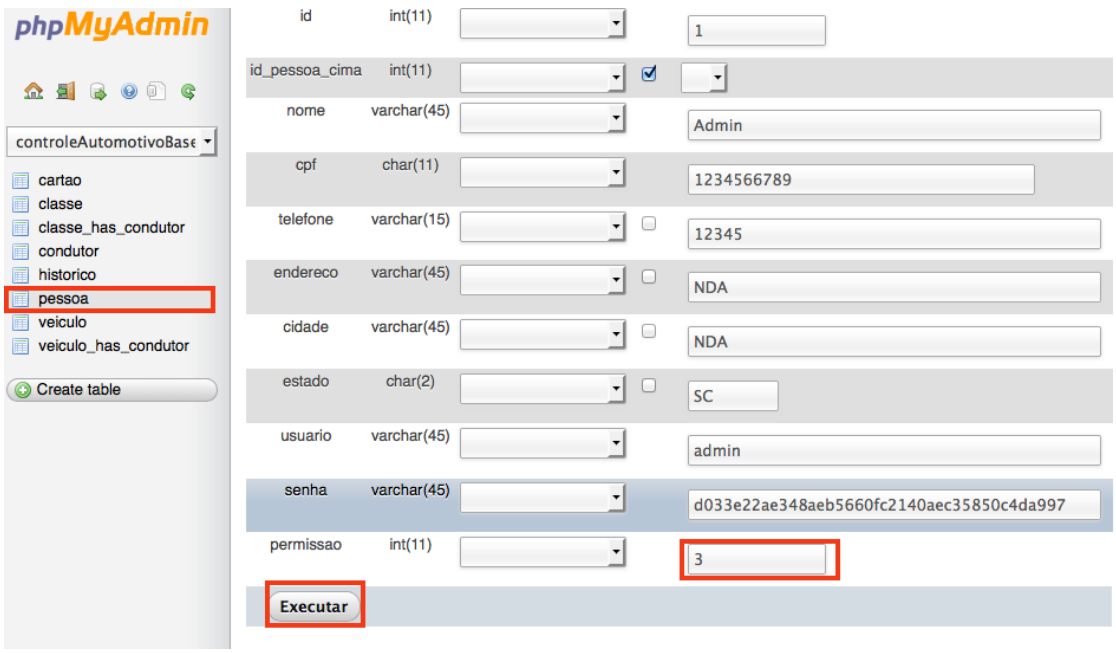

Figura E.14: Alterar usuário para administrador.

E.4 Implementação do subsistema de controle veicular

Este subsistema utiliza o *hardware* Raspberry Pi. Seu sistema operacional fica localizado em um cartão Secure Digital Card (SD), sendo necessário colocar o cartão em um computador para fazer a sua implementação. Os exemplos aqui mostrados, foram feitos em um **MacBook Pro** com o sistema operacional **MacOS X 10.9.5**.

A aplicação foi implementada em Java seguindo o projeto apresentado no Capítulo 3. Algumas adaptações foram reaizadas para contornar problemas encontrados.

E.4.1 Instalando o sistema operacional

O sistema operacional utilizado foi o Raspbian, por sua proximidade com sistemas operacionais populares como o Ubuntu e Debian.

Para instalar no SD é necessário formatá-lo em *File Allocation Table (FAT)* 32. Para isto foi utilizado programa *SDFormatter* (SD ASSOCIATION, 2014), indicado pela **Raspberry Pi**. O procedimentos de formatação é descrito no Anexo E.1.

Após o procedimento de formatação, é necessário instalar o sistema operacional no SD, para isto, deve-se acessar o *firmware* no endereço *http://www.raspberrypi.org/downloads/* e fazer o download do **Raspbian**, como mostra a Figura E.15.

Os procedimentos de instalação, são descritos no endereço *http://www.raspberrypi.org/documentation/i*

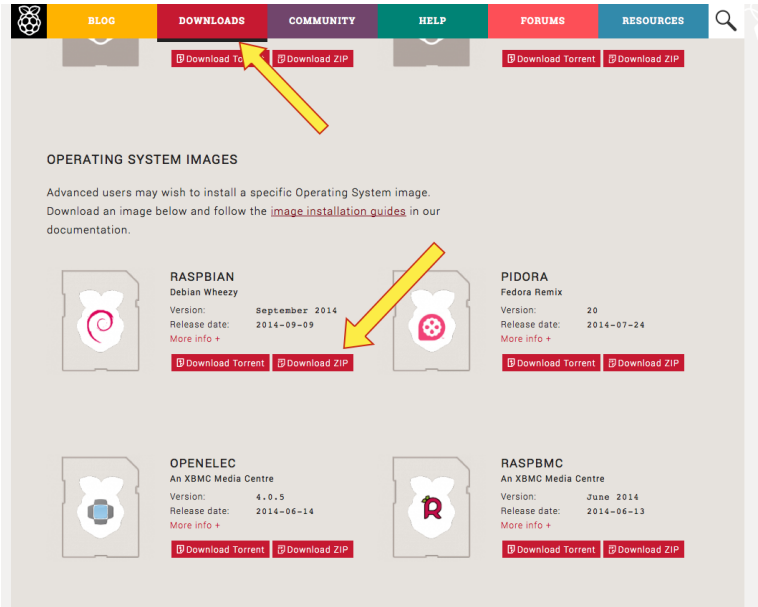

Figura E.15: Página de download do Raspbian.

images/README.md. Para **Mac**, será descrito no anexo E.2.

E.4.2 Configurando o sistema operacional

Deve-se retirar o cartão do leitor do computador, inserir no Raspberry e ligá-lo. Após alguns processos irá surgir a tela de configuração, bastando portanto seguir os seguintes passos:

1. Expandir arquivos selecionando a primeira opção;
2. Alterar a senha do usuário padrão, selecionando a segunda opção;
3. A terceira opção indica qual o tipo de boot é desejado, para o sistema é selecionado a primeira opção, na qual faz boot deixando o sistema no Shell;
4. A quarta opção é para configurações de idiomas;

   Em Configuring locales escolher a opção, pt_BR.UTF-8;

   Em Time Zone, escolha America – São_Paulo;

   Configurar o teclado de forma adequada.;
5. Ir direto a opção *Advanced Options* e ativar o **SSH**;
6. Selecionar *finish* e aguardar o processo de reinicialização.

E.4.3 Configuração das portas I/O

A placa Raspberry disponibiliza portas de IO para permitir o controle de sensores e atuadores. A Figura E.16 mostra a disposição física dos pinos na placa. A seguir é mostrado as funcionalidades dos pinos usados.

Figura E.16: Página de download do Raspbian.

- **Pino 15** é usado para informar bloqueio ou falha, no *software* ele é marcado como **pino 3**.
- **Pino 11** é usado para informar a liberação do condutor ou para informar que o sistema está aguardando passar o **identificador**, no *software* esta marcado como **pino 0**.
- **Pino 16** este pino mada um sinal de 3.3V quando o condutor está liberado, ele é marcado no *software* como **pino 4**.

E.4.4 Instalando pacotes e bibliotecas o subsistema

Após a inicialização do sistema é necessário fazer o comando de atualização do repositório de pacotes, *sudo apt-get update*. Para o funcionamento correto do subsistema é necessário a instalação de alguns pacotes e drivers.

Git

A instalação do Git é necessária para baixar do repositório o subsistema de controle veicular. Da mesma forma que foi feito com as aplicação anteriores, é necessário apenas efetuar uma instalação simples, *sudo apt-get install git*

MySQL

Este subsistema utiliza o banco de dados MySQL, sendo portanto necessário instalá-lo. Para fazer este procedimento é apenas necessário executar o comando *sudo apt-get install mysql-server*. Durante a instalação será solicitado o usuário e senha de acesso **root**.

Java e drivers

O Java é necessário, devido ao programa principal ser escrito nesta linguagem. Para instalá-lo deve-se executar o comando *sudo apt-get install openjdk-7-jre* e aguardar o fim da instalação. Para que o subsistema acesse o banco de dados é preciso instalar os drivers de acesso, bastando apenas executar o comando *sudo apt-get install libmysql-java*.

Instalando os drivers bluetooth e configurações

Para instalar os drivers Bluetooth, deve-se executar os comandos **sudo apt-get install bluetooth, sudo apt-get install libbluetooth3-dev** e confirmar as instalações. Os drivers bluetooth não funcionam corretamente no **Raspberry**, como é reportado por (VASSALLO, 2014) e (DEBIAN BUG REPORT LOGS, 2012), é necessário efetuar a alteração do arquivo de configuração no driver como mostrado abaixo.

1. Edite o arquivo **/etc/bluetooth/main.conf**;
2. Adicione a linha **DisablePlugins = pnat** FiguraE.17;
3. Salve o arquivo e saia, a alteração entrará em vigor após reiniciar.

Wiring Pi

Wiring Pi é uma excelente biblioteca para gerenciamento das entradas e saídas do Raspberry. Para instalá-la deve-se fazer os seguintes procedimentos:

```
GNU nano 2.2.6                                   Arquivo: /etc/bluetooth/main.conf

[General]

# List of plugins that should not be loaded on bluetoothd startup
#DisablePlugins = network,input

DisablePlugins = pnat

# Default adaper name
# %h - substituted for hostname
# %d - substituted for adapter id
Name = %h-%d

# Default device class. Only the major and minor device class bits are
# considered.
Class = 0x000100

# How long to stay in discoverable mode before going back to non-discoverable
# The value is in seconds. Default is 180, i.e. 3 minutes.
# 0 = disable timer, i.e. stay discoverable forever
DiscoverableTimeout = 0

# How long to stay in pairable mode before going back to non-discoverable
# The value is in seconds. Default is 0.
# 0 = disable timer, i.e. stay pairable forever
PairableTimeout = 0

# Use some other page timeout than the controller default one
# which is 16384 (10 seconds).
PageTimeout = 8192
```

Figura E.17: Resolvendo o bug no bluetooth do Raspberry.

1. Entrar em uma pasta pessoal e baixe o pacote, com o comando: *git clone git://git.drogon.net/wiringPi*;
2. Entrar na pasta **wiringPi** e de o comando: *git pull origin*;
3. Executar o arquivo **build**.

Após este procedimento a biblioteca **wiringPi** estará instalada.

Criando as tabelas

O subsistema veicular armazena os dados dos condutores em uma tabela MySql, portanto é necessário criá-las. Para tanto deve-se seguir os seguintes procedimentos:

1. Ler o arquivo de *script*, **cat Sistema-de-Identificacao-Veicular/vehicular application/-bancoDeDados/script.sql**.
2. Copiar o *script* para a área de transferência, conforme Figura E.18.
3. Entrar no MySql com o comando **mysql -u root -p**.
4. Colar o script e saia do programa.

Baixando a aplicação e configurando

Para implantar o subsistema de gerenciamento de usuários, deve-se realizar os seguintes procedimentos:

```
pi@raspberrypi ~ $ cat Sistema-de-Identificacao-Veicular/vehicular\ application/bancoDeDados/script
SET @OLD_UNIQUE_CHECKS=@@UNIQUE_CHECKS, UNIQUE_CHECKS=0;
SET @OLD_FOREIGN_KEY_CHECKS=@@FOREIGN_KEY_CHECKS, FOREIGN_KEY_CHECKS=0;
SET @OLD_SQL_MODE=@@SQL_MODE, SQL_MODE='TRADITIONAL,ALLOW_INVALID_DATES';

CREATE SCHEMA IF NOT EXISTS `car_core` DEFAULT CHARACTER SET utf8 COLLATE utf8_general_ci ;
USE `car_core` ;

-- -----------------------------------------------------
-- Table `car_core`.`condutor`
-- -----------------------------------------------------
CREATE TABLE IF NOT EXISTS `car_core`.`condutor` (
  `id` INT NOT NULL,
  `nome` VARCHAR(45) NOT NULL,
  `numero_carteira` VARCHAR(20) NOT NULL,
  `validade` DATETIME NOT NULL,
  `pontos` INT NOT NULL,
  `liberado` TINYINT NOT NULL,
  `cartao` VARCHAR(45) NOT NULL,
  PRIMARY KEY (`id`))
ENGINE = InnoDB;


SET SQL_MODE=@OLD_SQL_MODE;
SET FOREIGN_KEY_CHECKS=@OLD_FOREIGN_KEY_CHECKS;
SET UNIQUE_CHECKS=@OLD_UNIQUE_CHECKS;
```

Figura E.18: Script do banco de dados veicular.

1. baixar a aplicação do repositório do *GitHub*, em **/home/pi/** com o comando **git clone https://github.com/BlindNomad/Sistema-de-Identificacao-Veicular.git**;
2. Criar um arquivo chamado **configuratxt** com a senha do banco de dados, na pasta **/home/pi/**;
3. Criar o arquivo **placa.txt** e digite a placa do veículo;
4. Criar o arquivo **classe.txt** e coloque a categoria do veículo;
5. Criar o arquivo **url.txt** e digite o endereço do **subsistema de gerenciamento de usuários**.

A leitura do **identificador** passada por **bluetooth** é feita por um programa em separado feito em **C**. Deve-se compilar o programa com o comando **gcc Sistema-de-Identificacao-Veicular/bluetooth server/main.c -lbluetooth -o blueServer**. Para que tudo inicialize no *boot* do sistema, deve-se editar o arquivo **/etc/rc.local** e colocar o comandos como é mostrado na Figura E.19.

1. Inicializar o bluetooth como master;
2. Configurar a senha de pareamento como 1234;
3. Configurar o nome do dispositivo, de preferência colocar a placa do veículo;
4. Configurar em modo serial;
5. Iniciar o programa em C que controla a leitura da **Tag**;
6. Iniciar o subsistema de controle veicular.

Salvar o arquivo e reiniciar o subsistema com o comando **sudo reboot**.

```
thiagoj — pi@raspberrypi: ~ — ssh —
  GNU nano 2.2.6                                   Arquivo: /etc/rc.local

#!/bin/sh -e
#
# rc.local
#
# This script is executed at the end of each multiuser runlevel.
# Make sure that the script will "exit 0" on success or any other
# value on error.
#
# In order to enable or disable this script just change the execution
# bits.
#
# By default this script does nothing.

# Print the IP address
_IP=$(hostname -I) || true
if [ "$_IP" ]; then
  printf "My IP address is %s\n" "$_IP"
fi

sudo hciconfig hci0 piscan
sudo bluetooth-agent 1234 &
sudo hciconfig hci0 name LWX-7546
sdptool add SP
sudo /home/pi/blueServer &
sudo java -jar /home/pi/Sistema-de-Identificacao-Veicular/vehicular\ application/dist/CarCore.jar &

exit 0
```

Figura E.19: Inicialização do subsistema.

ANEXO F – Cadastrando categorias no banco de dados do subsistema de gerenciamento de usuários

Entre no phpMyAdmin do subsistema, selecione o banco **controleAutomotivoBase**, na tabela **classe**, clique em **inserir**, como mostra a Figura F.1.

Figura F.1: Inserindo dados na tabela classe

Preencha os dados das categorias existentes segundo o **Contran**, como mostra a Figura F.2.

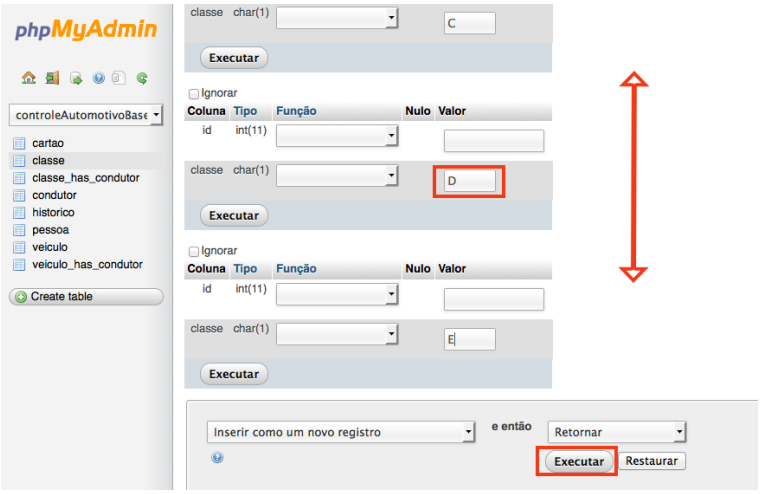

Figura F.2: Adicionando as categorias

ANEXO G – Testes de conectividade bluetooth

Na Figura G.1 é mostrado a tela de pareamento do celular Moto G, com a versão do Android 4.4.4.

Figura G.1: Pareamento.

Na Figura G.2 é mostrado a tela de configuração do pareamento no celular Moto G, com a versão do Android 4.4.4.

Na Figura G.3, é mostrado o erro na inicialização do subsistema de acesso.

Na Figura G.4, é mostrada a conexão efetuada com sucesso, via bluetooth.

Na Figura G.5 e mostrado o envio do identificador.

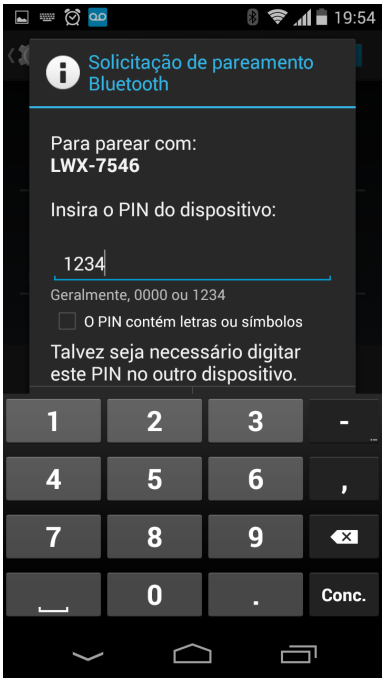

Figura G.2: Configuração do pareamento.

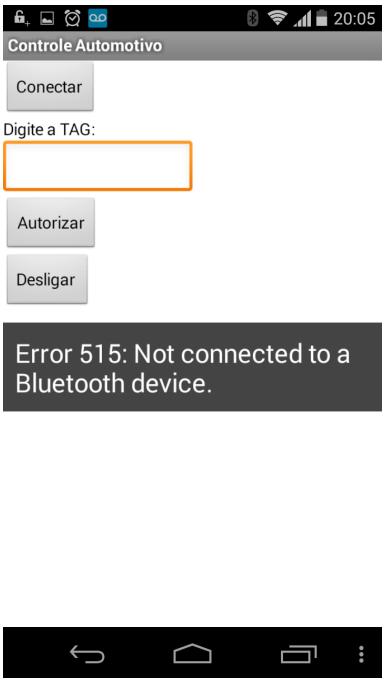

Figura G.3: Erro 515 no subsistema de acesso.

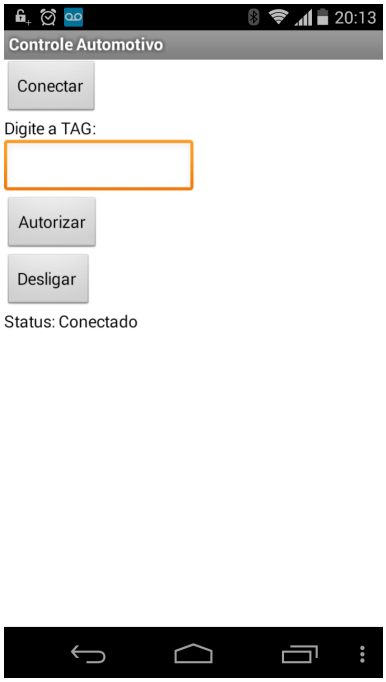

Figura G.4: Conexão com sucesso via bluetooth.

Figura G.5: Envio do identificador, com sucesso via bluetooth.

Lista de Abreviaturas

IC Integrated Circuit

SIG Special Interest Group

GATT Generic Attribute Profile

ATT Attribute Protocol

LE Low Energy

PDU Protocol Data Unit

UUID Universally Unique Identifier

L2CAP Logical Link Control and Adaptation Protocol

RF Radio Frequência

SD Secure Digital Card

FAT File Allocation Table

JSON JavaScript Object Notation

SMS Short Message Service

GPRS General Packet Radio Service

GPS Global Positioning System

MVC Model-View-Control

UML Unified Modeling Language

GIT GNU Interactive Tools

Referências Bibliográficas

AOYAMA, K. Next generation universal traffic management system. p. 649 – 653, 1998.

ARM. Arm1176 processor. 2014. Disponível em: <http://www.arm.com/products/processors/classic/arm11/arm1176.php>. Acesso em: 20 junho 2014.

BLUETOOTH. Covered core package version: 4.0. 2010.

BLUETOOTH. Generic attribute profile (gatt). 2014. Disponível em: <https://developer.bluetooth.org/TechnologyOverview/Pages/GATT.aspx>. Acesso em: 19 maio 2014.

BLUETOOTH. History of the bluetooth special interest group. 2014. Disponível em: <http://www.bluetooth.com/Pages/History-of-Bluetooth.aspx>. Acesso em: 18 maio 2014.

CONNECTBLUE. Shaping the wireless future with low energy applications and systems. 2014. Disponível em: <http://www.connectblue.com/press/articles/shaping-the-wireless-future-with-low-energy-applications-and-systems/>. Acesso em: 3 junho 2014.

CONNECTBLUE. Shaping the wireless future with low energy applications and systems. 2014. Disponível em: <http://www.connectblue.com/press/articles/shaping-the-wireless-future-with-low-energy-applications-and-systems/>. Acesso em: 3 junho 2014.

DEBIAN BUG REPORT LOGS. Rfcomm serving broken. 2012. Disponível em: <https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=690749>. Acesso em: 07 dezembro 2014.

DETRAN - PR. Tabela de correspondÊncia e prevalÊncia das categorias. 2014. Disponível em: <http://www.detran.pr.gov.br/modules/catasg/servicos-detalhes.php?tema=motorista&id=130>. Acesso em: 19 novembro 2014.

ELECTRONICS WEEKLY. What is... a gatt service? 203. Disponível em: <http://www.electronicsweekly.com/eyes-on-android/what-is/what-is-a-gatt-service-2013-08/>. Acesso em: 29 junho 2014.

EPX. Bluetooth: Att and gatt. 2014. Disponível em: <http://epx.com.br/artigos/bluetooth_gatt.php>. Acesso em: 19 maio 2014.

FORNARI, M. Hypermvc. 2014. Disponível em: <https://github.com/mateusfornari/HyperMVC>. Acesso em: 30 novembro 2014.

FORNARI, M. Hypersistence-alpha. 2014. Disponível em: <http://www.hypersistence.com>. Acesso em: 22 novembro 2014.

GOMES Érica de O. Alarme automotivo inteligente. 2012. Disponível em: <http://www.eletrica.ufpr.br/ufpr2/tccs/215.pdf>. Acesso em: 18 maio 2014.

GOOGLE BLOG. Where's my gphone? 2007. Disponível em: <http://googleblog.blogspot.com.br/2007/11/wheres-my-gphone.html>. Acesso em: 1 Fevereiro 2015.

GORDONS PROJECTS. Wiringpi. 2013. Disponível em: <https://projects.drogon.net/raspberry-pi/wiringpi/>. Acesso em: 19 novembro 2014.

IDGNOW. Android atinge 85smartphones. 2014. Disponível em: <http://idgnow.com.br/mobilidade/2014/07/31/android-atinge-85-de-participacao-no-mercado-mundial-de-smartphones/>. Acesso em: 1 Fevereiro 2015.

JSON. Javascript object notation. 2015. Disponível em: <http://www.json.org/>. Acesso em: 31 Janeiro 2015.

MIT. Mit app inventor. 2014. Disponível em: <http://appinventor.mit.edu>. Acesso em: 20 novembro 2014.

OPEN HANDSET ALLIANCE. Open handset alliance. 2015. Disponível em: <http://www.openhandsetalliance.com/>. Acesso em: 1 Fevereiro 2015.

OPEN SOURCE FOR U. Virtual machines for abstraction: The dalvik vm. 2015. Disponível em: <http://www.opensourceforu.com/2011/06/virtual-machines-for-abstraction-dalvik-vm/>. Acesso em: 18 Fevereiro 2015.

PETERVIS. Bcm2835 block diagram. 2014. Disponível em: <http://www.petervis.com/Raspberry_PI/BCM2835_Block_Diagram/BCM2835_Block_Diagram.html>. Acesso em: 20 junho 2014.

PI4J.COM/. The pi4j project. 2015. Disponível em: <http://pi4j.com/>. Acesso em: 31 Janeiro 2015.

RASPBERRY PI FOUDATION. Faqs. 2014. Disponível em: <http://www.raspberrypi.org/help/faqs/>. Acesso em: 20 junho 2014.

RASPBERRY PI FOUDATION. Faqs. 2014. Disponível em: <http://www.raspberrypi.org/help/faqs/#generalSoCUsed>. Acesso em: 20 junho 2014.

RASPBERRY PI FOUDATION. What are the power requirements? 2014. Disponível em: <http://www.raspberrypi.org/help/faqs/#powerReqs>. Acesso em: 20 junho 2014.

RENAULT. *Manual de proprietário, Fluence*. [S.l.], 2012. Disponível em: <http://www.renault.com.br/media/veiculos/fluence/docs/attb44dd4f7cca14399bf8bad813a2689d7/manual
Acesso em: 17 maio 2014.

SAFARI BOOKS ON LINE. Getting started with bluetooth low energy. 2015. Disponível em: <https://www.safaribooksonline.com/library/view/getting-started-with/9781491900550/ch01.html>. Acesso em: 18 Fevereiro 2015.

SD ASSOCIATION. Sd formatter 4.0 for sd/sdhc/sdxc. 2014. Disponível em: <https://www.sdcard.org>. Acesso em: 22 novembro 2014.

SERTRACEN. El salvador's smart driver's license and vehicle registration card. 2013. Disponível em: <http://www.gemalto.com/brochures/download/gov-el-salvador.pdf4>. Acesso em: 17 maio 2014.

SERTRACEN. Entrevista com roberto siegrist, diretor da sertracen. 2013. Disponível em: <http://www.gemalto.com/brasil/setor_publico/estudo_de_caso/el-salvador-interview.html>. Acesso em: 17 maio 2014.

SIQUEIRA, T. S. de. Bluetooth – características, protocolos e funcionamento. 2006. Disponível em: <http://www.ic.unicamp.br/d̃ucatte/mo401/1s2006/T2/057642-T.pdf>. Acesso em: 18 maio 2014.

SITEPOINT. The mvc pattern and php. 2015. Disponível em: <http://www.sitepoint.com/the-mvc-pattern-and-php-1/>. Acesso em: 18 Fevereiro 2015.

TELECO. Bluetooth: O que é? 2014. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialblue/pagina_1.asp>. Acesso em: 28 junho 2014.

TIOBE. Tiobe index for february 2015. 2015. Disponível em: <http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html>. Acesso em: 18 Fevereiro 2015.

UFRJ. 2012. Disponível em: <http://www.gta.ufrj.br/ensino/eel879/trabalhos_vf_2012_2/bluetooth/implementacao.htm>. Acesso em: 3 junho 2014.

VASSALLO, D. Android linux / raspberry pi bluetooth communication. 2014. Disponível em: <http://blog.davidvassallo.me/2014/05/11/android-linux-raspberry-pi-bluetooth-communication/>. Acesso em: 07 dezembro 2014.