UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

SISTEMA MÓVEL NA PLATAFORMA ANDROID PARA LOCALIZAÇÃO DE *FOOD TRUCKS* UTILIZANDO MAPAS

LEONARDO RIBEIRO

BLUMENAU
2015

2015/2-11

LEONARDO RIBEIRO

SISTEMA MÓVEL NA PLATAFORMA ANDROID PARA LOCALIZAÇÃO DE *FOOD TRUCKS* UTILIZANDO MAPAS

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Prof. Alexander Roberto Valdameri, Mestre – Orientador

BLUMENAU
2015

2015/2-11

SISTEMA MÓVEL NA PLATAFORMA ANDROID PARA LOCALIZAÇÃO DE *FOOD TRUCKS* UTILIZANDO MAPAS

Por

LEONARDO RIBEIRO

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

Presidente:	Prof. Alexander Roberto Valdameri, Mestre – Orientador, FURB
Membro:	Prof. Francisco Adell Péricas, Mestre – FURB
Membro:	Prof. Aurélio Faustino Hoppe, Mestre – FURB

Blumenau, 04 de dezembro de 2015.

Dedico este trabalho aos familiares, amigos, professores e a todos que torceram, acreditaram e contribuíram para a realização deste.

AGRADECIMENTOS

Aos meus pais, Benjamin Ribeiro e Mara de Paula Lima, por me incentivarem e me apoiarem em todos os momentos de minha vida.

Ao meu orientador, professor Alexander Roberto Valdameri, por ter acreditado na conclusão deste trabalho.

Aos professores do Departamento de Sistemas e Computação da Universidade Regional de Blumenau por suas contribuições durante os semestres letivos.

Que os vossos esforços desafiem as impossibilidades, lembrai-vos de que as grandes coisas do homem foram conquistadas do que parecia impossível.

Charles Chaplin

RESUMO

Há um movimento crescente na utilização de dispositivos móveis com acesso à internet na perspectiva de identificar serviços de interesse. Um movimento que tem ganhado espaço é o conhecido como *food truck*. Neste contexto, este trabalho apresenta o desenvolvimento de um aplicativo para a plataforma Android que através de um mapa utilizando os recursos da geolocalização apresenta a posição atual de *food trucks* cadastrados por usuários, que estão armazenados em um servidor *web service*. Para o desenvolvimento do mapa utilizou-se o Google Maps API v2, para o desenvolvimento utilizou-se a linguagem Java e para o servidor *web service* utilizou-se a linguagem PHP. Como resultado foi implementado um aplicativo que permite ao usuário cadastrar *food trucks*, visualizar veículos próximos a sua localização, realizar comentários sobre determinado veículo e adicionar os veículos aos favoritos. O *web service* implementado é responsável por armazenar registros e responder as requisições realizados pelo aplicativo.

Palavras-chave: *Food truck*. Geolocalização. Google Maps.

ABSTRACT

There is a growing movement in the use of mobile devices with access to the internet in order to identify services of interest. A movement that has gained ground is known as food truck. In this context, this paper presents the development of an application for the Android platform through a map using the resources of geolocation features the current position of food trucks registered by users, that are stored on a web server service. To develop the map we used the Google Maps API v2, for the development we used the Java language and the web service server used the PHP language. As a result it implemented an application that allows the user to register food trucks, view vehicles near your location, make comments about particular vehicle and add vehicles to your favorites. The web service implemented is responsible for storing records and answer the requests made by the application.

Key-words: Food truck. Geolocation. Google Maps.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API – *Application Programming Interface*

CDL – Câmara de Dirigentes Lojistas

GPS – Sistema de Posicionamento Global

JSON – *JavaScript Object Notation*

MER – Modelo Entidade Relacionamento

UML – *Unified Modeling Language*

SUMÁRIO

1 INTRODUÇÃO

No Brasil em 2014 o sistema operacional Android esteve em 89% dos dispositivos móveis (BUCCO, 2014). Por meio deste, os usuários tiveram acesso à internet, redes sociais, correio eletrônico, além de utilizar aplicativos de mobilidade urbana entre outras atividades.

Os aplicativos para plataforma Android alcançaram o gosto popular, não apenas pelas suas funcionalidades, mas também pela diversidade de aplicativos que são oferecidos (POZZEBON, 2015). Em 2014 foi registrado o crescimento de 40% de aplicativos desenvolvidos para o sistema operacional Android, que totalizaram 1,43 milhões de aplicativos desenvolvidos por 338 mil desenvolvedores (POZZEBON, 2015). É nesta perspectiva que se vislumbra um modelo de negócio para o segmento de *food truck*.

O modelo *food truck* utilizado no Brasil é baseado no modelo americano, que proporcionou a muitos chefes enxergaram este negócio como uma oportunidade de oferecer mais que uma simples refeição nas ruas. Eles começaram a focar em refeições mais específicas, determinadas como *gourmet*, atraindo a atenção e abrindo as portas para o sucesso do negócio. Os pontos móveis são considerados verdadeiros restaurantes sobre rodas, com cardápios especializados e segmentados, focando no bom gosto dos clientes, que passaram a acompanhar os roteiros de paradas dos *food trucks*, com a ajuda da divulgação em redes sociais (FERNANDES, 2014).

Um dos fatores do aumento dos *food trucks* no Brasil é o crescimento da procura por prestadores de serviços que são responsáveis pela adaptação de veículos para torná-los aptos para transportar cozinhas e condicionar adequadamente os alimentos. A empresa Fag Brasil indica que no primeiro semestre de 2014 a demanda já superou em 50% o resultado do mesmo período referente ao ano de 2013 (FERNANDES, 2014).

Para auxiliar na localização de *food trucks*, este trabalho se propõe a desenvolver uma aplicação *mobile* que auxilie os usuários a identificarem *food trucks* próximos a sua localização através da geolocalização.

1.1 OBJETIVOS

O objetivo geral do trabalho é o desenvolvimento de uma aplicação para Android para localização do serviço de *food trucks*.

Os objetivos específicos do trabalho proposto são:

a) aplicar o conceito de geolocalização para indicar a localização do serviço de *food trucks*;
b) proporcionar o rastreamento da oferta do serviço em determinada região.

1.2 ESTRUTURA

No primeiro capítulo é apresentado a introdução ao tema principal deste trabalho, seguido pela apresentação da justificativa e dos objetivos.

No segundo capítulo apresenta-se a fundamentação teórica sobre *Food trucks*, Geolocalização, Google Maps e trabalhos correlatos.

O terceiro capítulo apresenta o desenvolvimento do aplicativo, iniciando com o levantamento de informações e requisitos do sistema, tendo na sequência a especificação, diagrama de caso de uso, modelo entidade relacionamento, diagrama de classes, implementação, técnicas e ferramentas utilizadas, operacionalidade da implementação e resultados e discussão.

No quarto capítulo é apresentado as conclusões deste trabalho bem como sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

A seção 2.1 apresenta a breve contextualização sobre os *food trucks*, abrangendo sua criação e seu surgimento no Brasil. Na seção 2.2 são apresentados conceitos de geolocalização, cartografia e GPS. Na seção 2.3 são apresentados os *frameworks* de geolocalização para a plataforma Android. Por fim, a seção 2.4 apresenta os trabalhos correlatos.

2.1 *FOOD TRUCKS*

No ano de 1866 Charles Goodnight criou o primeiro veículo com o conceito de *food truck*, no estado do Texas. Charles sabia da dificuldade que os tocadores de rebanhos enfrentavam quando o assunto era alimentação. Devido a esse motivo decidiu adaptar um pequeno caminhão militar para levar comida a esses grupos de *cowboys* (CHAME, 2015).

Em 1872, o americano Walter Scott vendia tortas, sanduíches e cafés em uma carroça. Seus clientes eram os trabalhadores de jornais de Providence, no estado de Rhode Island nos Estados Unidos. O modelo foi muito copiado e se espalhou para outras regiões dos Estados Unidos. Após a Segunda Guerra Mundial, caminhões de comida móveis alimentavam os trabalhadores dos subúrbios nos Estados Unidos, regiões que tinham poucos restaurantes e uma população cada vez maior. Nessa época, os *food trucks* eram sinônimo de comida barata, sem muita preocupação com a qualidade (AGUIAR, 2015).

Os *food trucks* voltaram ao mercado após a crise de 2008, permitindo que as pessoas que perderam seus empregos conseguissem uma nova fonte de renda. Os *food trucks* também proporcionaram às pessoas uma forma mais barata de comida preparada na hora para o almoço (CHAME, 2015).

A moda chegou ao Brasil em 2012, quando os primeiros *food trucks gourmet* surgiram em São Paulo. Agora, os parques de *food trucks* já fazem parte do roteiro turístico das grandes cidades brasileiras e da paisagem urbana. A moda *gourmet* fez surgir uma tendência da alta gastronomia acessível (AGUIAR, 2015).

Segundo a coordenadora de *food trucks* da Câmara de Dirigentes Lojistas (CDL) de Florianópolis, Juliana Silva, Santa Catarina conta atualmente com cerca de cem *food trucks* (MORRIESEN, 2015). Há conhecimento de que pelo menos 20 veículos estão em funcionamento em Florianópolis, e no Vale do Itajaí operam mais dez, mesmo número de integrantes na Associação de *Food Trucks* de Joinville. Jaraguá do Sul tem oito, enquanto Blumenau e Lages também começam a fazer parte desta conta, em menor número (MORRIESEN, 2015).

Segundo Juliana Silva, a partir de 2016, ao mesmo tempo em que o número de carros irá aumentar, eventos como o Festival *Food Truck* irão diminuir (MORRIESEN, 2015). Atualmente, a falta de opções, e a ausência de legislação na maioria das cidades, impede que os *food trucks* possam atender diariamente em lugares públicos, fazendo que os empresários deste mercado estejam viajando por Santa Catarina e para outros estados para participar de eventos (MORRIESEN, 2015).

2.2 GEOLOCALIZAÇÃO

A Cartografia é a área do conhecimento que se preocupa em produzir, analisar e interpretar as diversas formas de se representar uma superfície, como os mapas e plantas (PENA, 2014). Segundo Woodward e Harley (1987, p. 1), os mapas são uma das formas mais antigas de comunicação entre os seres humanos. Conforme o mesmo autor, os mapas na antiguidade eram extremamente difundidos e já afetavam a vida, pensamento e imaginação das civilizações conhecidas.

A geolocalização entrou pela primeira vez na equação do Sistema de Posicionamento Global (GPS) na década de 1920, quando os sinais de rádio terrestres calculavam a direção e distância dos navios. O conceito de utilização de satélites para o GPS foi marcado após o lançamento do satélite Sputnik em 1957 pelos russos, onde os físicos William Guier e George Weiffenbach notaram que poderiam identificar a localização do satélite russo através da gravação de sinais de rádio frequência e corrigir para o efeito *Doppler*, no qual a frequência de rádio é maior quando um objeto está se aproximando e menor que o objeto se afasta (GRAZIADIO E-LEARNING, 2011).

Atualmente a geolocalização possui um alto potencial de poder social, devido ao fato de permitir que os usuários dos *smartphones* compartilhem a sua localização em redes sociais, assim aumentando a conscientização sobre determinada marca ou produto. A inclusão da geolocalização no desenvolvimento de aplicativos móveis permitiu que determinados aplicativos aumentassem a sua velocidade de aquisição de novos membros, sendo uma funcionalidade em franca expansão, especialmente desde marcas como Foursquare, Google e Facebook têm incorporado dentro de seus aplicativos móveis a geolocalização (NEKA, 2015).

2.3 GOOGLE MAPS

O Google Maps foi lançado e disponibilizado em 8 de fevereiro de 2005 aos usuários da internet, com o objetivo de fornecer tecnologia geoespacial para todos. A plataforma Google foi construída para disponibilizar toda infraestrutura e serviços do Google Maps, para

que milhões de usuários por dia possuam acesso às funcionalidades. O uso em escala dos mapas começou em 2005 com o desenvolvimento do Google Maps APIs (interface de programação Google Maps) que hoje é a engrenagem de mais de um milhão de *websites* e aplicações que acessam, com poucas linhas de programação, a maior base de dados geográficos (SCUSSEL, 2013).

A localização geográfica é uma informação preciosa às pessoas, podendo apresentar praticamente tudo, seja a simples localização de um ente querido, o roteiro de uma viagem de final de semana, um mapa para gestão de equipe de vendas de uma empresa ou até mesmo um detalhado plano de emergência integrando dezenas de secretarias de um Governo em questão (SCUSSEL, 2013).

O sucesso do serviço Google Maps APIs não é apenas pela disponibilidade aos usuários de 99.9%, mas também pela velocidade, simplicidade e familiaridade com o uso dos mapas Google (SCUSSEL, 2013). Fator esse que permite que os desenvolvedores não criem apenas aplicações com alto desempenho, mas também com uma cartografia de fácil compreensão (SCUSSEL, 2013).

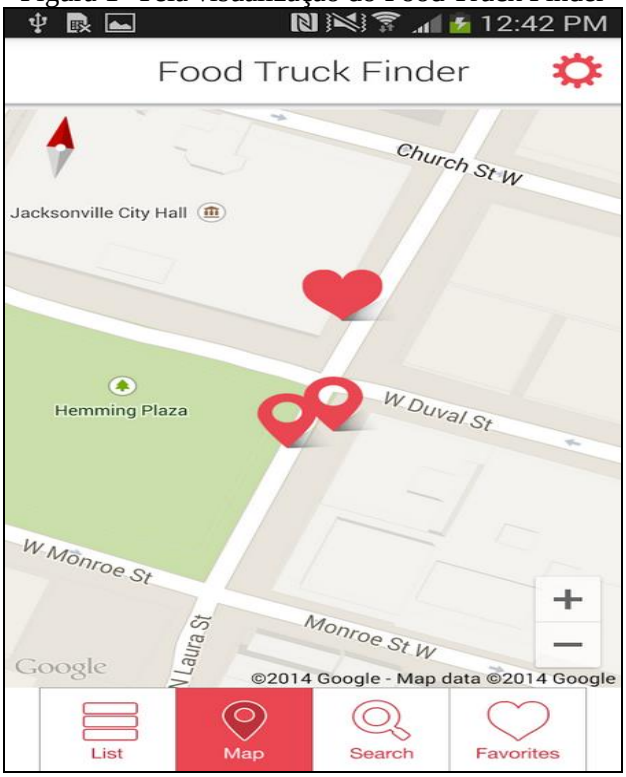
2.4 TRABALHOS CORRELATOS

Atualmente existem aplicativos similares ao que será desenvolvido, que estão disponíveis para os sistemas operacionais móveis iOS e Android. Entre eles estão o Food Truck Finder USA (FOOD TRUCK FINDER, 2015), Comida de Rua (FIBRA, 2014) e Food Truck nas Ruas (FERREIRA, 2015).

2.4.1 Food Truck Finder USA

Food Truck Finder USA é um aplicativo gratuito para consumidores e proprietários de *food trucks* que visa ajudar a interação entre usuários do segmento (FOOD TRUCK FINDER, 2015). O aplicativo permite visualizar os *food trucks* próximos ao usuário utilizando um mapa através das coordenadas do GPS. A Figura 1 mostra a tela de visualização de *food trucks* através do mapa.

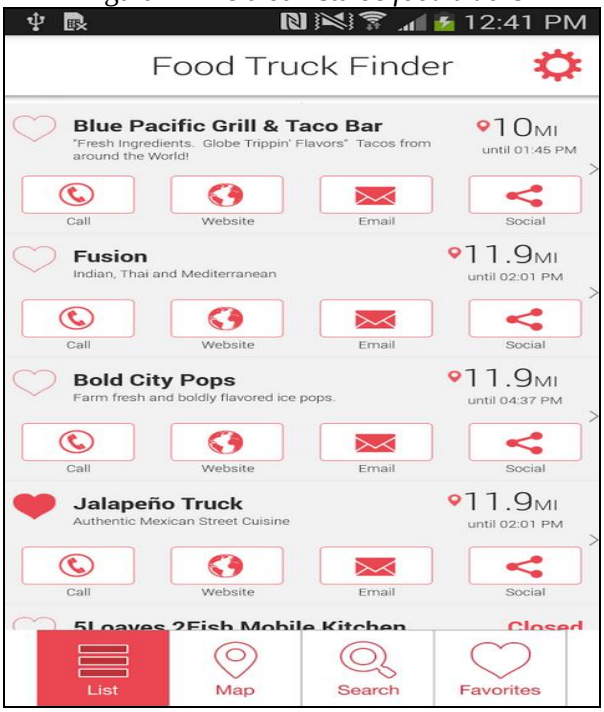
Figura 1- Tela visualização do Food Truck Finder



Fonte: Food Truck Finder (2015).

O aplicativo também apresenta ferramenta de busca através da lista de *food trucks* cadastrados, o qual disponibiliza informações como a distância entre o usuário e o veículo, opção de enviar e-mail, site do *food truck* e a opção de ligar para o veículo. A Figura 2 apresenta essa lista.

Figura 2 – Tela da lista de *food trucks*



Fonte: Food Truck Finder (2015).

2.4.2 Comida de rua

O aplicativo Comida de rua é um *software* gratuito que permite localizar *food trucks*, verificar programação de eventos, visualizar menus e ver fotos dos *food trucks*. Os veículos são demonstrados através de um mapa, apresentando informações como nome do veículo, nome da rua e número próximo para auxiliar a localização. O aplicativo disponibiliza a opção de promoções, onde essas promoções são buscadas através dos veículos que estejam ativos no mapa. O aplicativo disponibiliza a opção de visualizar eventos em parques que serão realizados, auxiliando na divulgação dos eventos. A Figura 3 apresenta a tela de localização dos veículos que é realizada através de um mapa.

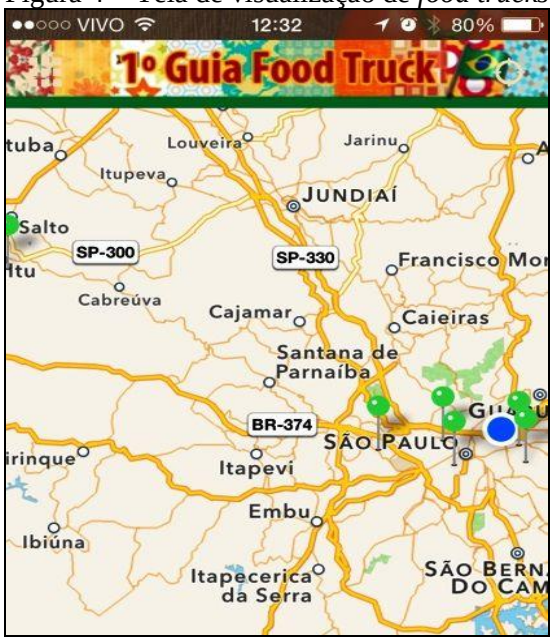
Figura 3 – Tela de localização de *food trucks*

Fonte: Fibra (2014).

2.4.3 Food Truck nas Ruas

O aplicativo Food Truck nas Ruas é um aplicativo gratuito que tem como objetivo ajudar os usuários a encontrarem o melhor da gastronomia em *food truck* nas ruas das cidades (FERREIRA, 2015). O aplicativo oferece ferramenta para localização de *food trucks* através de um mapa. A Figura 4 apresenta a tela de localização de veículos.

Figura 4 - Tela de visualização de *food trucks*



Fonte: Ferreira (2015).

O aplicativo também disponibiliza a opção de localizar *food trucks* cadastrados no aplicativo através da cidade onde este *food truck* esteja. Essa opção apresenta os *food trucks* em uma lista. A Figura 5 demonstra a tela de busca de *food trucks* por cidades.

Figura 5 – Tela busca *food truck* por cidade

Fonte: Ferreira (2015).

2.4.4 Análise comparativa

Os três aplicativos apresentados possuem o mesmo objetivo de permitir visualizar *food trucks* próximos à posição do usuário. O aplicativo Food Truck Finder USA possui

características de permitir buscar caminhões mesmo estando sem conexão com internet, basta somente o usuário carregar uma vez o aplicativo na internet que os veículos próximos serão armazenados no dispositivo móvel. Outro aspecto apresentado pelo aplicativo é habilitar a opção de ligação ao dono de um veículo visível no mapa, porém essa ligação não ocorre diretamente pelo aplicativo. Um dos diferenciais do aplicativo com os demais seria a demonstração do horário de encerramento do atendimento nos veículos.

O aplicativo Comida de rua possui a mesma característica do aplicativo Food Truck Finder USA de receber notificações de eventos, porém apresenta a característica de permitir visualizar fotos dos alimentos oferecidos pelos veículos. O aplicativo possui opção de visualizar o menu e a programação realizada pelos veículos.

O aplicativo Food Truck nas Ruas apresenta as mesmas características apresentadas nos aplicativos anteriores como notificação de eventos e a visualização de menus dos veículos.

O Quadro 1 exibe um comparativo básico sobre os três trabalhos correlatos.

Quadro 1 - Comparação dos trabalhos correlatos

Aplicativo	Locais	Plataforma	Notificação de eventos
Food Truck Finder USA (FOOD TRUCK FINDER, 2015)	Estados Unidos	Android e iOS	Não
Comida de rua Comida de Rua (FIBRA, 2014)	Brasil	Android	Sim
Food Truck nas Ruas (FERREIRA, 2015)	Brasil	Android e iOS	Sim

A parte de interação com o usuário nos aplicativos são muito parecidas, onde possuem opção de lista e mapa para localização de veículos próximos.

3 DESENVOLVIMENTO

Neste capítulo são apresentadas as características do aplicativo desenvolvido, como os requisitos funcionais e não funcionais o diagrama de caso de uso, diagrama de atividades e o Modelo Entidade Relacionamento (MER). São descritas as ferramentas e técnicas utilizadas durante o processo de desenvolvimento, a implementação, assim como os resultados obtidos.

3.1 LEVANTAMENTO DE INFORMAÇÕES

O aplicativo proposto tem como objetivo auxiliar aos usuários da plataforma Android na localização e divulgação de *food trucks*. A localização desses veículos será realizada através do posicionamento do usuário utilizando o GPS do *smartphone*.

No cenário atual, para que a população tenha conhecimento sobre a localização de *food trucks*, é necessário recorrer às mídias como televisão, rádio e internet que são as responsáveis pela divulgação dos eventos. Porém as divulgações realizadas nessas mídias não apresentam informações detalhadas sobre os veículos que estarão presentes nos eventos. O aplicativo proposto visa auxiliar na divulgação desses veículos, onde os usuários terão acesso a comentários realizados por outras pessoas que frequentaram os veículos, apresentando suas análises pessoais sobre os serviços prestados.

Os usuários cadastrados no aplicativo podem utilizar funcionalidades como cadastrar, localizar veículos no mapa, realizar comentários e adicionar veículos aos favoritos. A lista de veículos favoritos permite que os usuários recebam notificações sobre quando determinado veículo estiver disponível no mapa.

O aplicativo deve atender aos seguintes Requisitos Funcionais (RFs):

a) RF01: O aplicativo deve permitir que o usuário se cadastre no aplicativo;
b) RF02: O aplicativo deve permitir que o usuário realize o *login*;
c) RF03: O aplicativo deve permitir que o empreendedor cadastre *food trucks*;
d) RF04: O aplicativo deve permitir que o empreendedor adicione a localização de um *food truck* ao mapa;
e) RF05: O aplicativo deve permitir que o usuário visualize *food trucks* no mapa;
f) RF06: O aplicativo deve permitir que o cliente realize comentários aos *food trucks*;
g) RF07: O aplicativo deve permitir que o usuário visualize comentários realizados aos *food trucks*;
h) RF08: O aplicativo deve permitir que o cliente crie uma lista de *food trucks* favoritos.

Os Requisitos Não Funcionais (RNFs) do aplicativo são:

a) RNF01: O aplicativo deve ser desenvolvido na linguagem Java para plataforma Android;
b) RNF02: O aplicativo deve suportar versões do Android inferior à 4.1.2;
c) RNF03: O aplicativo deve utilizar o Google Maps Android API v2;
d) RNF04: O aplicativo deve ser desenvolvido utilizando Android Studio na versão 2.0.2.

Os Requisitos Funcionais (RFs) do servidor são:

a) RF09: O servidor deverá manter os registros dos *food trucks*;
b) RF10: O servidor deverá manter os registros da posição dos *food trucks*;
c) RF11: O servidor deverá manter os registros dos comentários;
d) RF12: O servidor deverá manter os registros dos favoritos;
e) RF13: O servidor deverá realizar a validação dos usuários.

Os Requisitos Não Funcionais (RNFs) do servidor são:

a) RNF05: O servidor deve utilizar uma base de dados MySQL 5.1;
b) RNF06: O servidor deverá ser desenvolvido em PHP na forma de *web service*;
c) RNF07: O servidor deverá utilizar *JavaScript Object Notation* (JSON) para o envio e recebimento de dados;
d) RNF08: O servidor deverá ser desenvolvido utilizando o Sublime Text 2 na versão 2.0.2.

3.2 ESPECIFICAÇÃO

A especificação deste trabalho foi desenvolvida utilizando diagramas de *Unified Modeling Language* (UML) e MER. Os casos de uso são apresentados através de um diagrama de casos de uso, seguido da descrição do cenário de cada caso de uso. Em seguida as classes do aplicativo e do servidor são demonstradas através do diagrama de classes. As entidades do banco de dados do servidor são demonstradas através do diagrama de MER. A ferramenta utilizada no desenvolvimento dos diagramas foi o Enterprise Architect versão 12.0.1215 para Windows 8.

3.2.1 Casos de uso

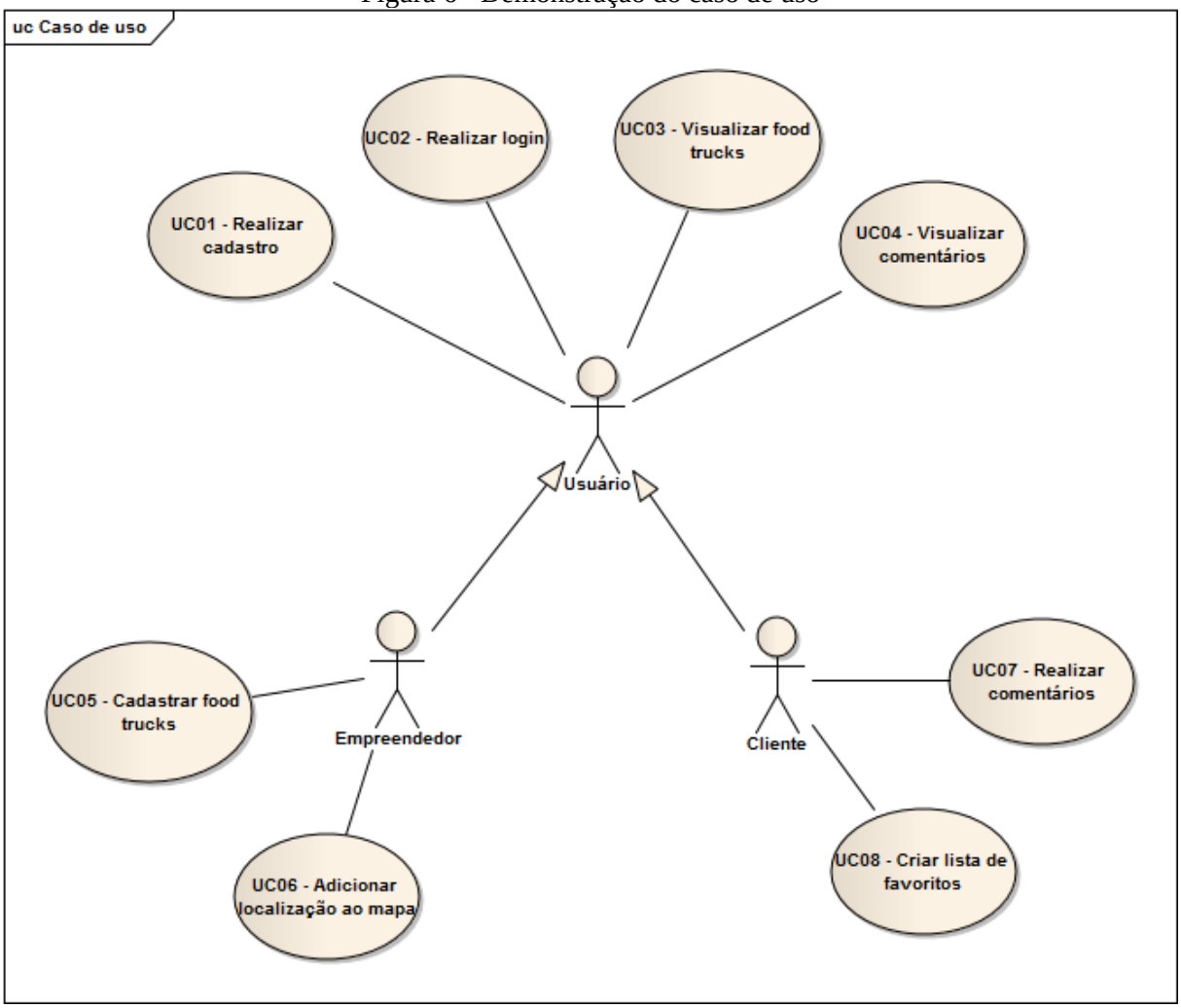
Esta subseção apresenta o diagrama de casos de uso do sistema desenvolvido. A partir dos requisitos do aplicativo, foram desenvolvidos oito casos de uso. Esses casos de uso

objetivam organizar os requisitos em funcionalidades que possam ser executadas de forma simples pelo usuário.

Os casos de uso são desempenhados por dois atores. O ator `empreendedor` representa o dono do *food truck*, que será responsável pelo cadastramento de seus veículos e marcação dos mesmos no mapa. O ator `cliente` representa o usuário comum do aplicativo, que visa somente buscar *food trucks* e avaliar os veículos.

O diagrama de casos de uso foi desenvolvido observando os padrões da UML, como pode ser visto na Figura 6. Os casos de uso estão associados aos atores responsáveis pela ação no caso de uso. A descrição dos casos de uso está apresentada no apêndice A.

Figura 6 - Demonstração do caso de uso



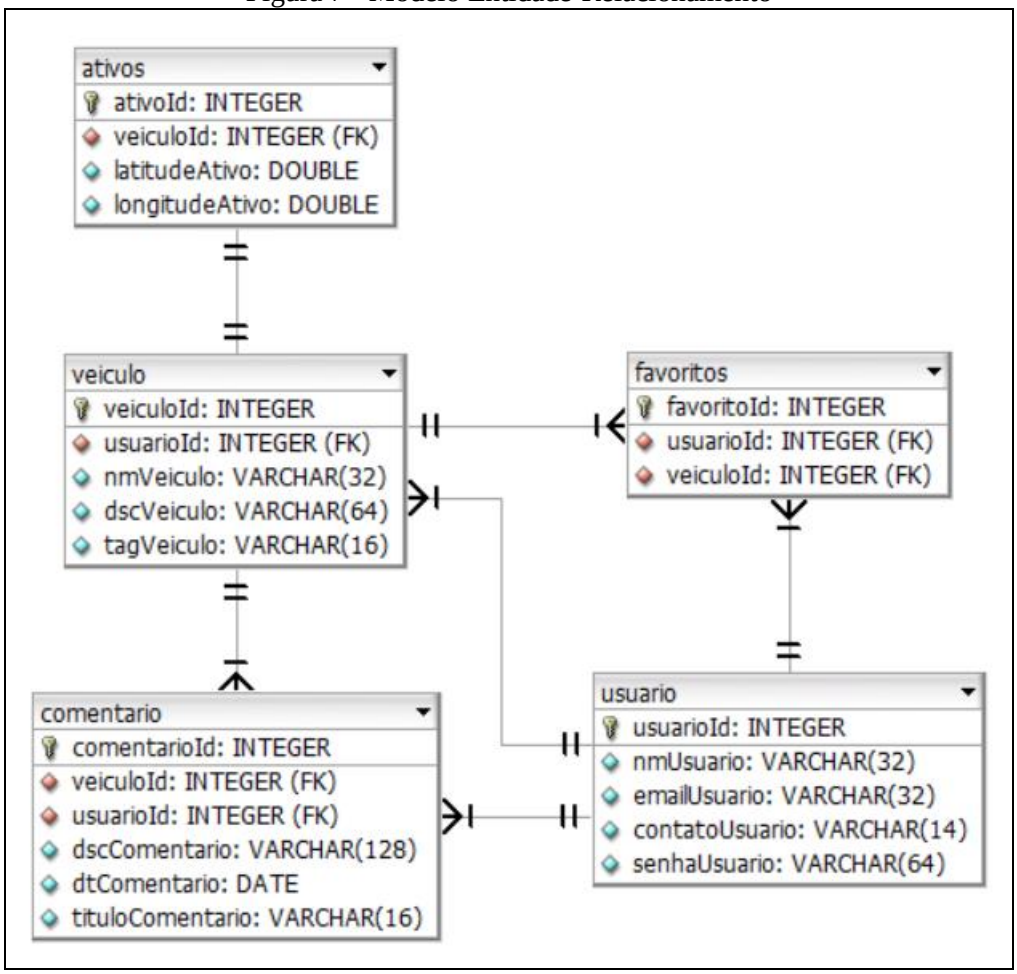
O caso de uso de `UC01 - Realizar cadastro` é utilizado quando o usuário não possui uma conta no aplicativo. O caso de uso `UC02 - Realizar login` é apresentado ao usuário quando ele não realizou o *login* no aplicativo. O caso de uso `UC03 - Visualizar food trucks` o usuário pode verificar quais veículos estão próximos a sua localização atual. No caso de uso `UC04 - Visualizar comentários`, ao selecionar um veículo ativo no mapa

o aplicativo carrega as informações do veículo, onde disponibiliza a opção de visualizar comentários do veículo. O caso de uso `UC05 - Cadastrar food trucks` o aplicativo permite que o usuário cadastre um novo veículo. No caso de uso `UC06 - Adicionar localização ao mapa`, ao acessar a página de cadastro do veículo o aplicativo disponibiliza a opção de habilitar o veículo no mapa. No caso de uso `UC07 – Realizar comentários`, ao selecionar um veículo no mapa o aplicativo permite o usuário realizar comentários sobre o veículo. O caso de uso `UC08 - Criar lista de favoritos` permite que o usuário adicione um veículo selecionado no mapa na sua lista de favoritos.

3.2.2 Modelo Entidade-Relacionamento

A Figura 7 demonstra o MER do sistema. O Dicionário de Dados está apresentado no Apêndice B.

Figura 7 - Modelo Entidade-Relacionamento



A tabela `ativos` é responsável pelo armazenamento da localização dos veículos. Esses registros serão utilizados no mapa do aplicativo, permitindo aos usuários visualizarem veículos próximos a sua localização.

A tabela `veiculo` é responsável pelo armazenamento dos veículos. Esses registros serão utilizados em funcionalidades como o mapa do aplicativo, comentários e favoritos.

A tabela `comentario` armazena avaliações dos usuários sobre determinado veículo. Esses registros serão utilizados para demonstrar a outros usuários avaliações sobre determinado veículo.

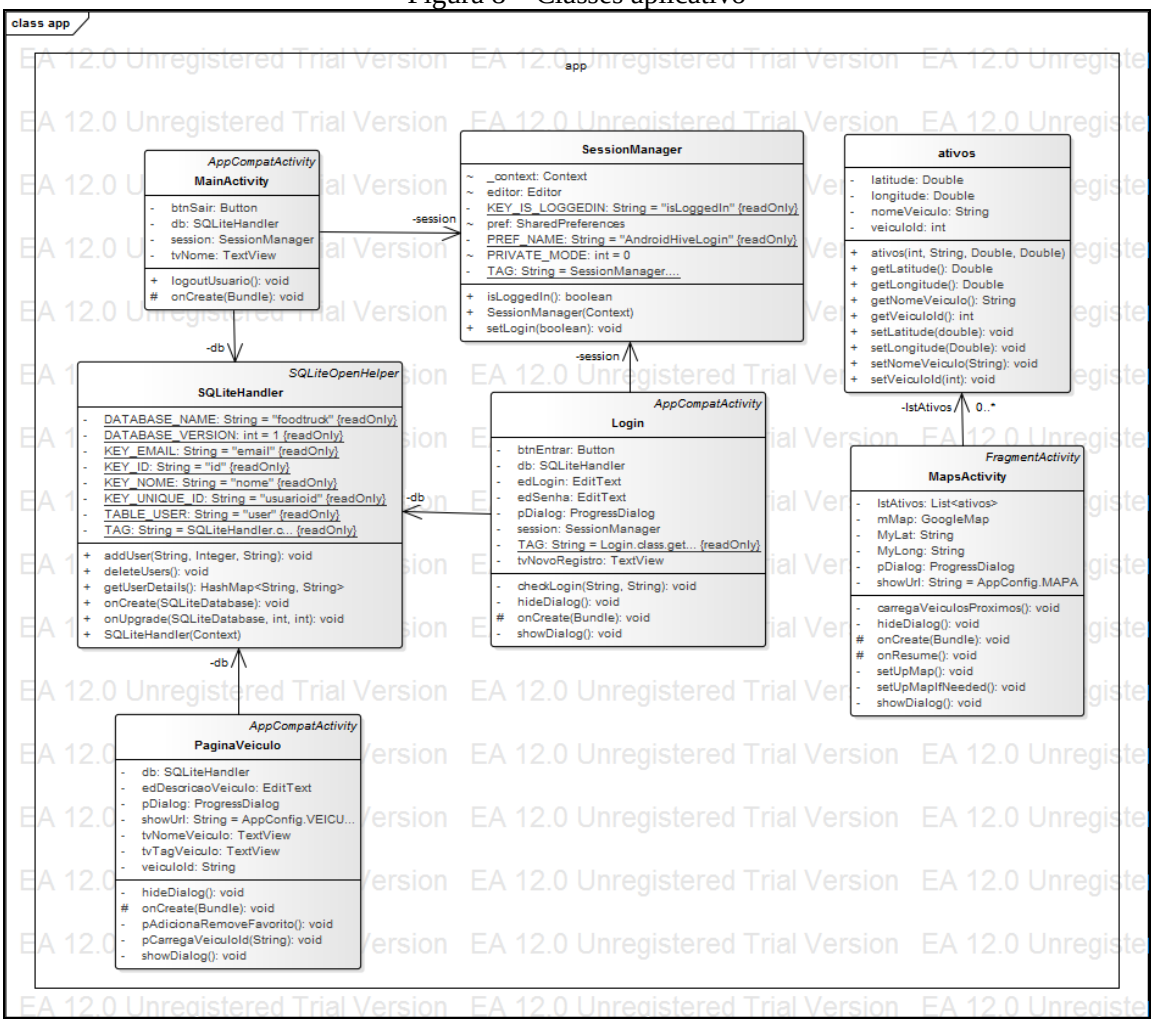
A tabela `favorito` armazena veículos de interesses dos usuários. Esses registros são utilizados para notificar aos usuários quando determinado veículo está em funcionamento.

A tabela `usuario` é responsável por armazenar informações dos usuários do aplicativo. Os registros contidos nessa tabela permitem que os usuários consigam utilizar as funcionalidades do aplicativo como cadastrar veículos, procurar veículos no mapa, realizar comentários entre outras funcionalidades.

3.2.3 Classes

A figura 8 exibe as classes existentes no aplicativo desenvolvido.

Figura 8 – Classes aplicativo



A classe `MainActivity` é a principal classe do aplicativo, responsável por realizar a chamada das funcionalidades do aplicativo, como rotina de cadastramento de veículos, localização de veículos e lista de favoritos.

A classe `SQLiteHandler` é responsável pela criação de tabelas internas no dispositivo móvel, como informações básicas do usuário que estiver utilizando o aplicativo.

A classe `PaginaVeiculo` é responsável pelo carregamento de informações sobre determinado veículo. Essa classe também realiza funcionalidades como adicionar veículo aos favoritos do usuário e chamar a rotina de comentários.

A classe `SessionManager` é responsável pelo controle de acesso do usuário no aplicativo. O controle de acesso permite que o usuário não necessite realizar o *login* toda vez que desejar utilizar o aplicativo.

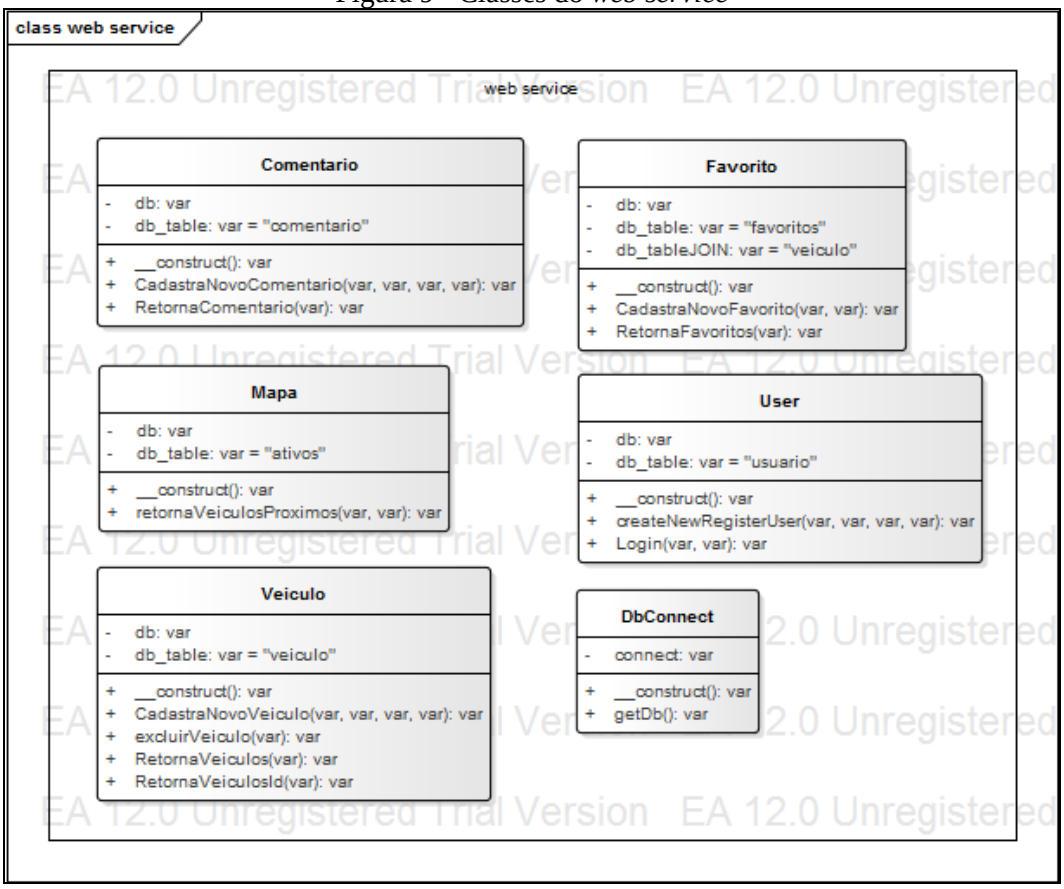
A classe `Login` é responsável pelo acesso do usuário ao aplicativo. Através da classe `SessionManager` o aplicativo verifica se o usuário já informou seu usuário e senha, caso informado o aplicativo realiza a chamada da classe `MainActivity` e permite acesso as funcionalidades do aplicativo. Caso o usuário não informou seu usuário o aplicativo solicita essas informações para permitir o acesso.

A classe `ativos` armazena as informações dos veículos que estejam disponíveis no mapa próximo ao usuário. Armazenando informações como localização, nome do veículo, descrição e *tag*.

A classe `MapsActivity` é responsável pelo carregamento dos veículos no mapa. Essa classe é responsável por enviar ao *web service* a localização do usuário, após isso o *web service* retorna os veículos ao aplicativo através de um arquivo em formato JSON, a classe fica responsável pela leitura desse arquivo e realiza a marcação dos veículos no mapa.

A figura 9 apresenta as classes existentes no *web service* desenvolvido.

Figura 9 - Classes do *web service*



A classe `Comentario` realiza o cadastramento de novos comentários realizados no aplicativo através do método `CadastraNovoComentario`. O método `RetornaComentario` realiza o envio de comentários ao aplicativo realizado sobre determinado veículo. Essa busca é realizada na tabela `comentario`, onde o aplicativo envia o código do veículo.

A classe `Mapa` possui os métodos responsáveis pelo envio de veículos próximos ao usuário a partir do método `retornaVeiculosProximos`, onde recebe por parâmetro a latitude e longitude do usuário para realizar essa busca na tabela `ativos`.

A classe `Veiculo` é responsável pelo cadastramento através do método `CadastraNovoVeiculo`, exclusão através do método `excluirVeiculo` e carregamento de veículos no aplicativo através dos métodos `RetornaVeiculos` e `RetornaVeiculosId`.

A classe `Favorito` é responsável pelo cadastramento de veículos através do método `CadastraNovoFavorito` e retorno de veículo favoritos de determinado usuário na tabela `favoritos` através do método `RetornaFavoritos`.

A classe `User` é responsável pelo cadastramento e liberação de acesso dos usuários no aplicativo. O cadastramento é realizado através do método `createNewRegisterUser` e o acesso do usuário ao aplicativo é realizado através do método `Login`.

A classe `DbConnect` é a classe responsável pela conexão com a base de dados MySQL. O método `getDb` é responsável pela configuração de acesso do *web service* com o banco de dados, onde realiza o envio de configurações como o local, usuário, senha e nome da base de dados.

3.3 IMPLEMENTAÇÃO

A seguir são mostradas as técnicas e ferramentas utilizadas e a operacionalidade da implementação.

3.3.1 Técnicas e ferramentas utilizadas

Nesta seção é apresentada a implementação do *web service*, demonstrando como foi realizada a conexão com a base de dados MySQL e também como ela envia as respostas ao aplicativo. Será apresentado a biblioteca Volley, que é responsável pela leitura e envio de informações no formado JSON entre aplicativo e *web service*. Será apresentado também o Google Maps utilizado para mostrar o carregamento de veículos próximos ao usuário.

O *web service* foi implementado na linguagem de programação PHP, onde no Quadro 2 é apresentado o construtor da classe `DbConnect` que é responsável pela configuração da conexão com o banco de dados.

Quadro 2 - Construtor da classe `DbConnect`

```
<?php
include_once 'config.php';
class DbConnect{
      private $connect;
      public function __construct(){
            $this->connect = mysqli_connect(DB_HOST, DB_USER,
                                            DB_PASSWORD, DB_NAME);
            if (mysqli_connect_errno($this->connect)){
              echo "Failed to connect to MySQL: " . mysqli_connect_error();
            }
      }
      public function getDb(){
            return $this->connect;
      }
}
?>
```

O Quadro 3 apresenta a classe `User` onde no construtor da classe instância o objeto da classe `DbConnect`.

Quadro 3 - Classe `User`

```
class User{
      private $db;
      private $db_table = "usuario";
      public function __construct(){
            $this->db = new DbConnect();
      }
```

A classe `User` é responsável pela validação das informações do usuário para acesso ao aplicativo. O método `Login` recebe por parâmetro o e-mail e a senha do usuário, através dessas informações a rotina realiza a busca na tabela `usuario`, caso as informações sejam encontradas a rotina retorna em formato JSON as informações necessárias do usuário para acesso ao aplicativo. Se os dados informados não forem encontrados é retornado pelo JSON a *tag* `success` com o valor zero, que indica falha. O quadro 4 apresenta o método `Login`.

Quadro 4 - Método `Login`

```
public function Login($emailUsuario, $senhaUsuario){
      $query = "select nmUsuario, emailUsuario, usuarioId from "
                  . $this->db_table . "
                 where emailUsuario = '$emailUsuario' AND
                    senhaUsuario = '$senhaUsuario' Limit 1";

       $result = mysqli_query($this->db->getDb(), $query);

       $number_of_rows = mysqli_num_rows($result);
       $json = array();
       if($number_of_rows > 0){

           while ($row = mysqli_fetch_assoc($result)){
            $json['success']              = 1;
            $json['usuario']['nmUsuario']      = $row['nmUsuario'];
            $json['usuario']['emailUsuario']   = $row['emailUsuario'];
            $json['usuario']['usuarioId']      = $row['usuarioId'];
            }
       }else{
            $json['success'] = 0;
      }

      mysqli_close($this->db->getDb());
      return $json;
}
```

A chamada do método `Login` é realizado na classe `UserLogin`, que é responsável pela chamada dos métodos existentes na classe `User`. Essa classe recebe através da função `$_POST` as informações repassadas pelo aplicativo, conforme demonstrado no Quadro 5.

Quadro 5 - Classe `UserLogin`

```
<?php
require_once 'userFunctions.php';

      $usuarioNome    = "";
      $usuarioContato = "";
      $usuarioEmail     = "";
      $usuarioSenha     = "";

      $tagAcao    = "";

if(isset($_POST['tagAcao'])){
      $tagAcao     = $_POST['tagAcao'];
}
if(isset($_POST['usuarioNome'])){
      $usuarioNome = $_POST['usuarioNome'];
}
if(isset($_POST['usuarioContato'])){
      $usuarioContato = $_POST['usuarioContato'];
}
if(isset($_POST['usuarioEmail'])){
      $usuarioEmail = $_POST['usuarioEmail'];
}
if(isset($_POST['usuarioSenha'])){
      $usuarioSenha = $_POST['usuarioSenha'];
}
```

Após o preenchimento das variáveis com os dados repassados pelo aplicativo, o sistema verifica através da variável `$tagAcao` qual o método que deve ser realizado, conforme demonstrado no Quadro 6.

Quadro 6 - Classe `UserLogin`

```
$userObject = new User();

if($tagAcao=="registrar"){
   if(!empty($usuarioNome) && !empty($usuarioContato) &&
      !empty($usuarioEmail) && !empty($usuarioSenha)){

      $hashed_password = md5($usuarioSenha);
      $json_registration = $userObject->createNewRegisterUser($usuarioNome,
                           $usuarioContato, $usuarioEmail, $hashed_password);
         echo json_encode($json_registration);
      }
}else if($tagAcao=="entrar"){
      // User Login
      if(empty($usuarioNome) && empty($usuarioContato) &&
         !empty($usuarioEmail) && !empty($usuarioSenha)){

         $hashed_password = md5($usuarioSenha);
         $json_array = $userObject->Login($usuarioEmail, $hashed_password);
         echo json_encode($json_array);
      }
}
```

A utilização da biblioteca Volley para a interação do aplicativo com o *web service* levou pela simplicidade e facilidade de uso. O Quadro 7 apresenta o envio de informações

para o *web service*, e determinando que nessa situação o método do *web service* a ser utilizado será o `Login`.

Quadro 7 - Método `checkLogin` da classe `Login`

```
@Override
protected Map<String, String> getParams() {
    // Posting parameters to login url
    Map<String, String> params = new HashMap<String, String>();
    params.put("usuarioEmail", email);
    params.put("usuarioSenha", senha);
    params.put("tagAcao"     , "entrar");

    return params;
}
```

Após o preenchimento das variáveis com os dados repassados pelo aplicativo, o aplicativo verifica através da variável `$tagAcao` qual o método que deve ser realizado, conforme apresentado no Quadro 8.

Quadro 8 - Classe `Login` - método `checkLogin`

```
StringRequest strReq = new StringRequest(Request.Method.POST,
AppConfig.LOGIN_URL, new Response.Listener<String>() {
   @Override
   public void onResponse(String response) {
      hideDialog();

      try {
           JSONObject jObj = new JSONObject(response);

           if (jObj.getInt("success")==1){
              session.setLogin(true);

              JSONObject jusuario = jObj.getJSONObject("usuario");
              String  nomeUsuario =
jusuario.getString("nmUsuario").toString();
              String  emailUsuario=
jusuario.getString("emailUsuario").toString();
              Integer idUsuario   = jusuario.getInt("usuarioId");

              db.addUser(nomeUsuario, idUsuario, emailUsuario);

              startActivity(new Intent(getBaseContext(),
MainActivity.class));
              finish();
           }else{
              Toast.makeText(getApplicationContext(), "E-mail ou senha
estão incorretos!",
                                          Toast.LENGTH_LONG).show();
           }
      } catch (JSONException e) {
        // JSON error
        e.printStackTrace();
      }
}
```

Conforme demonstrado no Quadro 8, o aplicativo informa a classe do *web service* deseja utilizar, informando através do parâmetro `AppConfig.LOGIN_URL`. O método

`onResponse` que recebe as informações enviadas pelo *web service* em formato JSON, onde essas informações serão buscadas através de suas respectivas *tags* e atribuídas no banco SQLite do aplicativo.

A classe `AppConfig` é responsável por armazenar o diretório das classes contidas no *web service*, conforme demonstrado no Quadro 9.

Quadro 9 - Classe `AppConfig`

```
public class AppConfig {

public static String LOGIN_URL      =
"http://meufoodtruck.mobi/userLogin.php";

public static String REGISTER_URL   =
"http://meufoodtruck.mobi/userLogin.php";

public static String VEICULOS_URL   =
"http://meufoodtruck.mobi/veiculo.php";

public static String INSERT_VEICULO =
"http://meufoodtruck.mobi/veiculo.php";

public static String COMENTARIO_URL =
"http://meufoodtruck.mobi/comentario.php";

public static String MAPA           =
"http://meufoodtruck.mobi/mapa.php";

public static String FAVORITO       =
"http://meufoodtruck.mobi/favorito.php";
}
```

O carregamento de veículos próximos no mapa é realizado através do método `retornaVeiculosProximos($latitudeAtivo, $longitudeAtivo)` que está presente na classe `Mapa` no *web service*, conforme demonstrado no Quadro 10.

Quadro 10 - Classe `Mapa` - Método `retornaVeiculosProximos`

```
public function retornaVeiculosProximos($latitudeAtivo, $longitudeAtivo){
      $query = "select ativ.latitudeAtivo, ativ.longitudeAtivo,
ativ.veiculoId, veic.nmVeiculo, (6371 *
            acos(
                  cos(radians(-19.83996)) *
                  cos(radians(ativ.latitudeAtivo)) *
                  cos(radians(-43.94910) - radians(ativ.longitudeAtivo)) +
                  sin(radians(-19.83996)) *
                  sin(radians(ativ.latitudeAtivo))
            )) as distance
            from " . $this->db_table . " ativ
                  join  veiculo veic on
                  (ativ.veiculoId = veic.veiculoId)

                  having distance <= 100000;";
```

O *select* realiza a busca na tabela de `ativos`, que contém o veículo e sua localização geográfica. Através da localização do usuário o método busca na tabela os veículos que esteja em até uma distância de 100 quilômetros do usuário.

No aplicativo a classe `MapsActivity` possui o método `carregaVeiculosProximos()`, que é responsável por informar ao *web service* a localização do usuário, após o envio o *web service* retorna ao aplicativo as informações em formato JSON. Se a *tag* `success` informada pela resposta do servidor seja um, o sistema localizou veículos próximos e realiza o carregamento dos mesmos ao mapa, conforme demonstrado no Quadro 11.

Quadro 11 - Classe `MapsActivity` - Método `caregaVeiculosProximos`

```
private void carregaVeiculosProximos(){
   StringRequest strReq = new StringRequest(Request.Method.POST, showUrl,
   new Response.Listener<String>() {

   @Override
   public void onResponse(String response) {
      try {
           JSONObject jObj = new JSONObject(response);

           if (jObj.getInt("success")==1) {
               JSONArray JArrayMapa = jObj.getJSONArray("mapa");

               for(int i = 0; i < JArrayMapa.length(); i ++) {
                   JSONObject ObjMapa = JArrayMapa.getJSONObject(i);

                   Integer id       = ObjMapa.getInt("veiculoId");
                   Double latitude  = ObjMapa.getDouble("latitudeAtivo");
                   Double longitude = ObjMapa.getDouble("longitudeAtivo");
                   String nomeVeiculo = ObjMapa.getString("nmVeiculo");

                   lstAtivos.add(new ativos(id, nomeVeiculo, latitude,
                                            longitude));

                   mMap.addMarker(new MarkerOptions().position(new
                                  LatLng(latitude,
                                         longitude)).title(nomeVeiculo));
               }
           }
       } catch (JSONException e) {
       }
   }
```

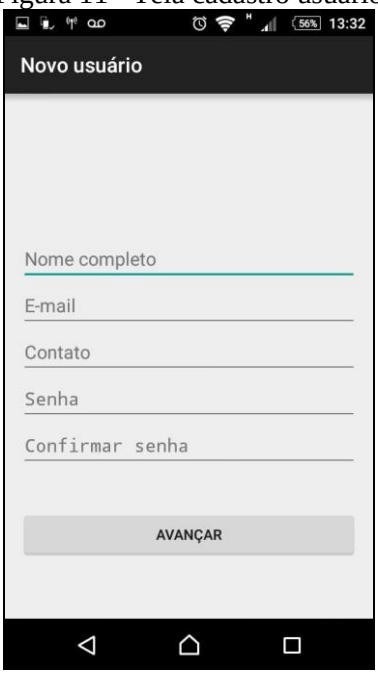
3.3.2 Operacionalidade da implementação

A tela *login* do aplicativo é apresentada para que o usuário possa acessar as funcionalidades do aplicativo. O botão entrar possui a funcionalidade de permitir que os usuários já cadastrados consigam acessar o aplicativo. A figura 10 apresenta a tela *login* do aplicativo.

Figura 10 - Tela *login* do aplicativo

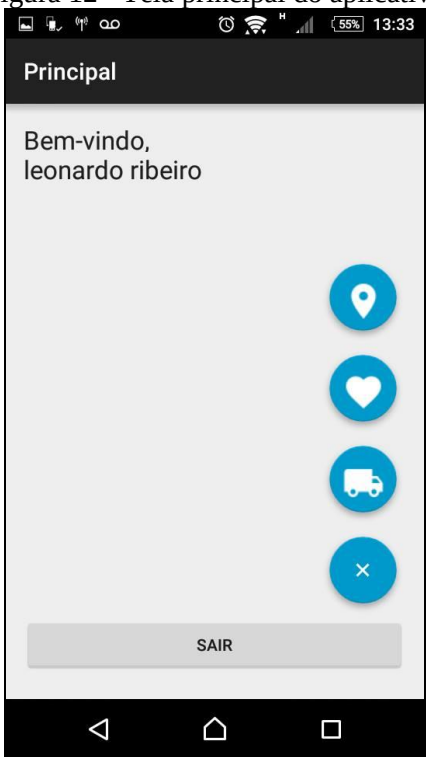
A tela *login* permite aos usuários que não são cadastrados no aplicativo, realizem o seu cadastro através da opção “Não é registrado? Increva-se”. Ao clicar nessa opção o aplicativo carrega a tela cadastro usuário, conforme apresentado na figura 11.

Figura 11 - Tela cadastro usuário



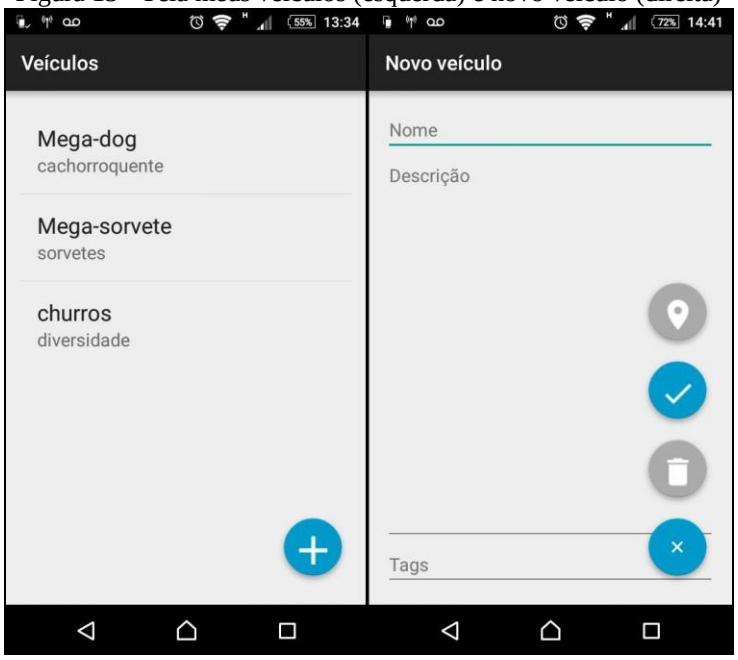
A tela principal do aplicativo habilita as opções de ver meus *food trucks* cadastrados, ver lista de veículos favoritos e localizar veículos no mapa. O botão sair permite que o usuário saia de sua conta do aplicativo, possibilitando a opção de realizar *login* com um usuário diferente. A figura 12 apresenta a tela principal do aplicativo.

Figura 12 - Tela principal do aplicativo



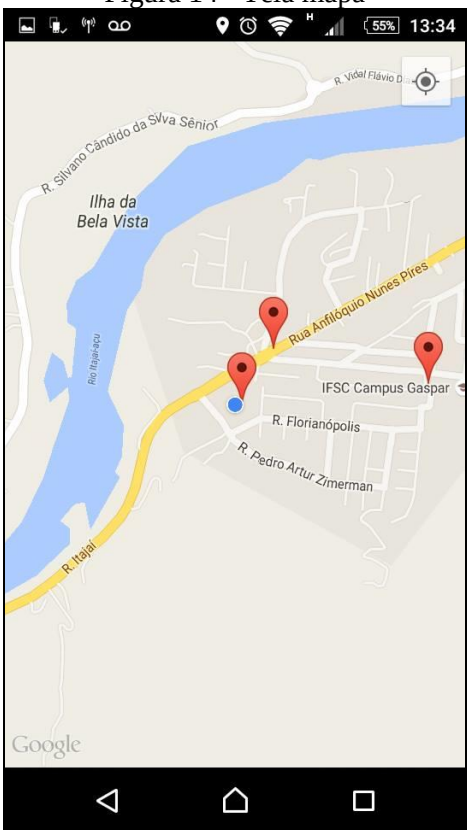
Ao selecionar a opção de meus veículos o aplicativo apresenta a tela veículos que lista os veículos que o usuário possui cadastrado. A tela possui a opção de cadastrar um novo veículo, conforme apresentado na figura 13.

Figura 13 - Tela meus veículos (esquerda) e novo veículo (direita)



A opção de localizar veículos no mapa apresentada na figura 12 apresenta ao usuário o mapa, onde é carregado os *food trucks* próximos a localização do usuário, como demonstrado na figura 14.

Figura 14 - Tela mapa



Os veículos ativos são marcados ao mapa, permitindo que o usuário clique em determinado veículo do mapa para acessar a página de informações do veículo.

3.4 ANÁLISE COMPARATIVA

Este trabalho desenvolveu um aplicativo para auxiliar aos usuários a localização e divulgação de *food trucks* oferecendo aos usuários funcionalidades como cadastramento, visualização de veículos e realização de comentários.

Comparando o aplicativo desenvolvido neste trabalho com os aplicativos Food Truck Finder USA, Comida de rua, Food Truck nas Ruas apresentados nos trabalhos correlatos, as seguintes características podem ser citadas a fim de destacar as similaridades entre os mesmos:

a) possuem o objetivo de permitir visualizar *food trucks* através de um mapa;
b) permitem aos usuários cadastrar veículos;
c) foram desenvolvidos para a plataforma Android.

O Quadro 12 apresenta uma comparação de algumas características do aplicativo desenvolvido neste trabalho, e os aplicativos Food Truck Finder USA (2015), Comida de rua (2014) e Food Truck nas Ruas (2015).

Quadro 12 - Comparativo com trabalhos correlatos

Característica	Aplicativo deste trabalho	Food Truck Finder USA (2015)	Comida de rua (2014)	Food Truck nas Ruas (2015)
Permite realizar comentários?	Sim	Não	Não	Não
Permite visualizar veículos ativos através de uma lista?	Não	Sim	Sim	Sim
Permite visualizar food trucks off-line?	Não	Sim	Não	Não
Permite upload de fotos?	Não	Sim	Sim	Sim

Com base no Quadro 12, convém destacar que, em relação aos demais trabalhos correlatos, aplicativo proposto possui o diferencial de permitir que os usuários realizem comentários sobre determinado veículo, que proporciona uma transparência sobre a experiência de pessoas que frequentaram determinado veículo em relação aos serviços prestados.

4 CONCLUSÕES

O trabalho desenvolvido atingiu o objetivo proposto que era a criação de um aplicativo para localização de *food trucks*, possibilitando que os usuários cadastrem e localizem veículos através de um dispositivo móvel que possua o sistema operacional Android. A principal vantagem do aplicativo é demonstrar aos usuários veículos que se encontram próximos através de um mapa, auxiliando na localização.

A proposta inicial para o desenvolvimento do *web service* foi realizado sobre a linguagem Java, mas foi verificado que a utilização da linguagem PHP tornaria o processo de desenvolvimento mais prático e ágil, devido a simplicidade apresentado pela linguagem para a conexão com o banco de dados.

A banco de dados também obteve mudanças de sua proposta, visando a simplicidade e agilidade. Inicialmente foi proposto a utilização do MongoDB, porém foi verificado que a utilização do MySQL seria mais funcional, devido a facilidade de conexão com o *web service* desenvolvido em PHP.

As ferramentas utilizadas para elaboração do aplicativo também se mostraram adequadas. A utilização das bibliotecas Volley e Google Maps API contribuíram na agilidade e aprimoramento do aplicativo. O maior desafio na produção do trabalho foi o envio de informações entre o aplicativo e o *web service*, onde foram necessários realizar vários ajustes para a leitura e geração correta das informações no formato JSON.

Acreditasse que o aplicativo possa auxiliar a popularização dos *food trucks*, já que permite aos donos desses veículos conquistarem novos clientes, justamente pelas informações mais significativas que são apresentadas na página dos veículos. A integração com o Google Maps proporcionou ao usuário uma maneira fácil de encontrar veículos próximos a sua localização.

4.1 EXTENSÕES

A partir deste trabalho, como sugestão para trabalhos futuros tem-se:

a) implementar um sistema de cardápio na página do veículo;
b) desenvolver uma opção de *upload* de fotos para ser adicionado na página do veículo;
c) desenvolver um sistema de rotas para chegar em determinado veículo;
d) implementar no mapa a opção do usuário verificar veículos de outras cidades;
e) destacar quando há um novo veículo no mapa;
f) adicionar opção de horário de funcionamento do veículo.

REFERÊNCIAS

AGUIAR, Anna C. **Food truck**: saiba como surgiu essa moda. [S.l.], 2015. Disponível em: <http://super.abril.com.br/blogs/historia-sem-fim/food-truck-saiba-como-surgiu-essa-moda/>. Acesso em: 22 mar. 2015.

BUCCO, Rafael. **Android está em 89% dos smartphones do Brasil**. [S.l.], 2014. Disponível em: <http://www.telesintese.com.br/android-esta-em-89-dos-smartphones-brasil/>. Acesso em: 04 abr. 2015.

CHAME, Roberto R. **A moda do food truck**:origem. [S.l.], 2015. Disponível em: <http://www.r2cpress.com.br/v1/2015/03/11/a-moda-do-food-truck-origem/>. Acesso em: 22 mar. 2015.

FERNANDES, Rosana. **Gastronomia de rua em alta com Food Trucks**. [S.l.], 2014. Disponível em: <http://www.mapadasfranquias.com.br/noticia/gastronomia-de-rua-em-alta-com-food-trucks/>. Acesso em: 04 abr. 2015.

FERREIRA, Kelly M. **Food Truck nas Ruas**. [S.l.], 2015. Disponível em: <https://play.google.com/store/apps/details?id=com.conduit.app_4fd548531023496e9df8380f467bdd7f.app&hl=pt_BR/>. Acesso em: 16 maio. 2015.

FIBRA. **Comida de Rua**. [S.l.], 2014. Disponível em: <https://play.google.com/store/apps/details?id=com.conduit.app_fb2548ffe8c84c93b7130259f43fa91d.app&hl=pt_BR/>. Acesso em: 16 maio. 2015.

FOOD TRUCK FINDER. **Food Truck Finder USA**. [S.l.], 2015. Disponível em: <https://play.google.com/store/apps/details?id=com.foodtruckfinderusa.foodtruckfinder/>. Acesso em: 16 maio. 2015.

GRAZIADIO E-LEARNING. **History of geolocation**. [Los Angeles], 2011. Disponível em: <https://wikis.pepperdine.edu/display/GSBME/History+of+Geolocation>. Acesso em: 21 mar. 2015.

MORRIESEN, Cláudia. **Food trucks ganham espaço e viram tendência de mercado em Santa Catarina**. [S.l.], 2015. Disponível em: <http://anoticia.clicrbs.com.br/sc/economia/noticia/2015/10/food-trucks-ganham-espaco-e-viram-tendencia-de-mercado-em-santa-catarina-4866830.html>. Acesso em: 15 out. 2015.

NEKA, Evelyn. **Geolocalização no desenvolvimento de aplicativos *mobile* é importante?**. [S.l.], 2015. Disponível em: <https://pt.yeeply.com/blog/geolocalizacao-no-desenvolvimento-de-aplicativos-mobile-e-importante/>. Acesso em: 06 dez. 2015.

PENA, Rodolfo A. **Cartografia. Produção de mapas e cartografia**. [S.l.], 2014. Disponível em: <http://www.brasilescola.com/geografia/cartografia.htm>. Acesso em: 21 mar. 2015.

POZZEBON, Rafaela. **Google Play ultrapassa App Store em número de aplicativos**. [S.l.], 2015. Disponível em: <http://www.oficinadanet.com.br/post/13963-google-play-ultrapassa-app-store-em-numero-de-aplicativos/>. Acesso em: 04 Abr. 2015.

SCUSSEL, Alexandre. **Artigo**: Por dentro do Google Maps. [S.l.], 2013. Disponível em: <http://mundogeo.com/blog/2013/07/01/artigo-por-dentro-do-google-maps/>. Acesso em: 16 maio. 2015.

WOODWARD, D.; HARLEY, J. B. **The History of Cartography**: Cartography in prehistoric, ancient, and medieval Europe and the Mediterranean. [Chicago: University of Chicago Press], 1987.

APÊNDICE A – Descrição dos casos de uso

A seguir são apresentados detalhamentos dos casos de uso conforme previstos nos diagramas apresentados na subseção 3.3.1.

O Quadro 13 apresenta a descrição do caso de uso “UC01 - Realizar cadastro”.

Quadro 13 – Descrição do caso de uso “UC01 - Realizar cadastro”

UC01. Realizar cadastro	
Requisitos atendidos	RF01.
Descrição	Permite ao usuário cadastrar-se no aplicativo, bem como alterar suas informações.
Cenário principal	O usuário realiza o preenchimento dos campos cadastrais na tela.
Pós-condição	O aplicativo envia os dados ao servidor, cadastrando o usuário no aplicativo.

O Quadro 14 apresenta a descrição do caso de uso “UC02 - Realizar *login*”.

Quadro 14 - Descrição do caso de uso "UC02 - Realizar *login*"

UC02. Realizar *login*	
Requisitos atendidos	RF02.
Descrição	Permite aos usuários cadastrados acesso ao aplicativo para o uso de suas funcionalidades.
Cenário principal	usuário abre o aplicativo; aplicativo apresenta tela de *login*; o usuário digita usuário e senha e clica em *login*; o aplicativo carrega a tela principal ao usuário.
Fluxo alternativo 01 – Usuário não registrado	usuário digita nome do usuário e senha desejada e clica em registrar; o aplicativo envia as informações para o servidor.
Fluxo alternativo 02 – Dados inválidos	aplicativo mostra mensagem de alerta informando dados inválidos; aplicativo exibe novamente os campos de *login*
Pré-condição	O usuário deve estar cadastrado no sistema.
Pós-condição	O aplicativo habilita as ferramentas para uso do usuário.

O Quadro 15 apresenta a descrição do caso de uso “UC03 - Visualizar *food trucks*”.

Quadro 15 - Descrição do caso de uso "UC03 - Visualizar *food trucks*"

UC03. Visualizar *food trucks*	
Requisitos atendidos	RF05.
Descrição	Permite ao usuário visualizar no mapa do aplicativo *food trucks* próximos a localização atual.
Cenário principal	usuário abre o aplicativo; usuário seleciona opção de visualizar *food trucks*; aplicativo carrega o mapa com os *food trucks* marcados; usuário navega pelo mapa.
Fluxo alternativo 01 – Não há food trucks ativos	aplicativo mostra mensagem que não foram encontrados *food trucks* na região atual.
Pré-condição	O usuário estar *online* no aplicativo.
Pós-condição	O aplicativo carrega no mapa os *food trucks* marcados.

O Quadro 16 apresenta a descrição do caso de uso “UC04 - Visualizar comentários”.

Quadro 16 - Descrição do caso de uso "UC04 - Visualizar comentários"

UC04. Visualizar comentários	
Requisitos atendidos	RF07.
Descrição	Permite ao usuário visualizar comentários realizados sobre os *food trucks*.
Cenário principal	seleciona *food truck* marcado no mapa; aplicativo carrega as informações do *food truck*; usuário seleciona opção de visualizar comentários; aplicativo carrega os comentários realizados sobre o veículo.
Fluxo alternativo 01 – Não há comentários cadastrados.	o aplicativo apresenta mensagem que não existem comentários sobre o veículo.
Pré-condição	Existirem comentários realizados nos *food trucks*.
Pós-condição	O aplicativo carrega os comentários realizados pelos usuários.

O Quadro 17 apresenta a descrição do caso de uso “UC05 - Cadastrar *food truck*”.

Quadro 17 - Descrição do caso de uso "UC05 - Cadastrar *food truck*"

UC05. Cadastrar *food truck*	
Requisitos atendidos	RF03.
Descrição	Permite ao empreendedor incluir, alterar ou excluir informações de um *food truck* no aplicativo.
Cenário principal	empreendedor seleciona opção de cadastrar *food truck*; aplicativo apresenta tela de cadastro de *food trucks*; empreendedor informa os dados do veículo; aplicativo envia as informações ao servidor.
Pré-condição	O usuário deve estar *online* no aplicativo.
Pós-condição	O *food truck* estará disponível para marcar ao mapa.

O Quadro 18 apresenta a descrição do caso de uso “UC06 - Adicionar localização ao mapa”.

Quadro 18 - Descrição do caso de uso "UC06 - Adicionar localização ao mapa"

UC06. Adicionar localização ao mapa	
Requisitos atendidos	RF04.
Descrição	Permite ao empreendedor marcar ou remover do mapa seus *food trucks* cadastrados.
Cenário principal	empreendedor solicita opção de visualizar seus *food trucks* cadastrados; aplicativo carrega os *food trucks*; empreendedor seleciona um *food truck*; aplicativo carrega informações do *food truck*; empreendedor seleciona opção de marcar *food truck* ao mapa; aplicativo realiza marcação no mapa da localização atual do *food truck*.
Pré-condição	O empreendedor deve possuir *food trucks* cadastrados.
Pós-condição	O *food truck* estará disponível para visualização dos clientes.

O Quadro 19 apresenta a descrição do caso de uso “UC07 - Realizar comentários”.

Quadro 19 - Descrição do caso de uso "UC07 - Realizar comentários"

UC07. Realizar comentários	
Requisitos atendidos	RF06.
Descrição	Permite ao cliente incluir e excluir comentários realizados sobre um *food truck* disponível no mapa.
Cenário principal	o cliente seleciona um *food trucks* cadastrados no mapa; o aplicativo carrega informações do *food truck*; o cliente seleciona opção para adicionar comentário.
Pré-condição	Devem existir *food trucks* cadastrados no mapa.
Pós-condição	O comentário estará disponível para visualização dos demais usuários.

O Quadro 20 apresenta a descrição do caso de uso “UC08 - Criar lista de favoritos”.

Quadro 20 - Descrição do caso de uso "UC08 - Criar lista de favoritos"

UC08. Criar lista de favoritos	
Requisitos atendidos	RF08.
Descrição	Permite ao cliente incluir e remover *food trucks* em sua lista de favoritos.
Cenário principal	cliente visualiza *food trucks* marcados no mapa; cliente seleciona opção de adicionar *food truck* aos favoritos; aplicativo carrega o *food truck* na lista dos favoritos.
Pré-condição	Deve existir *food trucks* cadastrados no aplicativo.
Pós-condição	Um favorito foi incluído ou removido da lista do cliente.

APÊNDICE B – Dicionário de Dados

Este Apêndice apresenta a descrição das tabelas do banco de dados apresentadas na subseção 3.3.2 deste trabalho.

Os tipos de dados utilizados nos atributos são:

a) int: armazena números inteiros;

b) string: armazena conjuntos de caracteres;

c) date: armazena data;

d) double: armazena números de precisão e escala fixos.

Quadro 21 - Tabela `ativos`

ativos		
Campo	Tipo	Descrição
ativoId	int	Chave gerada pelo MySQL
veiculoId	int	Chave do veículo
latitudeAtivo	double	Latitude de localização do veículo
longitudeAtivo	double	Longitude de localização do veículo

Quadro 22 - Tabela `veículo`

veiculo		
Campo	Tipo	Descrição
ativoId	int	Chave gerada pelo MySQL
usuarioId	int	Chave do veículo
nmVeiculo	string	Nome do veículo
dscVeiculo	string	Descrição do veículo
tagVeiculo	string	*Tag* de identificação do veículo

Quadro 23 - Tabela `comentario`

comentario		
Campo	Tipo	Descrição
comentarioId	int	Chave gerada pelo MySQL
veiculoId	int	Chave do veículo
usuarioId	int	Chave do usuário
dscComentario	string	Descrição do comentário
dtComentario	date	Data do comentário
tituloComentario	string	Título do comentário

Quadro 24 - Tabela `favoritos`

favoritos		
Campo	Tipo	Descrição
favoritoId	int	Chave gerada pelo MySQL
usuarioId	int	Chave do usuário
veiculoId	int	Chave do veículo

Quadro 25 - Tabela `usuario`

usuario		
Campo	Tipo	Descrição
usuarioId	int	Chave gerada pelo MySQL
nmUsuario	string	Nome do usuário
emailUsuario	string	E-mail do usuário
contatoUsuario	string	Número de contato do usuário
senhaUsuario	string	Senha criptografada do usuário