Revista da Graduação

Vol. 5 **No. 2** **2012** **13**

Seção: Faculdade de Informática

Título: EASYT – SISTEMA INTEGRADO DE MOBILIDADE URBANA

Autor: Lauro Madalosso Nunes e Fernando Landell de Moura Ruaro

Este trabalho está publicado na Revista da Graduação.
ISSN 1983-1374
http://revistaseletronicas.pucrs.br/ojs/index.php/graduacao/editor/submission/12419

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL
FACULDADE DE INFORMÁTICA
CURSO DE CIÊNCIA DA COMPUTAÇÃO

LAURO MADALOSSO NUNES
FERNANDO LANDELL DE MOURA RUARO

EASYT – SISTEMA INTEGRADO DE MOBILIDADE URBANA

Porto Alegre
2012

LAURO MADALOSSO NUNES
FERNANDO LANDELL DE MOURA RUARO

EASYT – SISTEMA INTEGRADO DE MOBILIDADE URBANA

Trabalho de conclusão de curso de graduação apresentado à Faculdade de Informática da Pontifícia Universidade Católica do Rio Grande do Sul, como requisito parcial para obtenção do grau de Bacharel em Ciência da Computação.

Orientador: Alexandre Agustini

Porto Alegre
2012

RESUMO

Há uma crescente popularidade no uso de smartphones e uma tendência de crescimento cada vez maior de tecnologias móveis. Na sociedade atual, percebe-se que as pessoas estão em busca de programas que facilitem suas vidas, especialmente programas que consigam facilitar suas ações cotidianas como, por exemplo, usufruir do transporte público.

É percebendo isto que estamos propondo o desenvolvimento de um sistema integrado de mobilidade urbana, que contemplará o desenvolvimento de aplicações para *smartphones*, que utilizem o sistema operacional Android, e aplicações web. Este sistema permitirá ao usuário interagir, por exemplo, com táxis, ônibus e lotações.

Palavras-chave: *Smartphones*. Android.

ABSTRACT

There is a growing popularity in the use of smartphones and a trend of ever increasing growth of mobile technologies. In today's society, people are looking for software that facilitates their lives, especially software that are able to facilitate their daily actions like, for example, the use of public transport.

It is realizing that we are proposing the development of an integrated urban mobility system, which will include the development of applications for smartphones, using the Android operation system, and web applications. This system will allow the user to interact, for example, with cabs, buses and jitneys.

Keywords: Smartphones. Android.

LISTA DE ILUSTRAÇÕES

LISTA DE TABELAS

LISTA DE SIGLAS E ABREVIATURAS

GPS – Global Positioning System
UTC – Universal Time Coordinated
SPS – Standard Positioning Service
PPS – Precise Positioning Service
XML – eXtensible Markup Language
HTTP – Hypertext Transfer Protocol
SMTP – Simple Mail Transfer Protocol
SOAP – Simple Object Access Protocol
WSDL – Web Service Description Language
UDDI – Universal Description Discovery and Integration
API – Application Programming Interface
IDE – Integrated Development Environment
WEB – World Wide Web – Rede Mundial de Computadores
SGBD – Sistema de Gerenciamento de Banco de Dados
SOA – Service-Oriented Architecture
WCF – Windows Communication Foundation
REST – Representation State Transfer
MVC – Model-View-Controller

SUMÁRIO

1 INTRODUÇÃO

Neste capítulo são apresentados o trabalho de conclusão de curso e a motivação para a realização dele, assim como seus objetivos e a organização do trabalho desenvolvido.

1.1 CONTEXTUALIZAÇÃO

Na atualidade, serviços rápidos e de fácil utilização têm dominado amplamente o mercado, pois cada vez mais as pessoas estão ocupadas e não têm tempo para desperdiçar. Através deste trabalho, pretendemos adicionar comodidade na vida daqueles que utilizam transporte público, assim como facilitar o trabalho daqueles que prestam esse tipo de serviço. Hoje em dia, por exemplo, no caso dos ônibus e lotações, o tempo de espera do passageiro é variável e incerto, pois dificilmente ele tem noção do real posicionamento do veículo. Já no caso dos táxis, o método mais comum para requisitar um táxi consiste em telefonar para uma cooperativa, informar o endereço onde se deseja o serviço e esperar que esta cooperativa encontre um táxi que tenha o interesse em atender a requisição, logo o passageiro deve esperar por um tempo incerto até que o táxi chegue.

1.2 MOTIVAÇÃO

Tendo em vista as situações descritas acima referentes à utilização de transporte público, podemos dividir o serviço de transporte em dois grupos: o grupo dos prestadores de serviço de táxi e o grupo dos prestadores de serviço de ônibus e lotações.

Com relação aos taxistas, grande parte depende de uma cooperativa para realizar suas corridas, pois é a cooperativa que recebe as chamadas dos passageiros e as redireciona a eles fornecendo as coordenadas do passageiro. Tendo em vista esta situação, por mais que o taxista esteja disposto a aproveitar ao máximo seu tempo de trabalho, ele acaba desperdiçando uma parte considerável deste tempo ocioso a espera de uma chamada. Através do sistema desenvolvido, apresentamos um modelo de interação que possibilita ao taxista ter contato direto

com os clientes. O taxista pode consultar possíveis passageiros, e até mesmo ser encontrado por um passageiro que esteja visualizando sua localização.

Com relação aos ônibus e lotações, propusemos um sistema que permita visualizar o posicionamento dos veículos e auxiliar na busca de trajetos a serem realizados. Ao informar um destino, o usuário terá acesso à quais linhas ele deverá utilizar.

1.3 OBJETIVO

O sistema foi desenvolvido com o objetivo de beneficiar todas as partes envolvidas no processo de mobilidade urbana: as empresas de transporte, os taxistas, as cooperativas de táxis, e as pessoas que utilizam serviços de transporte público e privado como meio de locomoção. Esse benefício é possibilitado através tanto do estabelecimento de uma comunicação direta entre taxistas e passageiros, quanto da visualização do posicionamento das linhas de ônibus e lotações. Nossa meta consistiu em desenvolver um sistema que tem três interfaces, cada uma direcionada especificamente para:

1. Passageiros: podem visualizar a localização dos taxistas, ônibus e lotações através de dispositivos móveis (*smartphone*[1], *tablet*[2],...) e tem a possibilidade de pesquisar informações relacionadas aos trajetos dos ônibus e lotações, além de requisitar táxis.
2. Taxistas: podem visualizar clientes que estão solicitando táxis e aceitar a requisição desses passageiros.
3. Cooperativas: podem visualizar a localização da sua frota de veículos e, ao receberem um telefonema de clientes que não utilizam o aplicativo,

---

[1] Telefones celulares com funcionalidades avançadas que podem ser estendidas através de programas executadas em seu sistema operacional.

[2] Dispositivo pessoal em formato de prancheta que pode ser utilizado para acessar a Internet e para recursos multimídia, como jogos e aplicativos.

podem sinalizar no mapa da aplicação a localização do cliente que está requisitando um táxi para que todos os taxistas vejam sua localização.

4. Empresas de ônibus e lotações: podem visualizar a localização da sua frota de veículos, além de ter o controle de horários e estados (em circulação, inativa, etc.) da mesma. Conseguem realizar o cadastro de novas linhas e editar o trajeto de alguma linha existente.

Neste projeto foi desenvolvido um sistema para utilização em dispositivos móveis (para passageiros e motoristas de táxi/ônibus/lotações) com sistema operacional *Android* e computadores (para a cooperativa de táxis e empresas de ônibus e lotações) com o intuito de proporcionar um melhor controle do sistema de transporte público, uma maior visibilidade dos meios de transporte pelos usuários e uma grande praticidade e rapidez na realização de chamadas de táxis.

1.4 ORGANIZAÇÃO DO TEXTO

O restante deste documento está organizado nos seguintes capítulos:

2. **Fundamentação Teórica:** Apresenta o embasamento teórico para as ideias apresentadas nesta introdução.
3. **Arquitetura do Sistema:** Padrões e tecnologias que foram utilizados ao longo do desenvolvimento do sistema.
4. **Desenvolvimento do Sistema:** Apresenta um maior detalhamento do sistema proposto e da sua construção, assim como suas funcionalidades.
5. **Modelagem de Dados:** Apresenta a modelagem lógica do banco de dados do sistema.
6. **Trabalhos Relacionados:** Apresenta trabalhos relacionados ao tema do sistema.
7. **Conclusão:** Apresenta as conclusões adquiridas com o desenvolvimento do trabalho, assim como propostas de trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Neste capítulo são apresentados assuntos que foram explorados durante o desenvolvimento do sistema.

2.1 TECNOLOGIAS MÓVEIS

O grande crescimento ocorrido ao longo dos últimos anos nas áreas de tecnologias portáteis nos permite acreditar que num futuro próximo qualquer recurso possa ser acessado e utilizado de qualquer lugar e a qualquer momento. Hoje em dia, temos uma enorme disseminação de *smartphones* na sociedade, devido ao seu preço estar mais acessível do que outrora e às diversas funcionalidades que esses dispositivos proporcionam aos usuários. A computação móvel representa um novo paradigma computacional. Este novo paradigma permite que usuários desse ambiente tenham acesso a serviços independente de onde estão localizados, e o mais importante, de mudanças de localização, ou seja, mobilidade. Dessa forma, a computação móvel amplia o conceito tradicional de computação distribuída. Isso é possível graças à comunicação sem fio que elimina a necessidade do usuário manter-se conectado a uma infraestrutura fixa e, em geral, estática [1].

Cada vez mais os usuários estão procurando celulares com diversos recursos como câmeras, *bluetooth*, ótima interface visual, jogos, *GPS* e acesso a internet e e-mails. O mercado corporativo também está cada vez mais buscando incorporar aplicações móveis a seu dia-a-dia para agilizar seus negócios e integrar as aplicações móveis com seus sistemas de *back-end*. Por exemplo, diversos bancos já oferecem serviços que utilizam tecnologias móveis aos seus clientes, tais como realizar pagamentos de contas e visualizar o extrato da conta bancária diretamente do seu aparelho celular [2].

2.1.1 Sistema Operacional Android

O Android é uma plataforma de desenvolvimento para aplicativos móveis, baseada em um sistema operacional Linux. Foi idealizado pelo Google, entretanto existem outras empresas líderes do mercado de telefonia que formaram uma

aliança, lideradas pelo Google, com o intuito de padronizar uma plataforma de código aberto e livre para celulares. Esta aliança é chamada de *Open Handset Alliance* [22]. A grande vantagem do Android é que a plataforma é livre, de código aberto e a licença é flexível e permite que cada fabricante possa realizar alterações no código fonte do sistema para customizar seus produtos. O desenvolvimento de aplicações para Android foi idealizado para ser feito utilizando a linguagem de programação Java, entretanto não existe uma máquina virtual Java (JVM) no sistema operacional, em seu lugar existe uma máquina virtual chamada Dalvik [23], que é otimizada especificamente para execução em dispositivos móveis [2].

2.2 GPS

GPS é um sistema de posicionamento global de navegação por satélite que fornece precisão contínua da posição e informação da velocidade de usuários que tenham o equipamento receptor apropriado, além de fornecer dados a um número ilimitado de usuários receptores. O GPS disseminou uma forma de tempo universal coordenado (UTC). Existem dois sistemas de navegação por satélite em funcionamento: o Glonass russo e o GPS americano. Na sua concepção, o GPS seria apenas para uso militar, porém tornou-se disponível para o uso de pessoas comuns de forma gratuita. Ele fornece dois serviços: o serviço de posicionamento padrão (SPS) e o serviço de posicionamento preciso (PPS). O SPS é para uso da sociedade em geral, já o PPS é para uso de autoridades americanas selecionadas [3].

2.3 WEB SERVICES

Web services são interfaces que descrevem uma coleção de operações que estão acessíveis na rede através do padrão XML para troca de mensagens [4]. Um *web service* é um pedaço de lógica de negócio, localizado em algum lugar da Internet, que é acessível através de protocolos da Internet, tais como HTTP ou SMTP. Ele tem como características [5]:

- Baseado em XML: possibilita uma forma de linguagem neutra para representar os dados. Usando XML como a camada de representação

dos dados para todos os protocolos de web services e tecnologias que são criadas, estas tecnologias podem ser interoperáveis.

- Baixo acoplamento: um consumidor de um web service não está vinculado diretamente ao web service, ou seja, a interface do web service pode mudar ao longo do tempo sem comprometer a capacidade do cliente de interagir com o serviço. Adotando esta arquitetura de baixo acoplamento, possibilita-se fazer programas que permitam integração com diferentes sistemas.
- Capacidade de ser síncrono ou assíncrono: Em chamadas síncronas, o cliente bloqueia e espera pela resposta do serviço para completar sua operação antes de continuar. Já as operações assíncronas permitem ao cliente chamar um serviço e continuar outras funções. Nas operações assíncronas, os clientes têm seu resultado em um tempo futuro, enquanto nas operações síncronas os clientes têm seu resultado assim que o serviço terminar sua execução.

Muitas tecnologias estão sendo criadas para web services, porém três tecnologias primárias emergiram como padrões. São elas [5]:

- Simple Object Access Protocol (SOAP): SOAP encapsula as mensagens XML em pacotes. Define uma estrutura simples para fazer chamadas remotas a procedimentos: a troca de documentos. Tendo um mecanismo de transporte padrão, clientes heterogêneos e servidores podem se tornar interoperáveis.
- Web Service Description Language (WSDL): WSDL é uma tecnologia de XML que descreve a interface de um web service de uma forma padronizada. Ele padroniza como um web service representa os parâmetros de entrada e saída de uma chamada externa e a estrutura das funções. Permite que diferentes clientes automaticamente entendam como interagir com o web service.
- Universal Description Discovery and Integration (UDDI): UDDI fornece um registro mundial de web services para propósitos de propaganda, descoberta e integração. Analistas de negócio e de tecnologias utilizam UDDI para descobrir web services disponíveis procurando pelo nome,

identificador, categoria ou especificações implementadas pelo web service.

Na figura 1, apresentamos a arquitetura de um web service. Ela é composta por três elementos: serviço requisitante, serviço fornecedor e diretório de registro de serviços. O serviço requisitante solicita a execução de um serviço e deve ser capaz de localizar as descrições dos serviços. O serviço fornecedor provê um serviço e é responsável por publicar a descrição deste serviço. O diretório de registro de serviços é o meio através do qual a descrição de um web service é publicada e disponibilizada para localização.

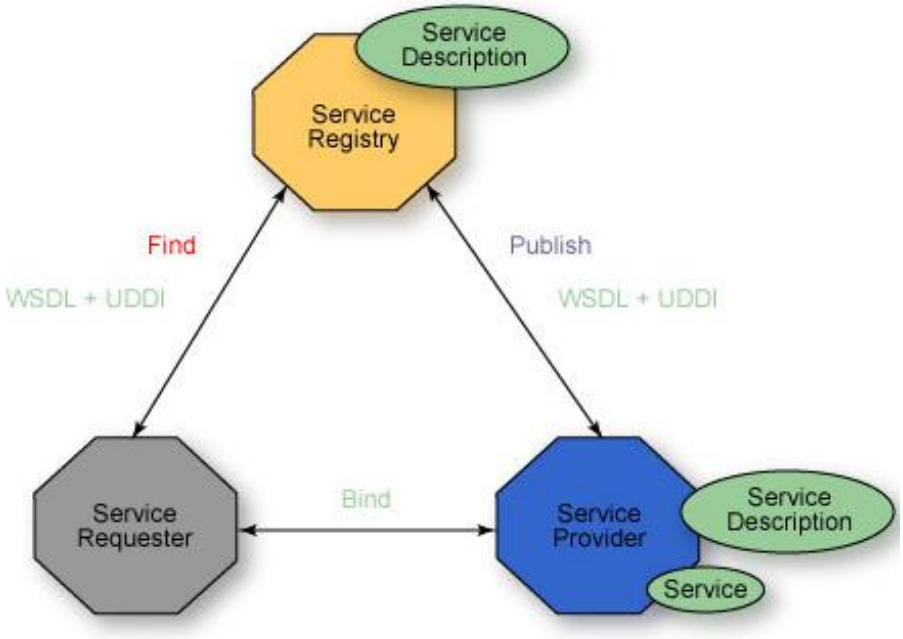

Figura 1: Arquitetura de um Web Service

Fonte: [6]

2.4 RECEBIMENTO DE ATUALIZAÇÕES

Ao desenvolver aplicações do tipo cliente-servidor, o fluxo padrão das requisições é caracterizado pelo cliente realizar a requisição ao servidor, que então retorna os dados solicitados. Entretanto, ao desenvolver aplicações onde o cliente tenha que ser notificado sobre alguma atualização ocorrida no servidor, é constatada a necessidade de que o servidor realize uma requisição para o cliente, ou então que, de alguma maneira, o cliente fique sabendo das atualizações. Sabendo que o sistema desenvolvido necessita que suas aplicações clientes recebam atualizações

do servidor, foram estudados alguns modos para manter o cliente sincronizado. São eles: *Pooling*, *Long Pooling* e *Streaming*.

2.4.1 Polling

A sincronização dos dados por *polling* caracteriza-se basicamente em realizar requisições de tempos em tempos ao servidor do sistema, sendo este tempo pré-determinado, com o intuito de verificar se existem novas atualizações a serem recebidas pela aplicação do cliente. Na figura 2, apresentaremos um exemplo de *polling*, onde um cliente faz requisições ao servidor automaticamente de acordo com um tempo pré-determinado.

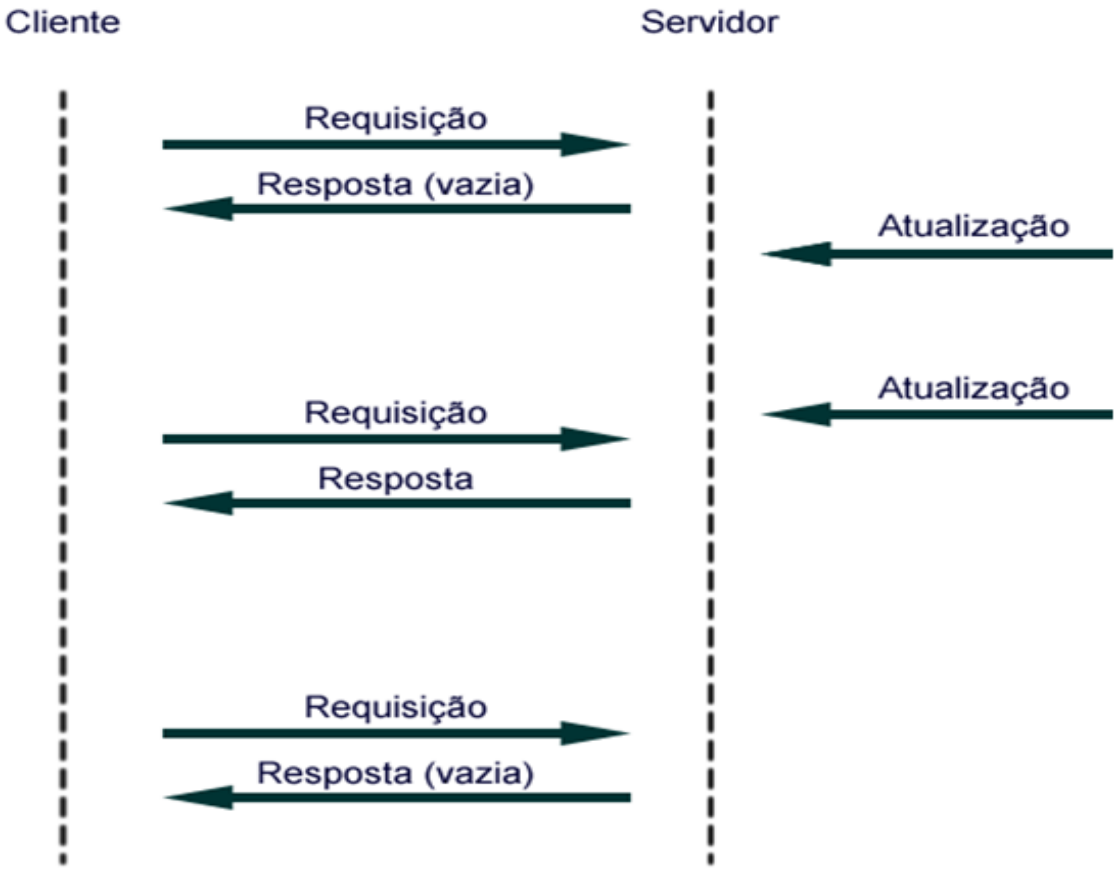

Figura 2: Exemplo de *Polling*

Vantagens:

- Facilidade de implementação se comparado às outras técnicas abordadas, pois na aplicação cliente basta manter-se em um laço de repetição realizando pedidos por atualizações, e na parte do servidor

deve-se apenas controlar as atualizações validando se elas devem ser informadas ao cliente.

Desvantagens:

- Quanto menor a frequência das requisições ao servidor, maior é o tempo que o cliente pode ficar sem informações atualizadas.
- Quanto maior a frequência das requisições ao servidor, mais requisições podem retornar sem nenhuma informação nova, ou seja, haverá um grande número de requisições inúteis ao servidor podendo causar a sobrecarga do mesmo.
- Para cada abertura de requisição, existe um custo computacional para o estabelecimento desta conexão.

O *polling* normalmente é utilizado em aplicações onde a atualização das informações não necessita ocorrer em tempo real. Exemplo: Sistemas de e-mail onde a busca por novos e-mails ocorre em um intervalo de tempo pré-determinado.

2.4.2 *Long Polling*

A sincronização dos dados por *long polling* é uma técnica que permite a realização de requisições que não serão respondidas imediatamente pelo servidor. As requisições tendem a ficar abertas até que o servidor receba alguma atualização que seja útil ao cliente. Na figura 3, apresentaremos um exemplo de *long polling*.

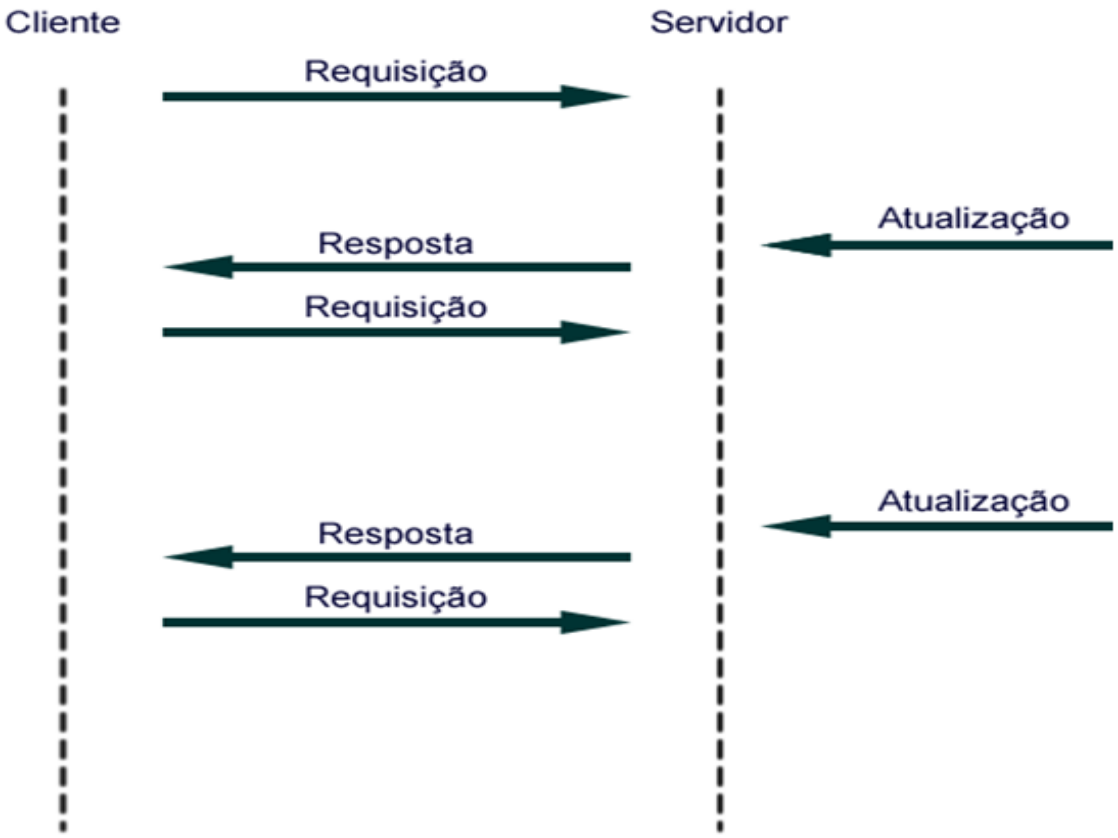

Figura 3: Exemplo de *Long Polling*

Vantagens:

- Redução de chamadas desnecessárias com relação à técnica de *polling*.
- É possível que ocorra o recebimento de atualizações em tempo real, pois imediatamente após receber uma atualização o servidor encaminha a resposta para o requisitante.

Desvantagens:

- Caso o servidor tenha uma grande quantidade de atualizações por segundo, o *long polling* irá caracterizar-se como um *polling* de alta frequência, podendo causar uma sobrecarga ao servidor.
- Para cada abertura de requisição, existe um custo computacional para o estabelecimento desta conexão.

A técnica de long polling normalmente é utilizada em sistemas que necessitam receber atualizações em tempo real, onde a ocorrência destas atualizações não tenha uma frequência muito alta. Exemplo: Sistemas de chat onde um usuário recebe a mensagem imediatamente após o envio de outro usuário.

2.4.3 Streaming

A sincronização baseada em *streaming* é uma técnica que tem como objetivo abrir uma requisição e não fechá-la imediatamente após receber alguma resposta, ou seja, reutilizar a mesma requisição mesmo após o recebimento de uma resposta do servidor. Isto possibilita receber pacotes a qualquer momento de acordo com as atualizações ocorridas no servidor. Na figura 4, apresentaremos um exemplo de *streaming*.

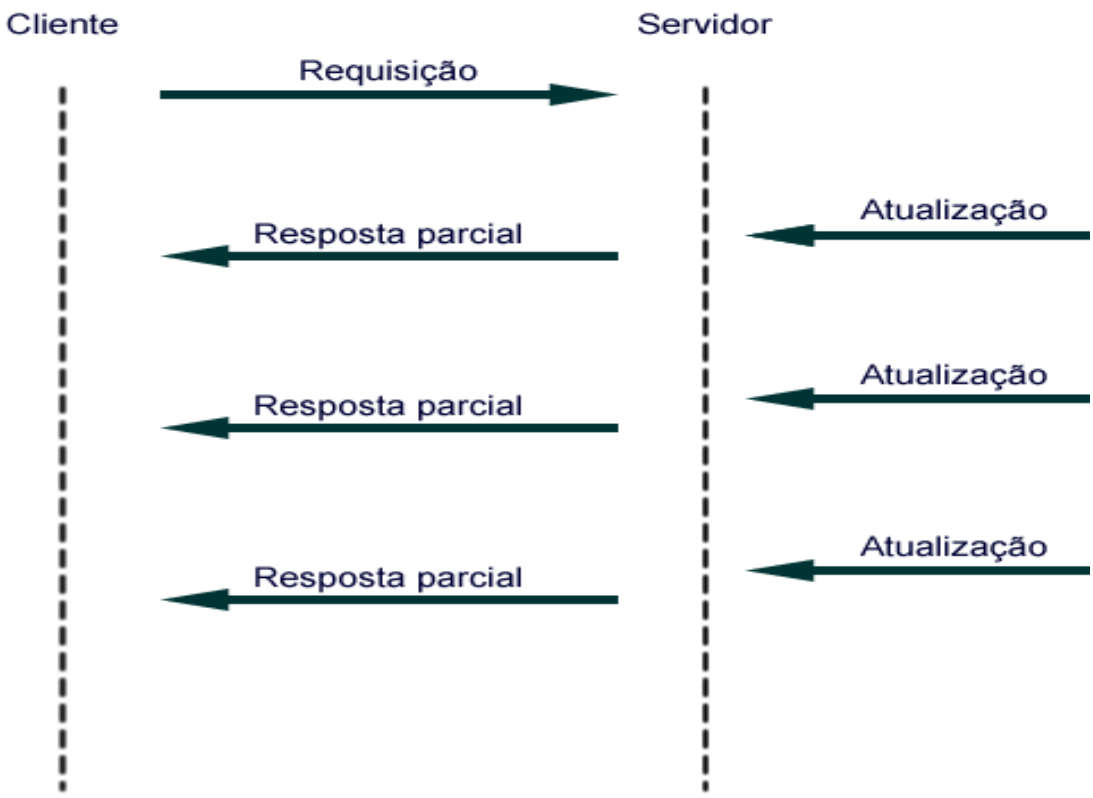

Figura 4: Exemplo de *Streaming*

Vantagens:

- Redução de *overhead* causado pela abertura de diversas requisições em relação às outras técnicas abordadas.
- Possibilita o recebimento de atualizações em tempo real.

Desvantagens:

- O tratamento de eventos da conexão estabelecida entre o cliente e o servidor, tais como quedas de conexão, acentuado pelo fato de estarmos lidando com dispositivos móveis, dificulta muito a implementação desta técnica.

A técnica de *streaming* é normalmente utilizada em sistemas que necessitam de atualizações em tempo real, onde a ocorrência destas atualizações tenha uma frequência muito alta. Exemplo: Sistemas de visualização de dados de bolsas de valores.

3 ARQUITETURA DO SISTEMA

Neste capítulo serão abordados tecnologias e padrões de projetos utilizados no desenvolvimento do sistema, além da maneira que ele foi estruturado. O sistema é do tipo cliente-servidor e, para esse tipo de sistema, uma boa prática é a utilização de Web Services, de forma que o servidor disponibilizará serviços para as aplicações clientes através de contratos/interfaces que devem ser implementados. Na figura 5, apresentaremos uma visão macro de como os pacotes de classes e componentes estão distribuídos no projeto.

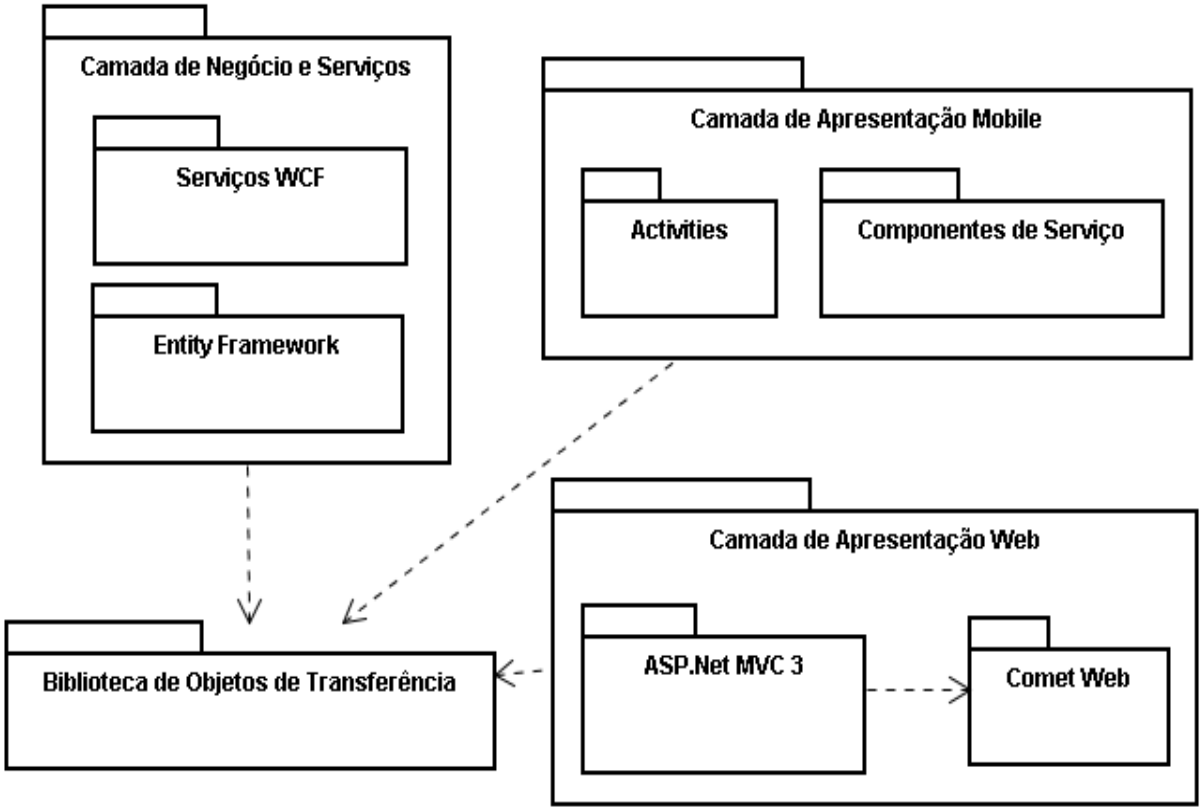

Figura 5: Organização em Pacotes do Sistema

3.1 CAMADA DE NEGÓCIO E SERVIÇOS

A camada de negócio e serviços é responsável pela disponibilização das interfaces dos serviços e a implementação dos mesmos, tanto para as aplicações móveis quanto para o sistema web. Nesta camada são feitas a validação das regras de negócio e a persistência de dados do sistema. A tecnologia mais relevante que será utilizada para o desenvolvimento dessa camada é o WCF, utilizando o

protocolo REST, juntamente com o Entity Framework para realizar o acesso à base de dados.

3.1.1 Serviços WCF

O WCF é uma tecnologia desenvolvida pela Microsoft feita para oferecer uma abordagem viável e segura de programação distribuída, com ampla interoperabilidade e suporte direto à orientação a serviços [15]. Dentro do escopo do sistema que desenvolvemos, o WCF é utilizado para prover acesso à implementação das lógicas de negócio, disponibilizando os serviços pelos protocolos SOAP e REST.

REST define um conjunto de princípios de arquitetura pelos quais é possível desenvolver Web Services que focam em um recurso do sistema, incluindo como esses recursos são abordados e transferidos via HTTP para uma ampla gama de clientes escritos em diversas linguagens [16]. Os serviços REST foram implementados para consumo via JSON com o intuito de proporcionar maior desempenho, menor overhead e maior facilidade de implementação com relação aos serviços em SOAP [17], pois, no sistema que desenvolvemos, os serviços baseados em REST são consumidos pelas aplicações para dispositivos móveis Android, que não possuem suporte nativo ao SOAP.

JSON é um formato de transferência de dados leve, que é de fácil manipulação pelos computadores [18].

3.1.2 Entity Framework

Ao desenvolver um sistema orientado a objetos que realize acesso a bancos de dados relacionais, um problema frequentemente encontrado é a diferença dos modelos de representação de dados. Para facilitar o desenvolvimento desse tipo de sistema, o Entity Framework provê uma camada transparente de acesso a dados que abstrai do desenvolvedor a necessidade de conhecer o modelo relacional que está trabalhando [19].

Dentro do sistema desenvolvido, o Entity Framework foi utilizado como uma camada de acesso ao banco de dados de maneira orientada a objetos. Para realizar

a configuração dessa camada, foi necessário apenas informar ao Entity o esquema físico dos dados, para que a partir do mesmo, fossem geradas todas as classes necessárias para acesso aos dados.

3.2 CAMADA CLIENTE

A camada cliente tem como responsabilidade fornecer ao usuário uma interface para o acesso aos serviços disponibilizados pela camada de serviços. No sistema desenvolvido, dividimos essa camada em três aplicativos, um aplicativo web que servirá para o gerenciamento de ônibus, lotações e táxis, e dois aplicativos para o sistema operacional *Android*, sendo uma para uso em táxis e ônibus, e outro para o usuário final. Ambos os aplicativos *Android* seguirão a mesma arquitetura.

3.2.1 Aplicação Web

A aplicação web foi implementada utilizando basicamente a tecnologia ASP.NET MVC 3 [21].

3.2.1.1 ASP.NET MVC 3

O MVC é um padrão que separa a modelagem do domínio, a apresentação, e as ações baseadas nas entradas do usuário em três classes distintas [20]:

- *Model*: Representa os dados que serão utilizados no *controller* e na *view*.
- *View*: Corresponde a maneira com que os dados do *model* serão apresentados.
- *Controller*: Realiza as ações necessárias baseadas no estado do *model*.

Na figura 6, apresentamos a relação das estruturas do MVC.

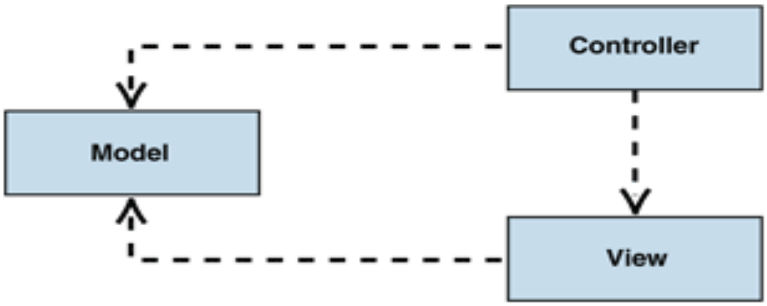

Figura 6: Diagrama de Representação do MVC

Fonte: [20]

O ASP.NET MVC 3 é um framework para construção de aplicações escaláveis, baseadas em padrões web e utilizando padrões de projetos bem estabelecidos [21].

No desenvolvimento da aplicação web, por utilizarmos o ASP.NET MVC 3, que é baseado no padrão MVC, temos os módulos *model*, *view* e *controller*. O módulo *controller* ficou responsável, além de receber as requisições e retornar as *views* (telas) correspondentes, pelas chamadas aos serviços do sistema.

3.2.2 Aplicações Android

As aplicações *Android* do sistema fazem uso da arquitetura padrão de desenvolvimento *Android*, que consiste em trabalhar com objetos do tipo *Activity* e layouts de telas em XML.

3.2.2.1 Activity

Uma *Activity* tem como função fornecer uma interface para que o usuário possa interagir com o sistema. É de responsabilidade da *Activity* controlar todos os comportamentos e ações pertinentes da tela a qual ela está atrelada, tais como realizar chamadas de serviços, validar dados e inicializar outras *Activities* [7]. Para que uma *Activity* construa a sua interface, são utilizados XMLs que ficam em um diretório específico para layouts. Esses XMLs descrevem os componentes que serão utilizados na tela, além de seus atributos e nomes de variáveis.

Na figura 7, apresentamos o clico de vida de uma *Activity*.

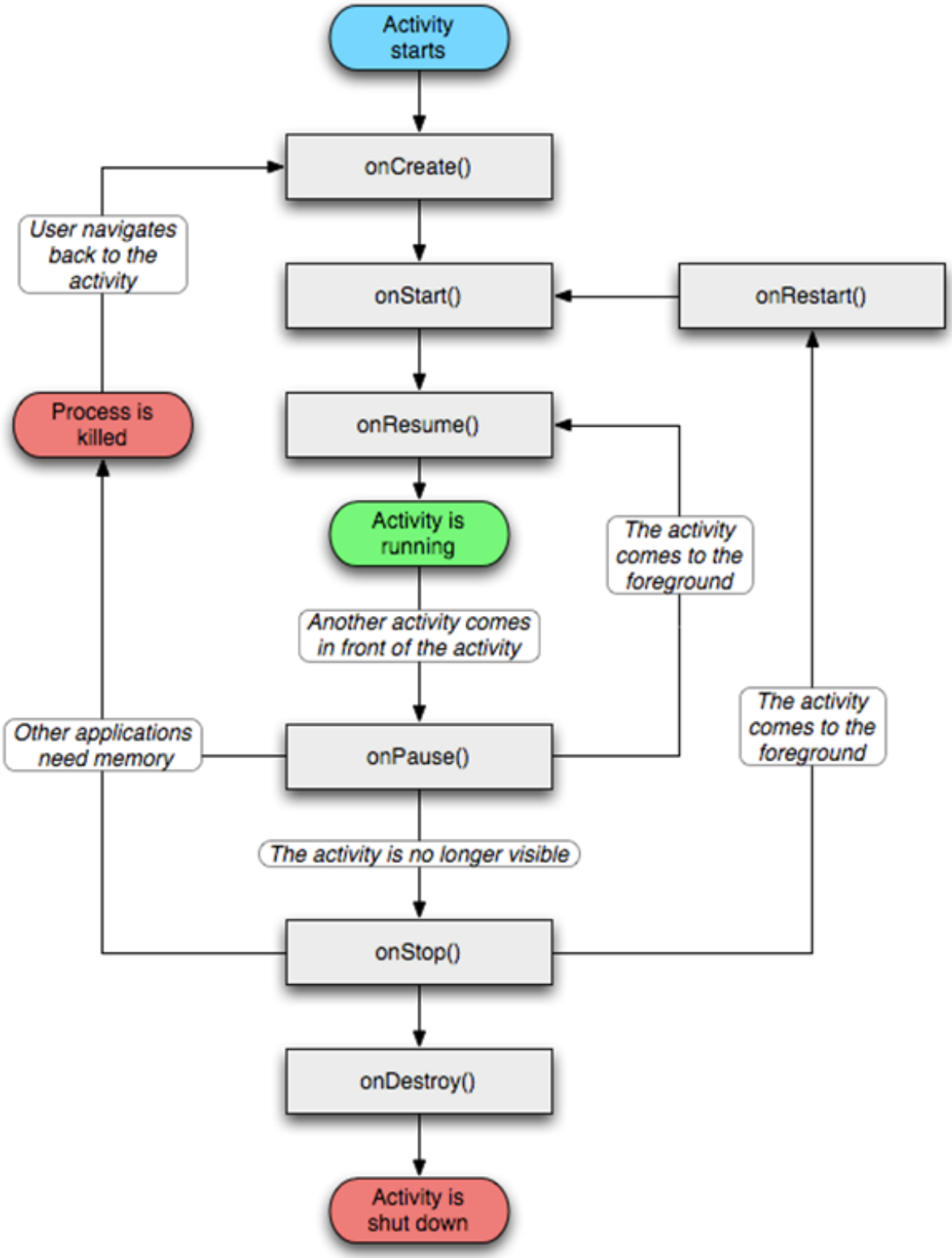

Figura 7: Ciclo de vida de uma *Activity*

Fonte: [7]

4 DESENVOLVIMENTO DO SISTEMA

Neste capítulo, abordaremos as funcionalidades do sistema e explicaremos o seu desenvolvimento.

O sistema foi divido em quatro módulos:

1. Aplicativo do usuário (passageiro)
2. Aplicativo do prestador de serviço
3. Aplicativo da empresa prestadora de serviços de transporte
4. Servidor do sistema

O diagrama abaixo exemplifica esta estrutura:

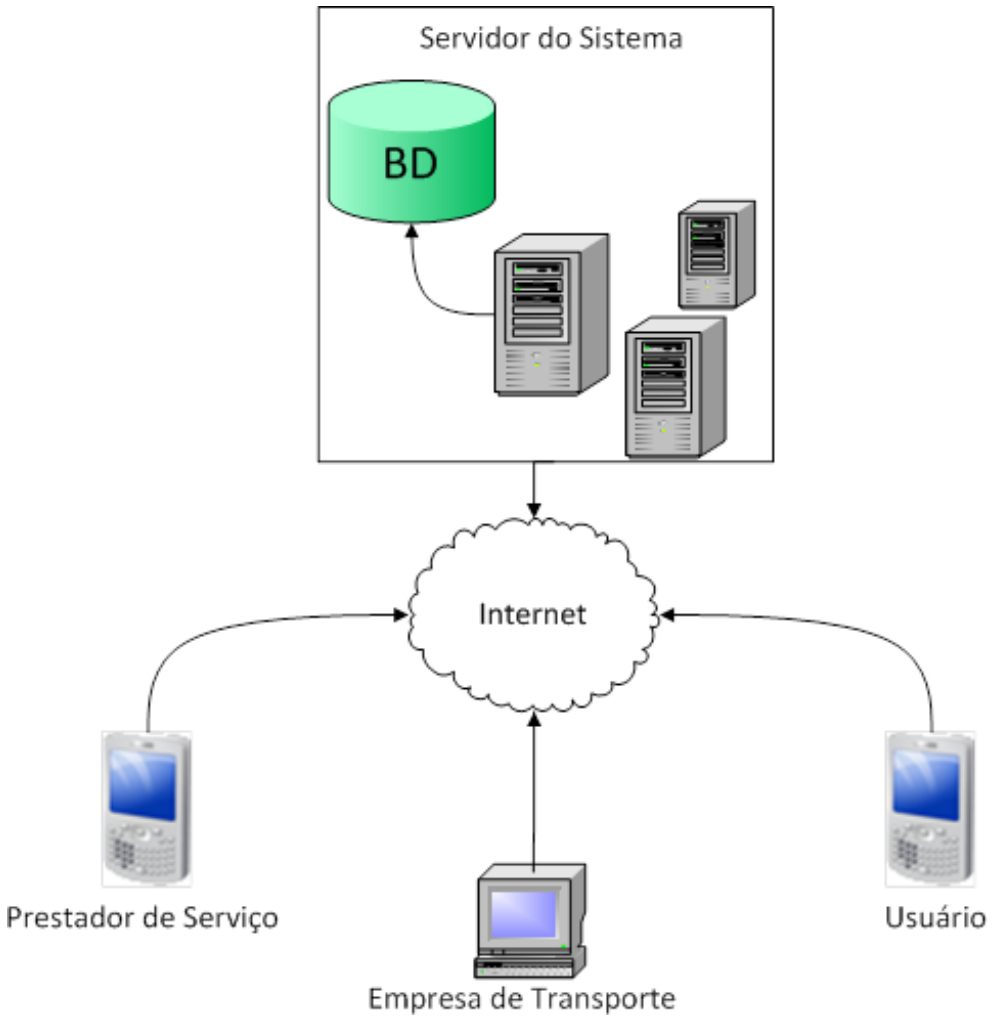

Figura 8: Estrutura do Sistema

4.1 APLICATIVO DO USUÁRIO

Este é o principal elemento desenvolvido, pois representa a parcela do sistema em que o usuário final da aplicação tem acesso. Este aplicativo é encarregado de fornecer ao usuário uma interface para as seguintes funcionalidades:

- Procurar um táxi disponível.
- Requisitar um táxi.
- Acompanhar o deslocamento do táxi requisitado até chegar a sua posição.
- Acompanhar a movimentação dos ônibus e lotações.
- Editar informações pessoais.
- Visualizar mapa.

Esta aplicação foi desenvolvida para que seja utilizada somente em dispositivos móveis com sistema operacional *Android*. A figura 9 apresenta o diagrama de casos de uso dessa aplicação.

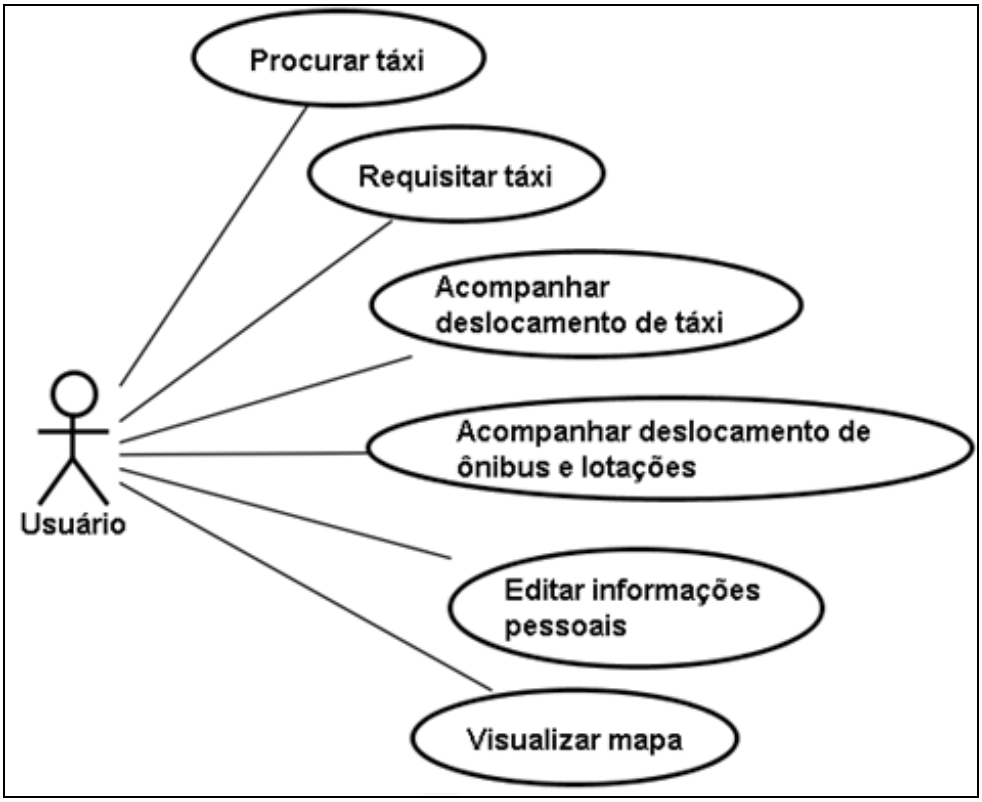

Figura 9: Diagrama de Casos de Uso da Aplicação do Passageiro

O apêndice A apresenta o detalhamento de casos de uso da aplicação do passageiro.

Ao iniciar esta aplicação é apresentada a tela principal, onde o usuário tem acesso às funcionalidades descritas acima. A figura 10 apresenta a tela principal.

Figura 10: Tela principal do Passageiro

4.1.1 Editar Informações

Ao acessar a opção “Perfil” da tela principal, o usuário tem acesso às suas informações pessoais. A figura 11 apresenta a tela de edição de informações pessoais.

Figura 11: Tela de Edição de Informações Pessoais

4.1.2 Visualizar Mapa

Ao acessar a opção “Ver Mapa” da tela principal, o usuário deverá optar por visualizar táxis ou ônibus/lotações. A figura 12 apresenta a tela de escolha de veículos a serem visualizados no mapa.

Figura 12: Tela de Escolha de Veículos a Serem Visualizados no Mapa

Ao escolher, por exemplo, “Ônibus/Lotação”, o usuário visualizará um mapa onde são apresentadas as localizações dos ônibus e lotações em circulação. A figura 13 apresenta um mapa com um ônibus em circulação.

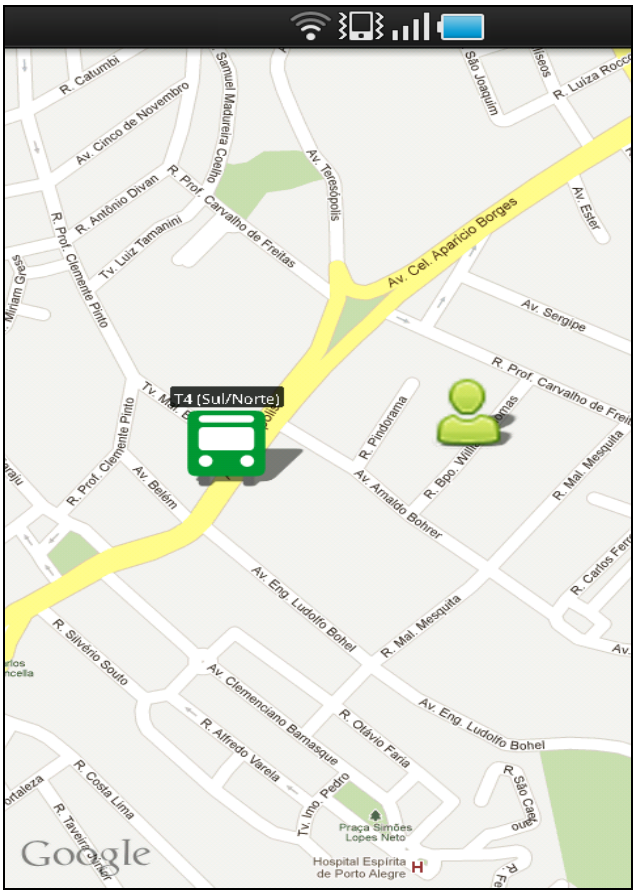

Figura 13: Mapa com um Ônibus em Circulação

4.1.3 Acompanhar Deslocamento de Ônibus e Lotações

Caso o usuário esteja visualizando o mapa para ônibus e lotações, ele tem a opção de acompanhar seu deslocamento automaticamente. Para isso, basta selecionar o veículo e clicar em “Acompanhar”. A figura 14 apresenta esta funcionalidade.

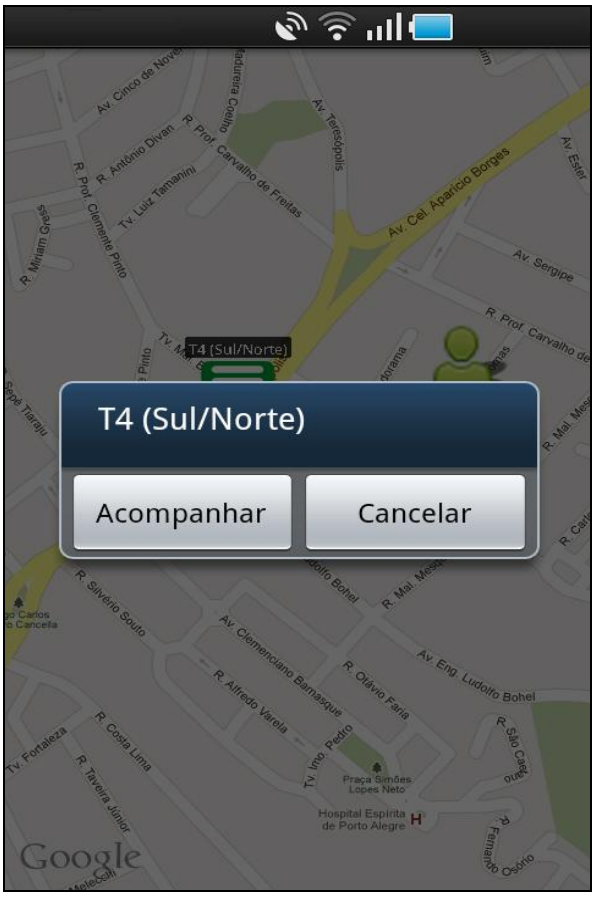

Figura 14: Acompanhar Deslocamento de Ônibus

4.1.4 Requisitar Táxi

Ao acessar a opção “Requisitar Táxi” da tela principal, o usuário consegue enviar uma requisição de táxi para todos os táxis do sistema. A figura 15 apresenta a tela de requisição de táxi.

Figura 15: Requisição de Táxi

Se o usuário clicar em “Sim”, ele tem a possibilidade de escrever uma pequena observação para os taxistas antes de enviar a requisição. Caso o usuário clique em “Alterar”, ele tem a possibilidade de corrigir o endereço capturado pelo GPS e também pode escrever uma observação para os taxistas antes de enviar a requisição. A figura 16 apresenta a tela de edição do endereço e observação.

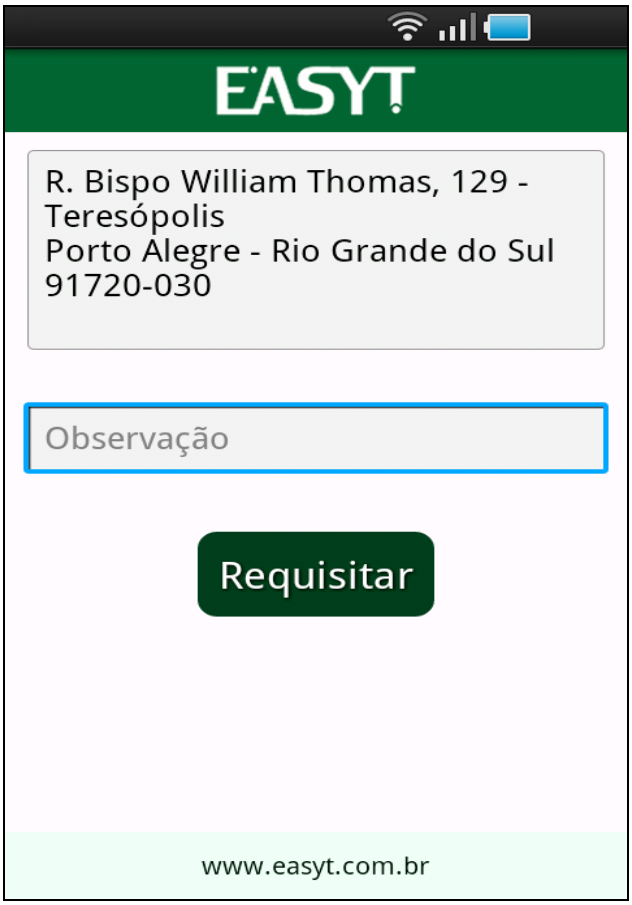

Figura 16: Tela de Edição do Endereço e Observação

Ao realizar a requisição o usuário fica no máximo sessenta segundos aguardando por uma resposta. A figura 17 apresenta a tela de espera do usuário enquanto ele aguarda uma resposta.

Figura 17: Tela de Espera do Passageiro

4.1.5 Procurar Táxi

Caso o usuário esteja visualizando o mapa para táxis, ele tem a opção de requisitar um táxi específico que esteja em seu campo de visão do mapa. Ao selecionar o táxi desejado, o usuário visualiza as informações do taxista e tem a opção de requisitá-lo. A figura 18 apresenta a tela com o táxi no campo de visão do cliente e a tela após o cliente clicar no veículo.

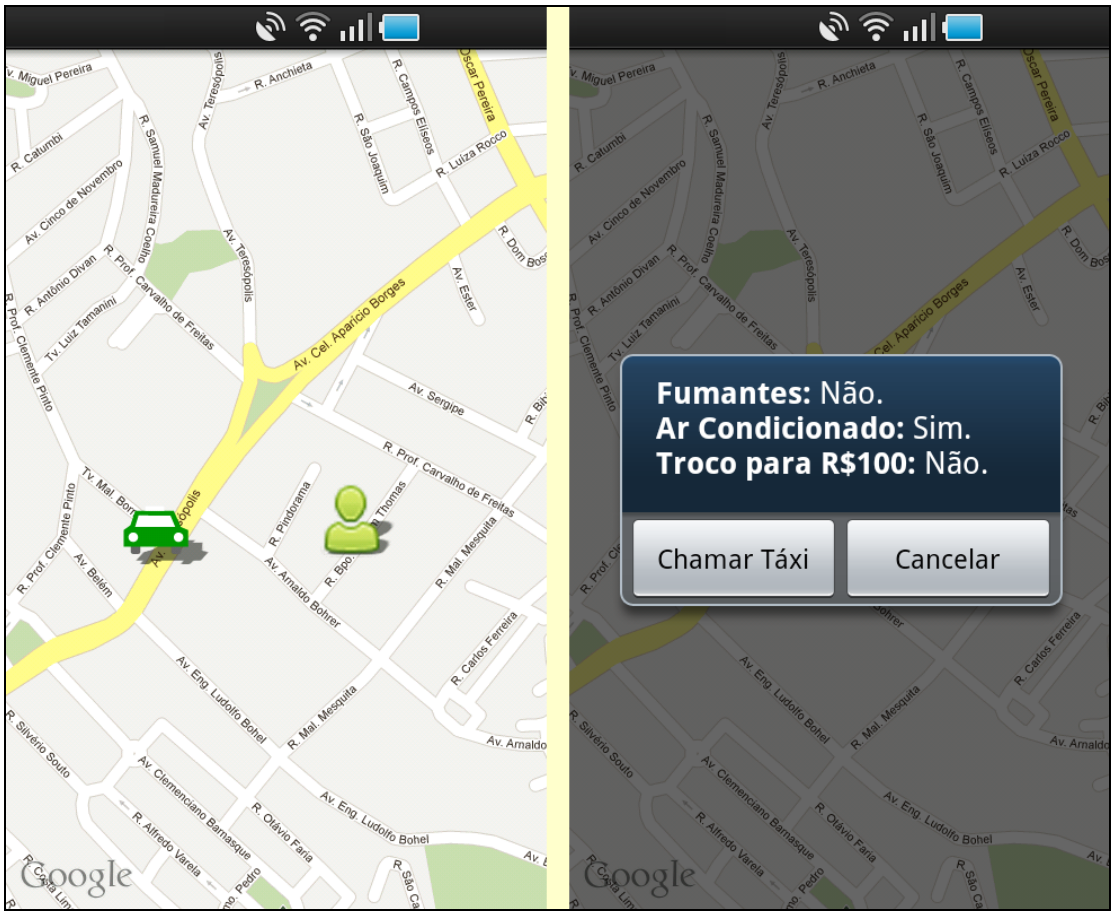

Figura 18: Requisição de Táxi Específico

4.1.6 Acompanhar Deslocamento de Táxi

Quando o passageiro tem uma requisição de táxi aceita por um taxista, é iniciado o acompanhamento automático do deslocamento do veículo até a posição onde se encontra o usuário.

4.2 APLICATIVO DO PRESTADOR DE SERVIÇOS DE TRANSPORTE

Este aplicativo é a parcela da aplicação cuja responsabilidade é possibilitar ao prestador de serviço de transporte a comunicação com os usuários, ou seja, possibilitar realizar as seguintes funcionalidades:

- Aceitar uma requisição de táxi.
- Visualizar posição do passageiro a ser pego.
- Editar informações pessoais.

- Alterar seu status no sistema.
- *Login* no sistema.

Esta aplicação foi desenvolvida para que seja utilizada somente em dispositivos móveis com sistema operacional *Android*. A figura 19 apresenta o diagrama de casos de uso dessa aplicação.

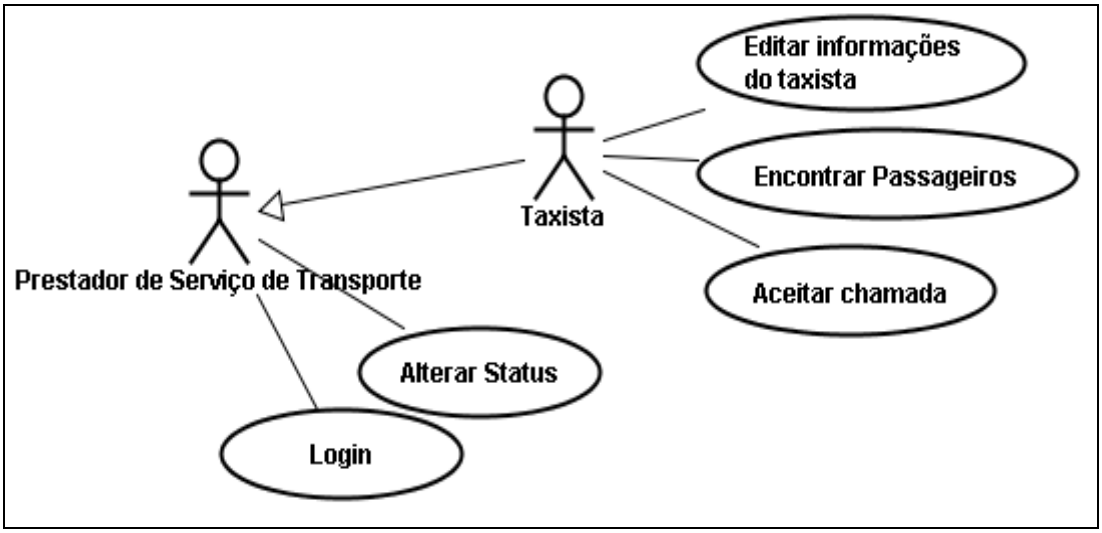

Figura 19: Diagrama de Casos de Uso da Aplicação do Prestador de Serviços de Transporte

O apêndice B apresenta o detalhamento dos casos de uso da aplicação do prestador de serviço de transportes.

Ao iniciar esta aplicação, o motorista do veículo necessita realizar o *login* no sistema. Para isto ele deverá informar seu nome, senha e o código do veículo que ele está utilizando. A figura 20 apresenta a tela de *login*.

Figura 20: Tela para Realizar Login na Aplicação do Prestador de Serviços de Transporte

Ao realizar o login com sucesso é apresentada a tela principal da aplicação, onde o usuário tem acesso às funcionalidades descritas acima. A figura 21 apresenta a tela principal para um taxista.

Figura 21: Tela Principal do Taxista

A figura 22 apresenta a tela principal para um motorista de ônibus ou lotação.

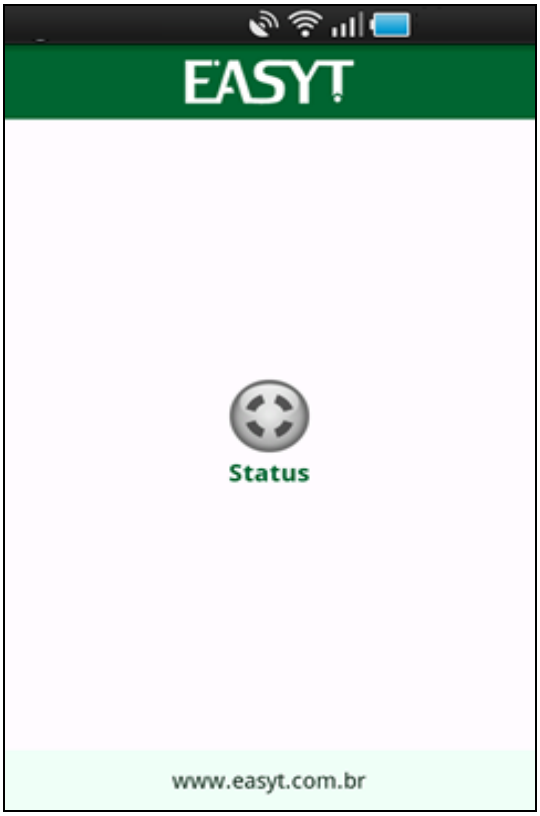

Figura 22: Tela de Principal do Motorista de Ônibus ou Lotação

4.2.1 Alterar Status

Ao acessar a opção “Status” da tela principal, o usuário pode escolher seu status atual:

- Disponível, ocupado ou invisível para população: status possíveis de um táxi.
- Disponível ou invisível para população: status possíveis de ônibus ou lotação.

Esta funcionalidade é apresentada para todos os prestadores de serviços de transporte. A figura 23 apresenta a execução de uma alteração de status de “Invisível para População” para “Disponível”.

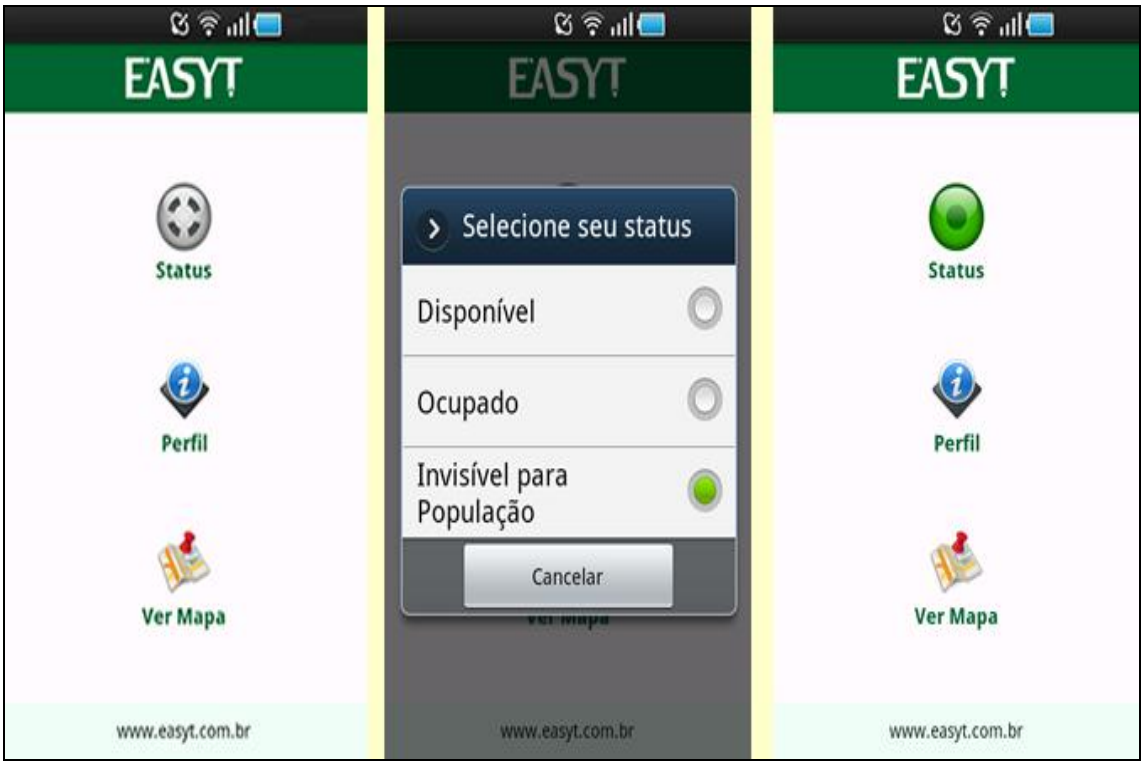

Figura 23: Alteração de Status

4.2.2 Editar Informações

Ao acessar a opção “Perfil” da tela principal, o usuário tem acesso às informações que ele pode disponibilizar ao passageiro. Esta funcionalidade é apresentada somente para taxistas. A figura 24 apresenta a tela de edição de informações.

Figura 24: Tela de Edição de Informações

4.2.3 Encontrar Passageiros

Ao acessar a opção “Ver Mapa” da tela principal, o usuário visualizará um mapa onde são apresentadas as localizações dos passageiros que estão requisitando um táxi. Ao clicar sobre o passageiro que está no mapa, o taxista visualiza o endereço e uma observação adicional que o passageiro pode ter escrito. Ele tem a opção de aceitar ou cancelar a requisição do cliente. A figura 25 apresenta um mapa com os passageiros e a tela que é apresentada quando o taxista clica em um determinado cliente.

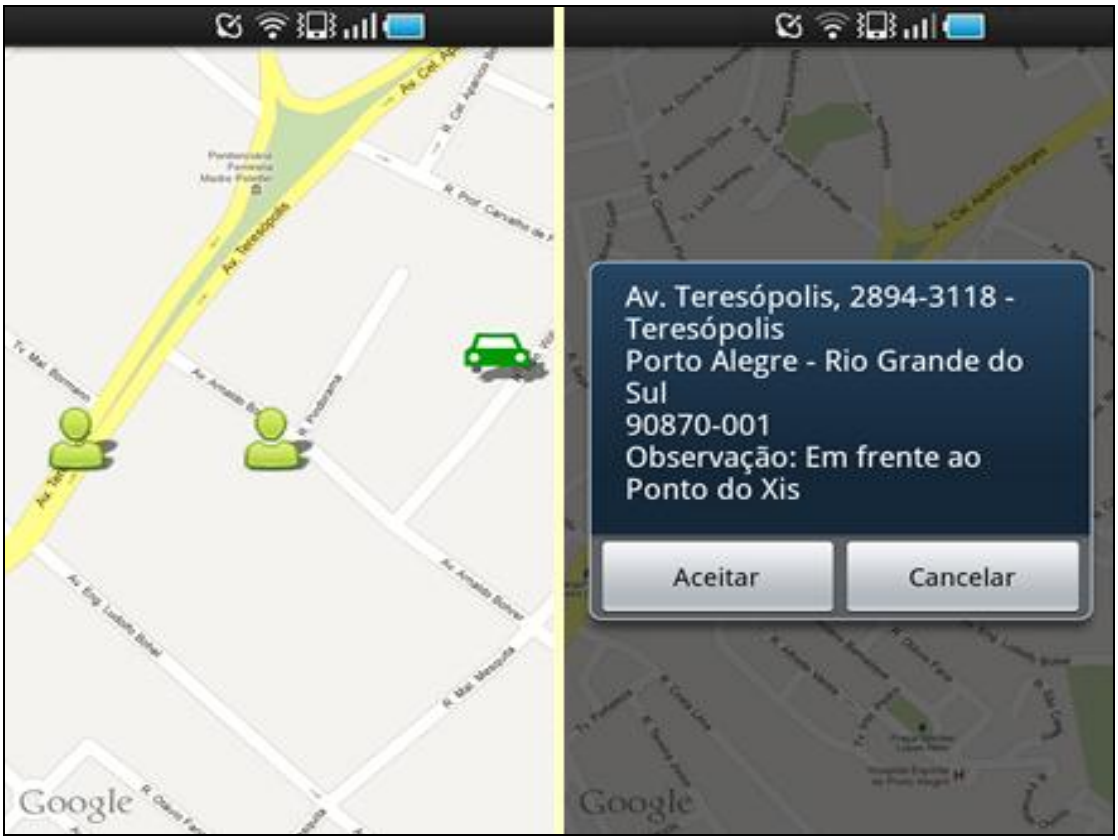

Figura 25: Encontrar Passageiros

4.2.4 Aceitar Chamada

Caso algum passageiro solicite um taxi específico, o taxista receberá uma notificação informando que ele está sendo chamado. Ao clicar nesta notificação o taxista visualiza o endereço e uma observação adicional que o passageiro pode ter escrito. Ele tem a opção de aceitar ou cancelar a chamada do cliente. Se o taxista não responder a chamada em 1 minuto ela será cancelada. A figura 26 apresenta a notificação recebida pelo taxista.

Figura 26: Notificação de Recebimento de Chamada

4.3 APLICATIVO DA EMPRESA PRESTADORA DE SERVIÇOS DE TRANSPORTE

Este aplicativo possibilita às empresas prestadoras de serviço de transporte gerenciar o uso do sistema com relação aos seus veículos e motoristas. A empresa pode usufruir das seguintes funcionalidades:

- Gerenciar funcionários.
- Gerenciar veículos.
- Gerenciar pontos de ônibus (paradas).
- Atualizar informações sobre as linhas de ônibus/lotações oferecidas.
- Visualizar no mapa o posicionamento e status de seus veículos credenciados.
- Realizar *login*.

Esta aplicação foi desenvolvida para que seja compatível com os navegadores de internet atuais. A figura 27 apresenta o diagrama de casos de uso dessa aplicação.

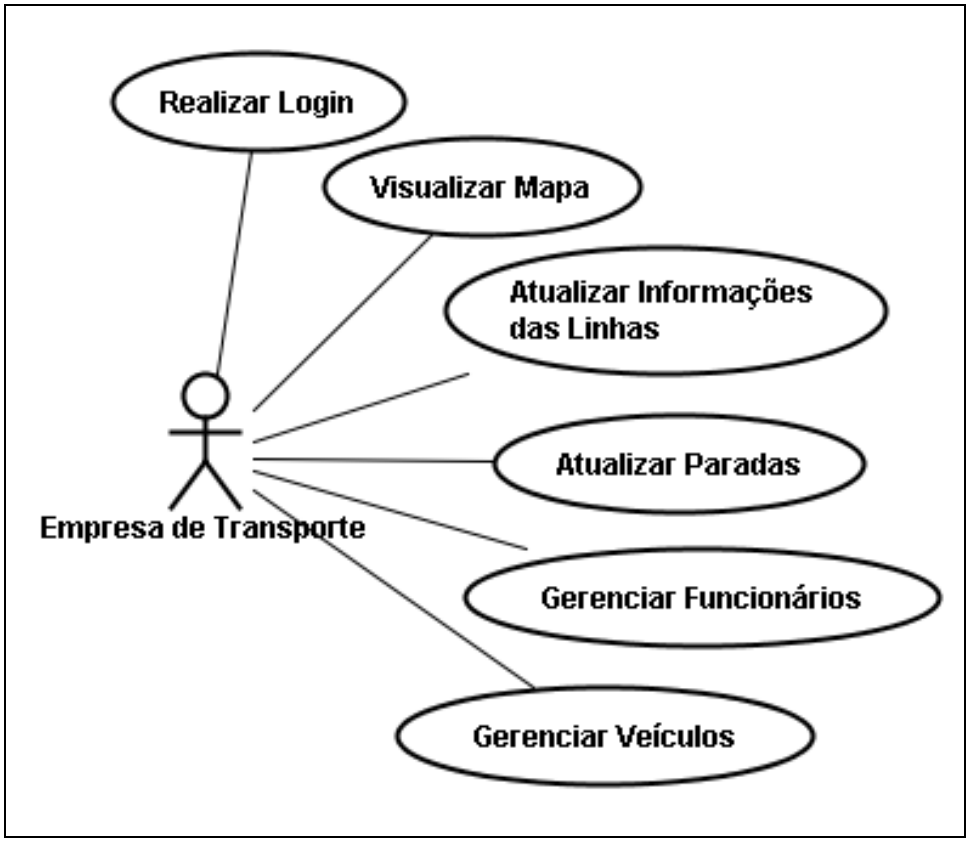

Figura 27: Diagrama de Casos de Uso da Aplicação da Empresa Prestadora de Serviços de Transporte

O apêndice A apresenta o detalhamento dos casos de uso da aplicação da empresa prestadora de serviços de transporte.

Ao iniciar esta aplicação, o usuário necessita realizar o *login* no sistema. Para isto ele deverá informar seu nome e senha. Após realizar o *login* com sucesso, é apresentada a tela inicial do sistema. A figura 28 apresenta a tela inicial da aplicação web.

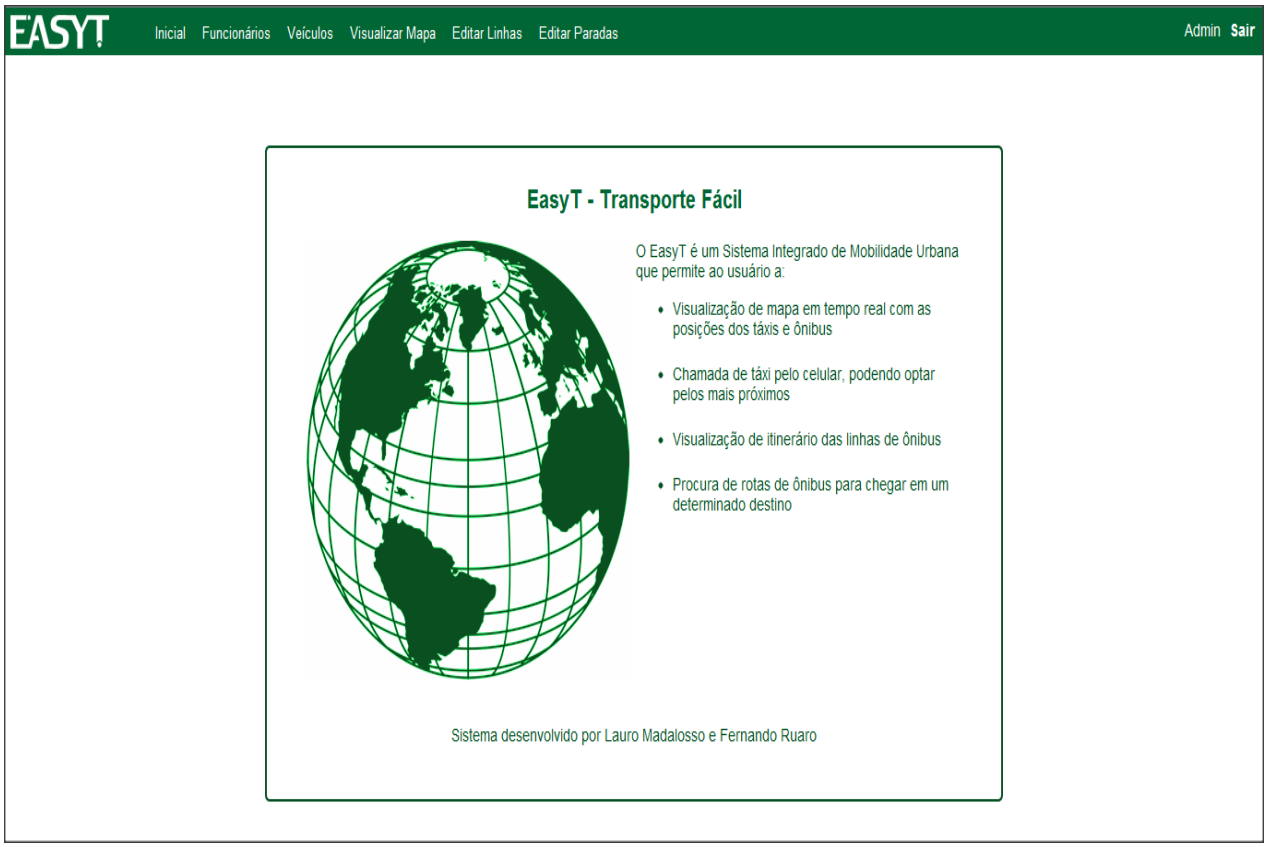

Figura 28: Tela Inicial da Aplicação da Empresa Prestadora de Serviços de Transporte

4.3.1 Gerenciar Funcionários

Ao acessar a opção de menu “Funcionários”, o usuário pode gerenciar a lista de funcionários da empresa, podendo incluir, editar ou excluir empregados. Na figura 29, apresentamos a tela de cadastro de um funcionário.

Figura 29: Tela de Cadastro de Funcionário

4.3.2 Gerenciar Veículos

Ao acessar a opção de menu “Veículos”, o usuário pode visualizar uma listagem dos veículos da empresa, podendo incluir, editar ou excluir veículos. Na figura 30, apresentamos a tela de listagem dos veículos.

Figura 30: Tela de Listagem dos Veículos

4.3.3 Gerenciar Pontos de Ônibus (Paradas)

Quando o usuário acessa a opção de menu “Editar Paradas”, é exibido a ele um mapa com todas as paradas do sistema. Nesta tela é permitido arrastar as paradas no mapa, atualizando sua posição, ou então realizar o cadastramento, edição ou deleção de novas paradas. Na figura 31, apresentamos essa funcionalidade.

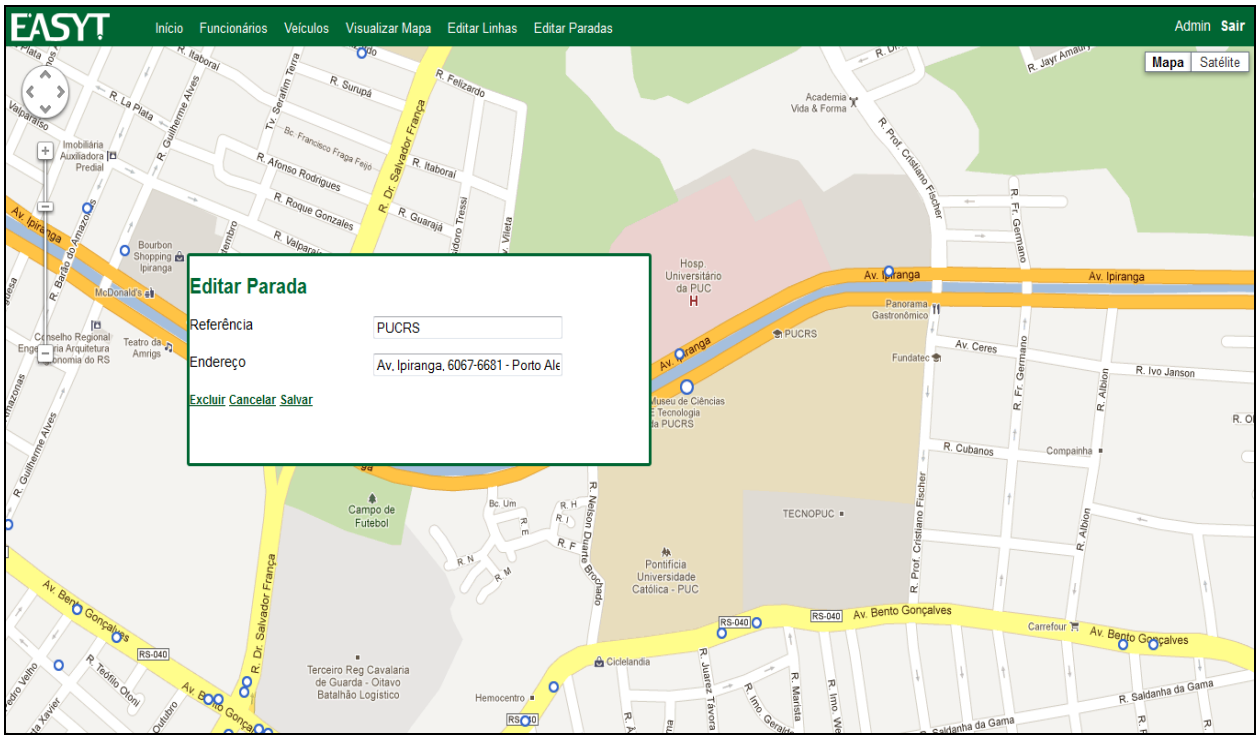

Figura 31: Tela de Edição de Paradas

4.3.4 Atualizar Informações das Linhas

Ao acessar a opção de menu “Editar Linhas”, é apresentado ao usuário a listagem de todas as linhas cadastradas no sistema e uma opção para cadastrar uma nova linha. A partir dessa tela é possível entrar nas funcionalidades de “Cadastrar Linha” e “Editar Linha”.

Tanto na funcionalidade de “Cadastrar Linha” quanto na de “Editar Linha”, o usuário deve seguir um fluxo de três etapas para concluir a edição/cadastramento:

- Etapa um: Edição do nome da linha.
- Etapa dois: Desenho do itinerário da linha em um mapa. Nessa tela é possível ao usuário cadastrar ou remover pontos, além de arrastar os pontos já existentes para uma nova posição.

Na figura 32, apresentamos a tela da etapa dois.

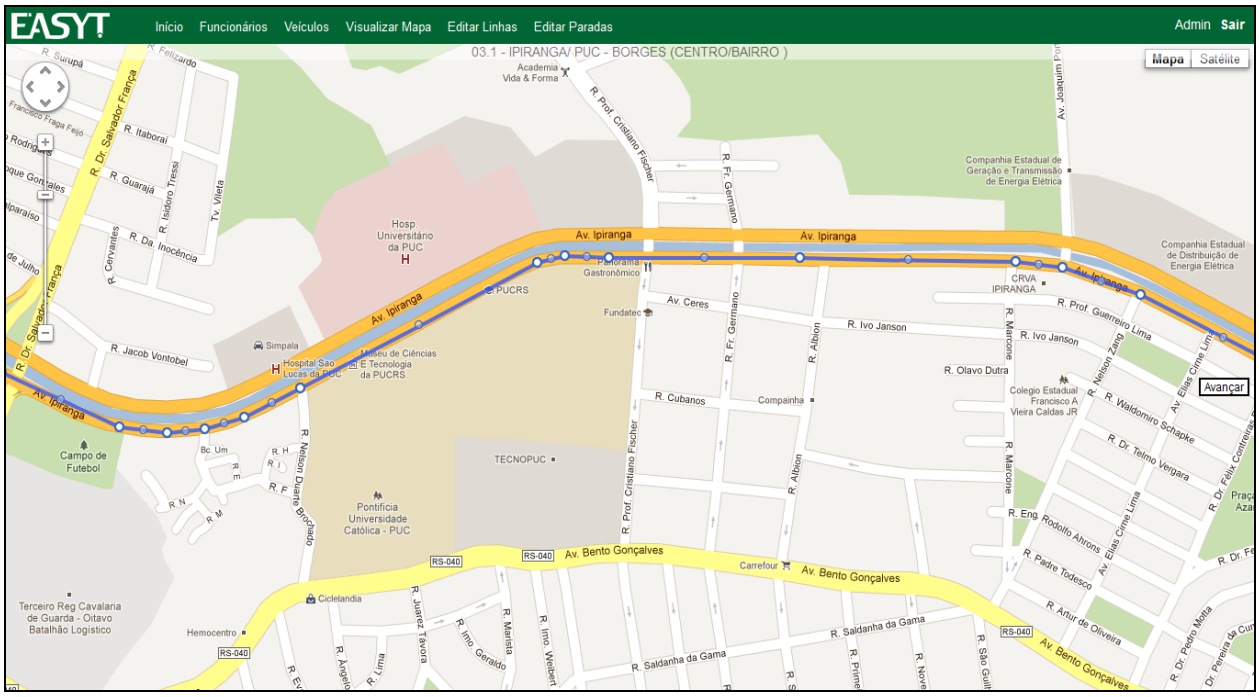

Figura 32: Tela de Atualização/Cadastro de Linhas

- Etapa três: Seleção das paradas que a linha passa. Nessa tela, a linha desenhada na etapa dois e as paradas cadastradas no sistema são exibidas, possibilitando ao usuário marcar ou desmarcar as paradas indicando se a linha passa ou não na respectiva parada.

Na figura 33, apresentamos a tela da etapa três.

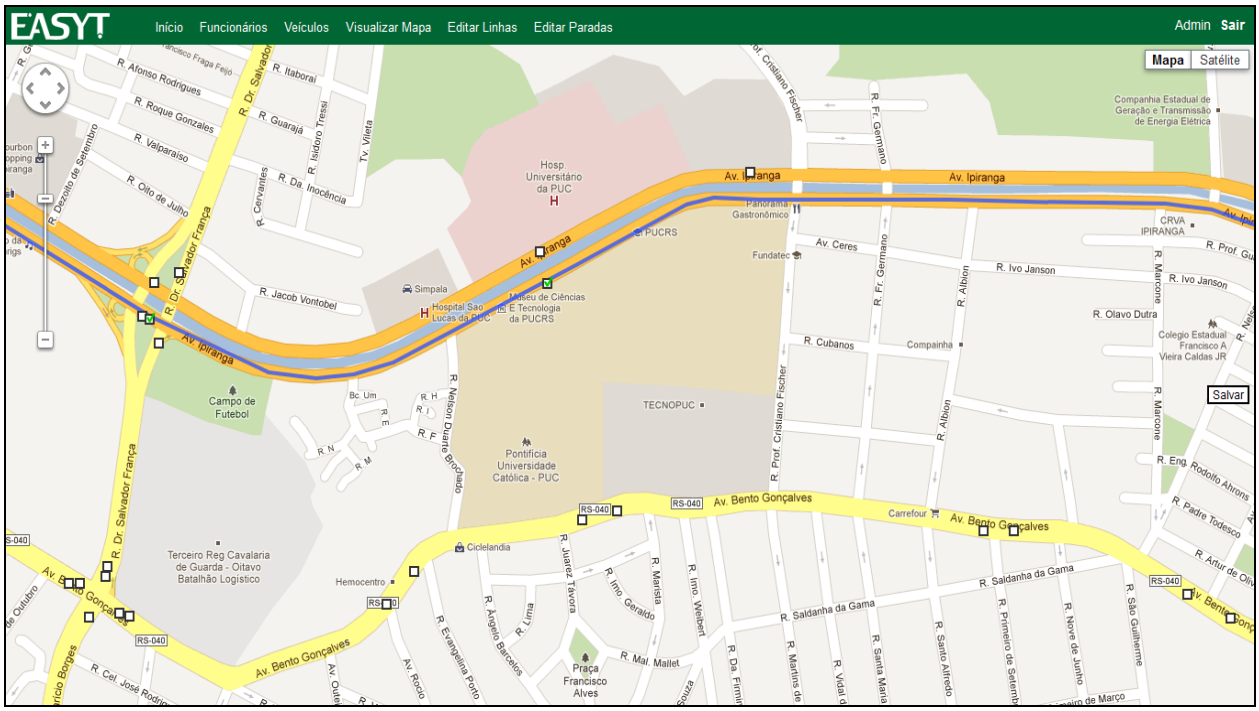

Figura 33: Tela de Seleção de Paradas

4.3.5 Visualizar Mapa

Ao acessar a opção de menu “Visualizar Mapa”, o usuário é redirecionado para uma página que possui um mapa e a lista de veículos da empresa. Nesta tela é possível selecionar quais os veículos que se deseja realizar o acompanhamento no mapa. Isto permite ao usuário verificar em que posição e estado se encontram os veículos.

A partir da seleção do veículo, o sistema passa a receber atualizações temporárias com as coordenadas do veículo e seu respectivo estado. A técnica de atualização utilizada nessa tela foi a de long polling, sendo que o long polling foi customizado para ficar no mínimo 2 segundos sem retornar informações visando evitar a sobrecarga do servidor caso haja constantes atualizações.

Na figura 34, apresentamos a tela que representa essa funcionalidade no sistema.

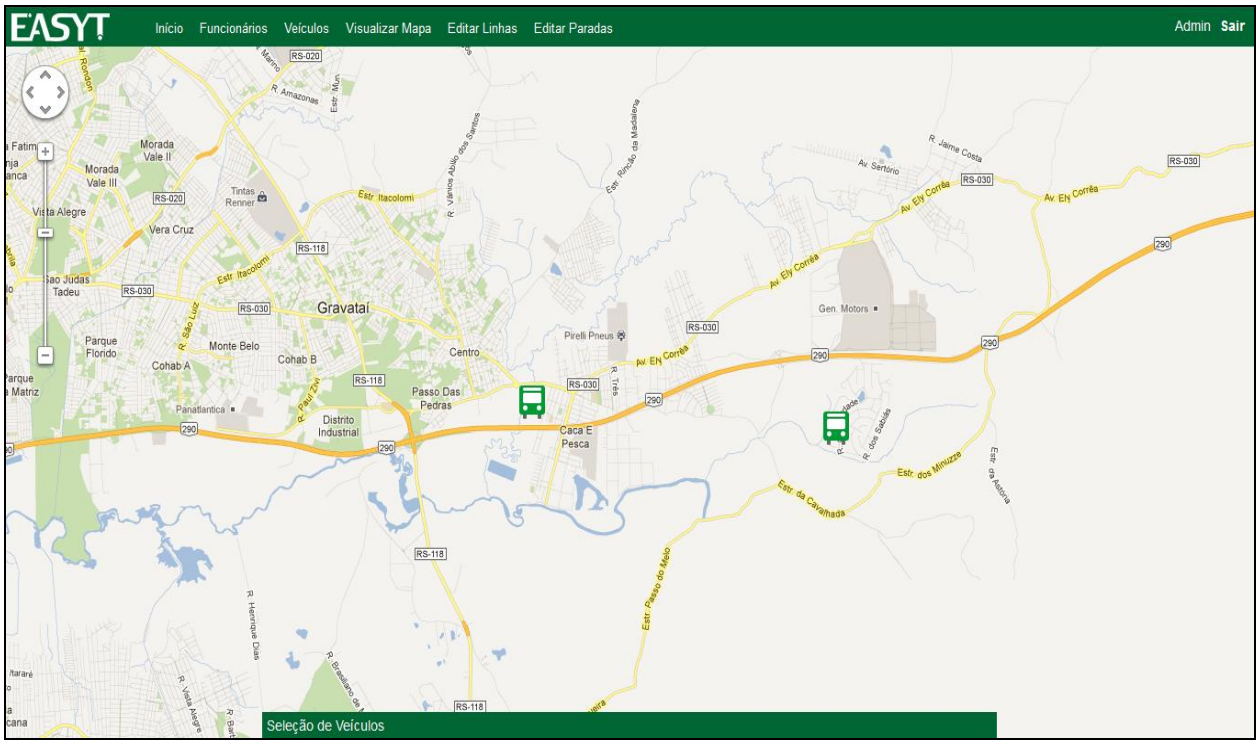

Figura 34: Tela de Acompanhamento dos Veículos

4.4 SERVIDOR DO SISTEMA

Esta parcela do sistema é responsável por proporcionar o acesso a grande parte das funcionalidades do sistema, pois hospedará os serviços do projeto. É o servidor que realiza o acesso ao banco de dados e controla a persistência dos dados do sistema. Os serviços que foram implementados estão divididos da seguinte maneira:

- Serviço de autenticação: controla o acesso ao sistema.
- Serviço para gerência de funcionários: controla as operações realizadas sobre os funcionários da empresa.
- Serviço para gerência das linhas e paradas: controla as operações realizadas sobre as linhas ou pontos de ônibus do sistema.
- Serviço para gerência de veículos: controla as operações realizadas sobre os veículos da empresa.
- Serviço de atualizações: controla as atualizações das posições dos veículos e a realização das requisições de táxi.

Todos os serviços que foram utilizados nas aplicações Android possuem suporte a acesso seguindo o padrão REST e serialização JSON.

4.4.1 Sistema de Atualizações

Com o intuito de dar suporte ao recebimento das atualizações das posições dos veículos e ao sistema de requisições de táxis, foram implementadas algumas funcionalidades para realizar estas atualizações.

4.4.1.1 Recebimento das posições dos veículos

Tanto na aplicação do passageiro quanto na aplicação da empresa prestadora de serviços de transporte, é necessário a exibição da posição dos veículos sempre atualizada nos mapas. Para isso, o servidor do sistema foi projetado para ser informado pelos usuários quais são os veículos que eles desejam receber atualizações. A partir disso, o servidor começa a guardar em um *buffer* as

atualizações que devem ser enviadas para cada usuário do sistema. Quando um usuário solicita as atualizações dos veículos, o servidor retorna todas as atualizações pertencentes a este usuário que estão no *buffer*. Esta solicitação de atualizações é realizada através de *polling*.

4.4.1.2 Requisição de Táxis

Ao desenvolver as funcionalidades relacionadas à requisição de táxis, julgamos necessário desenvolver um sistema de notificação instantânea. Abaixo apresentamos o fluxo básico do funcionamento de requisições de táxi:

1. Usuário faz uma requisição por um táxi (podendo ser para um táxi específico ou não).
2. Taxista recebe a requisição e aceita ou rejeita a chamada.
3. Usuário recebe a resposta do taxista.

Para que o fluxo acima funcione e não ocorra uma espera excessiva pelo recebimento de requisições por parte do taxista, ou uma espera excessiva pela resposta da requisição por parte do cliente, foram feitas as seguintes implementações:

- Implementação na aplicação do taxista: o taxista fica em contínuo *long polling* procurando por requisições. As condições para que o servidor retorne as requisições ao taxista são: existência de uma requisição de táxi específica a ele ou quando é identificado um tempo de espera de dez segundos por uma resposta do servidor
- Implementação na aplicação do passageiro: ao realizar uma requisição de táxi, independente de ser para um táxi específico ou não, é aberto um *long polling* que possui condição de retorno imediato caso receba uma resposta positiva de algum taxista. Caso não haja nenhuma resposta no período de sessenta segundos, é informado ao usuário que nenhum taxista respondeu ao seu chamado.

4.5 DETALHAMENTO DOS PRINCIPAIS ALGORITMOS UTILIZADOS NO SISTEMA

Nesta seção apresentamos alguns algoritmos que possuem grande relevância para o correto funcionamento das funcionalidades do sistema.

4.5.1 Sistema de Requisição de Táxis

O sistema de requisição de táxis (descrito na seção 4.4.1.2) foi projetado para fornecer atualizações imediatas, ou seja, para notificar o usuário imediatamente que ocorra algum evento no servidor. Seguem exemplos de notificações que são imediatas no sistema:

- Quando um passageiro realiza uma requisição de táxi específica a um taxista, o taxista é imediatamente alertado sobre esta notificação.
- Quando um taxista aceita uma requisição feita por um passageiro, o passageiro é imediatamente notificado. No caso de requisições específicas, caso um taxista receba uma chamada específica, independente se ele aceitar ou recusar a requisição, o cliente recebe imediatamente a resposta.

Para realizar a implementação desse sistema de requisição, foi criada uma classe que recebe chamadas de um serviço e que é instanciada uma única vez no sistema (padrão *Singleton*), para que todos que utilizarem esta classe usufruam das mesmas informações. Esta classe armazena informações das requisições dos taxistas (busca por passageiros) em aberto e dos pedidos de táxi dos passageiros (requisições de táxis), bem como toda a lógica necessária para esse sistema de requisições. As implementações dos sistemas do taxista e do passageiro, para que tenham caráter de atualização imediato, foram baseadas nos conceitos de *long polling* (seção 2.4.2), onde uma requisição é aberta e, até que haja uma atualização, essa requisição não é respondida.

Para implementar a técnica de *long polling*, no caso das requisições de táxi, utilizamos, na classe descrita acima, um método de espera (*WaitOne(time)*) para que a thread aberta pelo serviço requisitor fique dormindo até que o tempo passado por parâmetro seja atingido ou até que alguém sinalize que esta thread deve acordar (método *set()*). Com isso foi possível realizar as seguintes funcionalidades:

- O taxista fica com requisição aberta por até dez segundos (“*WaitOne*(*10*)”) esperando alguma requisição de táxi específica. Se algum passageiro requisitá-lo especificamente antes do fim desses dez segundos, a thread é liberada através do método “*Set*()”. Após a liberação desta thread, independente da maneira que foi liberada, o sistema retorna ao taxista todas as requisições de táxi que se aplicam a ele.
- O passageiro realiza uma requisição por táxi e fica com essa requisição aberta por até sessenta segundos (“*WaitOne*(*60*)”), se essa requisição for específica e o taxista responder, ou se a requisição for global e algum taxista responder positivamente, é acionado o método “*Set*()” e essa resposta é retornada ao passageiro imediatamente. Se nenhum taxista responder a requisição, ao fim dos sessenta segundos o sistema retornará que nenhum taxista respondeu ao chamado.

4.5.2 Sistema de Atualização das Posições dos Veículos

Para realizar a implementação do sistema de atualização das posições dos veículos (seção 4.4.1.1), primeiramente havia sido pensando em desenvolver um sistema baseado em streaming (seção 2.4.3) que seria aplicado tanto na aplicação web como na camada das aplicações Android, porém, pelas dificuldades encontradas em sua implementação, foi optado por uma implementação mais simples.

Para que houvesse suporte ao sistema de atualizações, a camada do servidor recebeu a implementação de serviços que recebem as atualizações dos veículos e guardam as mesmas em uma cache, além disso, fornece métodos que devolvem os valores contidos na cache. Para o consumo das atualizações foram utilizadas duas implementações, uma para a aplicação web, e uma para a aplicação Android.

Para a implementação das atualizações na aplicação web, foi criado um módulo chamado de Comet Web, que é responsável pela gerência de atualizações do posicionamento dos veículos no mapa da aplicação web. Para que esse módulo se mantenha atualizado, é feita uma requisição a cada 250 milissegundos

perguntando por novas atualizações contidas servidor. A comunicação entre usuário (mapa) e o módulo do *Comet* Web é feita através da implementação de um sistema de long polling que possui um sistema de assinatura, ou seja, o usuário indica quais são os veículos que está assinando atualizações e somente essas são repassadas a ele. O sistema de long polling implementado no *Comet* Web foi baseado no princípio de que a exibição das atualizações não necessita ser imediata, por isso, o sistema acumula atualizações por no mínimo dois segundos do momento em que foi aberta a requisição por novas atualizações, evitando assim um overhead causado pela chamada de muitas requisições por segundo. Outro fator importante na implementação desse módulo é que, mesmo que uma requisição tenha sido respondida, as atualizações para o usuário continuam sendo acumuladas para que, caso seja aberta uma nova requisição, o usuário receba todas as atualizações que ocorreram nesse período.

Para a implementação no módulo Android, foi realizado um polling simples que, de tempos em tempos, requisita ao servidor quais são as posições dos veículos que devem ser exibidas no mapa da aplicação

5 MODELAGEM DE DADOS

Neste capítulo, apresentamos o modelo lógico de dados para o sistema e seu respectivo dicionário de dados.

5.1 MODELO LÓGICO

De acordo com os casos de uso que foram estabelecidos, na figura 35 apresentamos o modelo lógico utilizado como solução para o sistema desenvolvido.

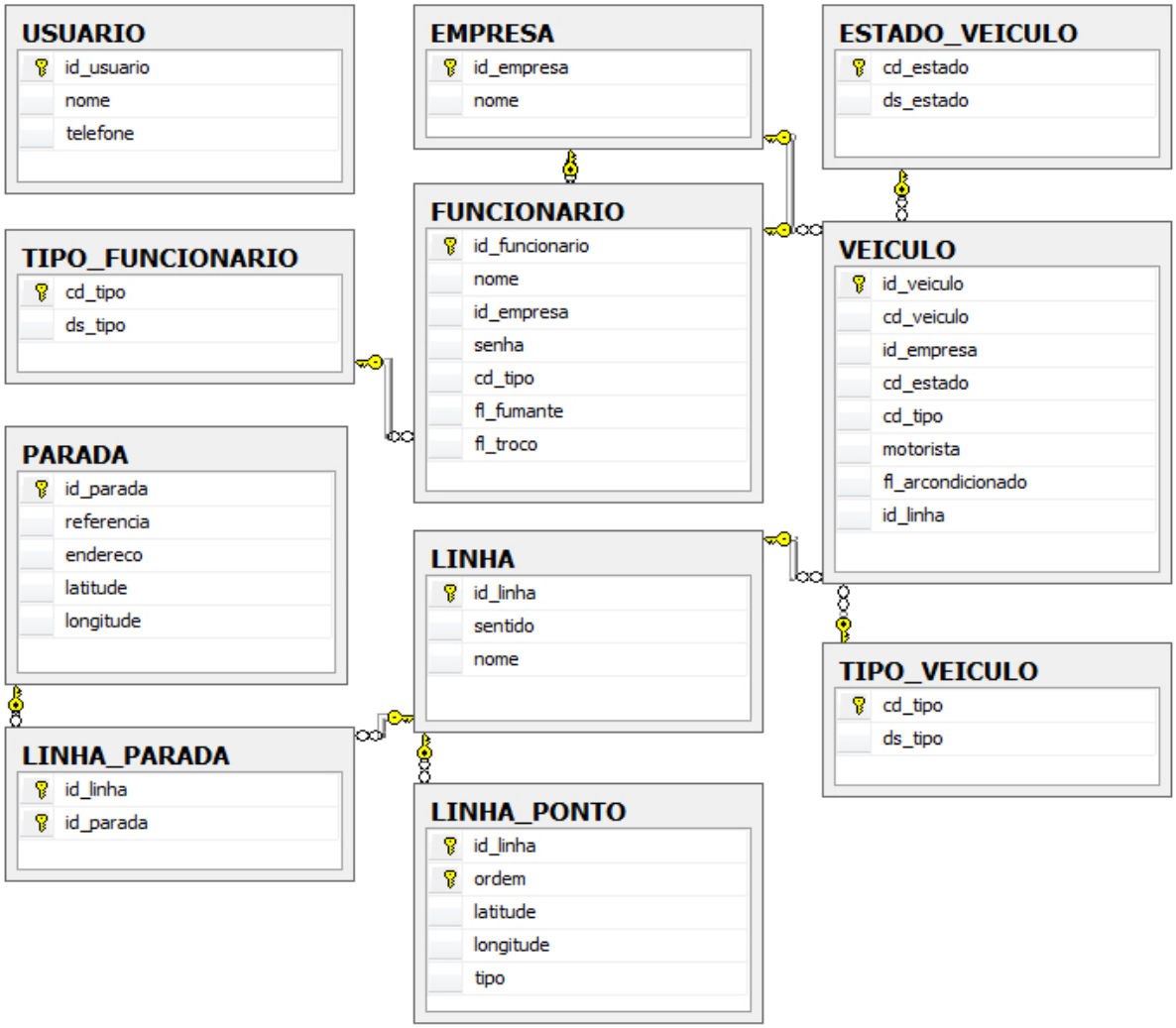

Figura 35: Modelo Lógico de Dados

A partir deste modelo de dados, na figura 36 apresentamos o modelo de representação que foi gerado para o sistema utilizando o *entity framework*.

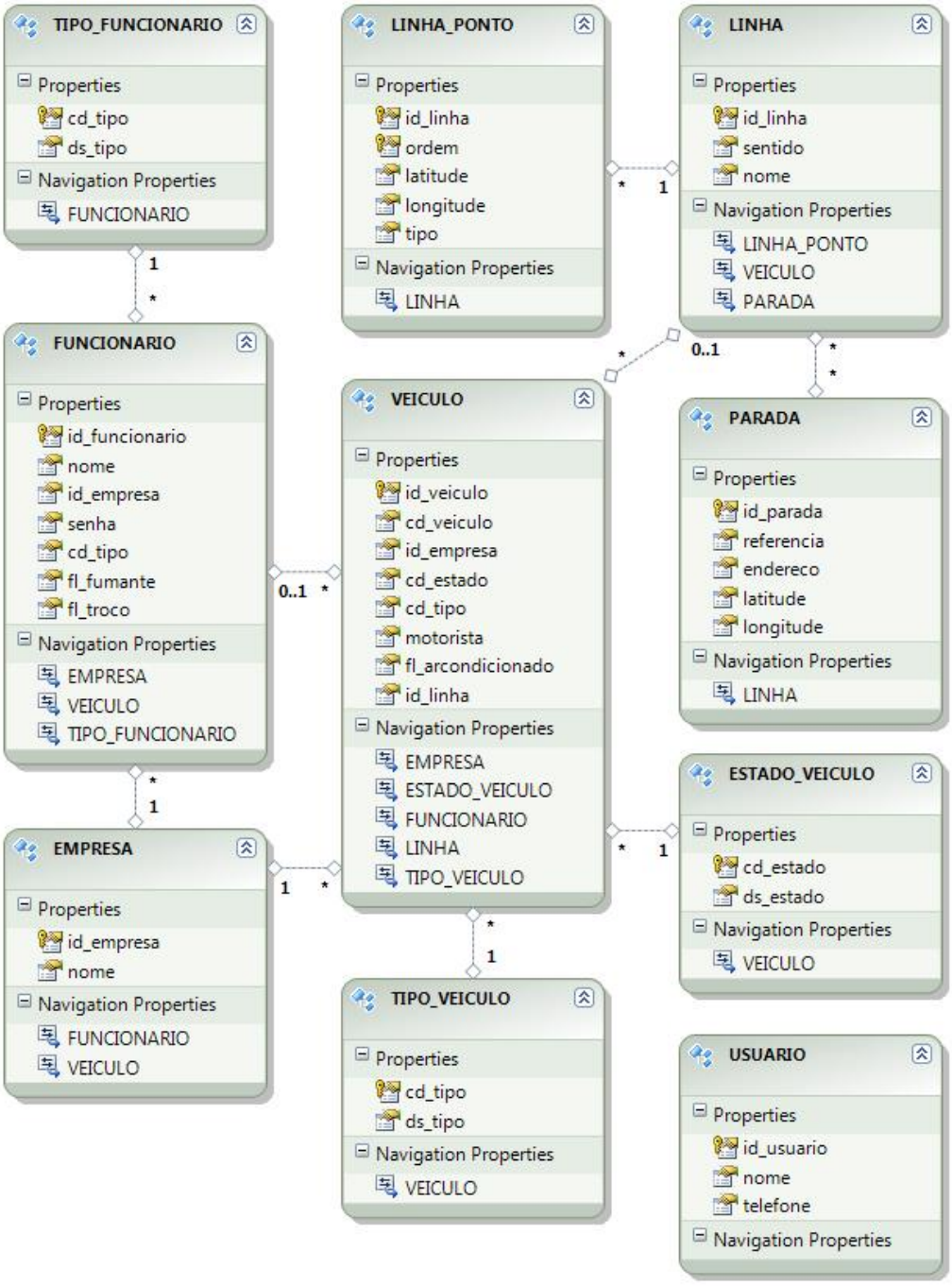

Figura 36: Modelo de Representação dos Dados

5.2 DICIONÁRIO DE DADOS

Nesta seção, apresentaremos a descrição das tabelas do sistema e seus atributos.

5.2.1 Tabela Usuário

Armazena as informações dos usuários do sistema.

Tabela 1: USUARIO

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
id_usuario	Chave primária	Não nulo
nome	Nome do usuário	Nulo
telefone	Telefone do usuário	Nulo

5.2.2 Tabela Veículo

Armazena as informações dos veículos do sistema.

Tabela 2: VEICULO

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
id_veiculo	Chave primária	Não nulo
cd_veiculo	Código do veículo	Não nulo
id_empresa	Identificador da empresa do veículo	Não nulo
cd_estado	Estado do veículo	Não nulo
cd_tipo	Tipo do veículo	Não nulo
motorista	Motorista Atual do veículo	Nulo
fl_arcondicionado	Indicador de ar condicionado	Não Nulo
id_linha	Identificador da linha (se veículo for ônibus ou lotação)	Nulo

5.2.3 Tabela Estado do Veículo

Armazena os estados que os veículos (ônibus, táxi e lotação) podem possuir, tais como: disponível, ocupado e invisível para a população.

Tabela 3: ESTADO_VEICULO

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
cd_estado	Chave primária	Não nulo
ds_estado	Descrição do estado	Não nulo

5.2.4 Tabela Tipo do Veículo

Armazena os tipos de veículos do sistema, ou seja: ônibus, táxi e lotação.

Tabela 4: TIPO_VEICULO

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
cd_tipo	Chave primária	Não nulo
ds_tipo	Descrição do tipo	Não nulo

5.2.5 Tabela Empresa

Armazena as empresas do sistema.

Tabela 5: EMPRESA

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
id_empresa	Chave primária	Não nulo
nome	Nome da empresa	Não nulo

5.2.6 Tabela Funcionário

Armazena as informações dos funcionários do sistema.

Tabela 6: FUNCIONARIO

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
id_funcionario	Chave primária	Não nulo
nome	Nome do funcionário	Não nulo
id_empresa	Identificador da empresa do funcionário	Não nulo
senha	Senha do funcionário	Não nulo
cd_tipo	Tipo do funcionário	Não nulo
fl_fumante	Indicador se aceita fumantes (somente para taxistas)	Não Nulo
fl_troco	Indicador se possui troco para mais de R$50,00 (somente para taxistas)	Não Nulo

5.2.7 Tabela Tipo do Funcionário

Armazena os tipos de funcionários do sistema, ou seja: administrador, motorista e atendente.

Tabela 7: TIPO_FUNCIONARIO

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
cd_tipo	Chave primária	Não nulo
ds_tipo	Descrição do tipo	Não nulo

5.2.8 Tabela Linha

Armazena as informações das linhas de ônibus e lotação do sistema.

Tabela 8: LINHA

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
id_linha	Chave primária	Não nulo
nome	Nome da linha	Não nulo
sentido	Sentido da linha	Não nulo

5.2.9 Tabela Parada

Armazena informações das paradas de ônibus.

Tabela 9: PARADA

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
id_parada	Chave primária	Não nulo
referencia	Ponto de referência da parada	Nulo
endereco	Endereço da parada	Nulo
latitude	Latitude da parada	Não nulo
longitude	Longitude da parada	Não nulo

5.2.10 Tabela Linha Parada

Representa a relação de linhas de ônibus com paradas.

Tabela 10: LINHA_PARADA

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
id_linha	Chave primária	Não nulo
id_parada	Chave primária	Não nulo

5.2.11 Tabela Linha Ponto

Armazena todos os pontos que uma determinada linha de ônibus ou lotação percorre.

Tabela 11: LINHA_PONTO

COLUNA	DESCRIÇÃO	OBSERVAÇÃO
id_linha	Chave primária	Não nulo
ordem	Chave primária (indica a ordem do percurso de uma linha)	Não nulo
tipo	Tipo do ponto (início, normal, fim)	Não nulo
latitude	Latitude do ponto	Não nulo
longitude	Longitude do ponto	Não nulo

6 TRABALHOS RELACIONADOS

Nesta seção apresentamos trabalhos que estão, de alguma forma, relacionados ao trabalho que desenvolvemos.

6.1 CAB4ME – THE MOBILE CAB FINDER

O cab4me[5] é um sistema de requisições de táxis que dá ao cliente a visualização do mapa para que ele selecione o local onde deseja um táxi. Ele possibilita ao cliente realizar as seguintes ações:

- Requisitar um táxi em um determinado local do mapa.
- Visualizar pontos de táxi disponíveis na sua área.
- Obter informações adicionais (tipos de carros disponíveis, métodos de pagamento, etc.) sobre cooperativas que estão cadastradas no sistema.

A figura 37 apresenta a interface do aplicativo para um cliente portador de um smartphone.

Figura 37: Interface do aplicativo cab4me

Fonte: [8]

6.1.1 Funcionamento do Sistema

As cooperativas de táxis que desejam se registrar no sistema devem possuir uma conta do Google [9] para que seja feita uma autenticação do usuário. Ao cadastrarem-se no sistema, as cooperativas devem criar seu perfil e informar a sua área de serviço. Antes de disponibilizar a cooperativa no sistema, é realizada uma pesquisa para verificar a autenticidade da empresa, sendo assim, impedindo que empresas mal intencionadas se cadastrem na aplicação. O objetivo do cab4me é construir uma rede de requisições de táxis automática, onde o cliente não precisa telefonar para cooperativas de táxis para requisitar o serviço. O aplicativo está disponível para *iPhone* e para smartphones que utilizam o sistema operacional Android [8].

6.1.2 Relacionamento com o Sistema Desenvolvido

Em relação ao módulo de requisições de táxis do nosso sistema, além de possibilitar que o cliente realize as operações que o cab4me também oferece, estamos permitindo que o cliente visualize no mapa todos os táxis que desejam ser visualizados, sendo assim, possibilitando que ele requisite um táxi específico e que acompanhe o seu deslocamento no mapa.

6.2 GOOGLE TRANSIT

O Google Maps [10] é um serviço gratuito de visualização de mapas e imagens de satélite desenvolvido pelo Google. Já o Google Transit [11] é um serviço que utiliza o Google Maps para, por exemplo, realizar as seguintes ações:

- Obter rotas de transporte público passo a passo em um smartphone.
- Localizar paradas de transporte público.
- Visualizar informações e horários de veículos coletivos.

A figura 38 apresenta a interface do sistema durante a locomoção de um usuário. Como veremos a seguir, a figura nos mostra o sistema informando ao usuário em quanto tempo ele deve trocar de transporte para conseguir chegar ao seu destino.

Figura 38: Interface do Google Transit

Fonte: [12]

6.2.1 Funcionamento do Sistema

Entre outras funcionalidades, ele permite ao usuário visualizar quais coletivos, metrô ou trens ele deve pegar para chegar ao seu destino. Além disso, informa qual o momento certo para o usuário descer do veículo. Estas funcionalidades possuem diversos problemas no Brasil, pois a cobertura de transportes públicos do serviço do Google é limitada a poucas áreas. O sistema integra paradas de trânsito, rotas, horários e informações de tarifas para tentar realizar um planejamento de um deslocamento, utilizando transportes públicos, de uma forma rápida e fácil. A funcionalidade Transit está disponível através do Google Maps para smartphones dos mais diversos tipos, como Android, iPhone, Windows Phone e BlackBerry.

6.2.2 Relacionamento com o Sistema Desenvolvido

O Google Transit permite ao usuário, por exemplo, visualizar a rota de uma linha de transporte público. Isto possui relação com o módulo de visualização dos veículos de transporte do nosso trabalho.

6.3 POABUS

O poabus [13] é um projeto que fornece informações das linhas de ônibus da cidade de Porto Alegre, tais como o trajeto e onde estão localizadas as paradas, através de uma aplicação web gratuita. Ele foi desenvolvido por um estudante da Universidade Federal do Rio Grande do Sul, Bruno Jurkovski.

A figura 39 apresenta a interface do poabus.

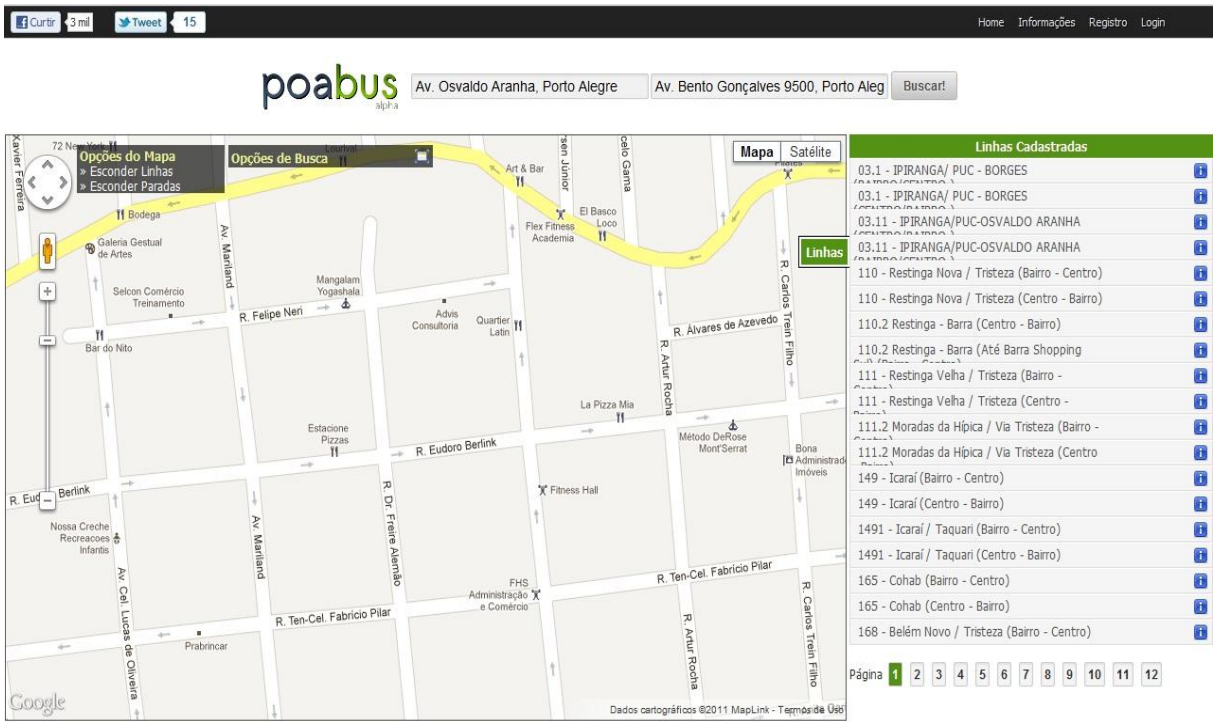

Figura 39: Interface web do poabus

Fonte: [13]

6.3.1 Funcionamento do Sistema

O poabus foi desenvolvido colaborativamente, ou seja, os próprios usuários podem cadastrar as linhas e paradas. Para se tornar um usuário, basta realizar login no sistema utilizando uma conta do Google. Atualmente, é possível buscar linhas de duas maneiras:

- Indicando qual é o endereço de origem e endereço de destino do trajeto, sendo assim, o sistema lhe mostrará quais linhas você pode pegar para chegar ao seu destino.
- Colocando marcadores em algum ponto do mapa, sendo assim, o sistema indicará quais linhas passam pelos pontos marcados.

6.3.2 Relacionamento com o Sistema Desenvolvido

O poabus tem grande relacionamento com o módulo de transporte coletivo do nosso sistema, pois ele apresenta diversas funcionalidades que estão no escopo do sistema que desenvolvemos. Além das funções oferecidas pelo poabus, nosso sistema também permite que o usuário veja a localização atual dos veículos.

6.4 TAXIMETRO PRO

O Taximetro Pro [14] é um aplicativo desenvolvido para o sistema operacional Android que tem como objetivo calcular, aproximadamente, a tarifa de um trajeto percorrido por um táxi. Este aplicativo foi desenvolvido por um estudante da Pontifícia Universidade Católica do Rio Grande do Sul, Irineu Licks Filho.

Na figura 40, apresentamos a interface do aplicativo.

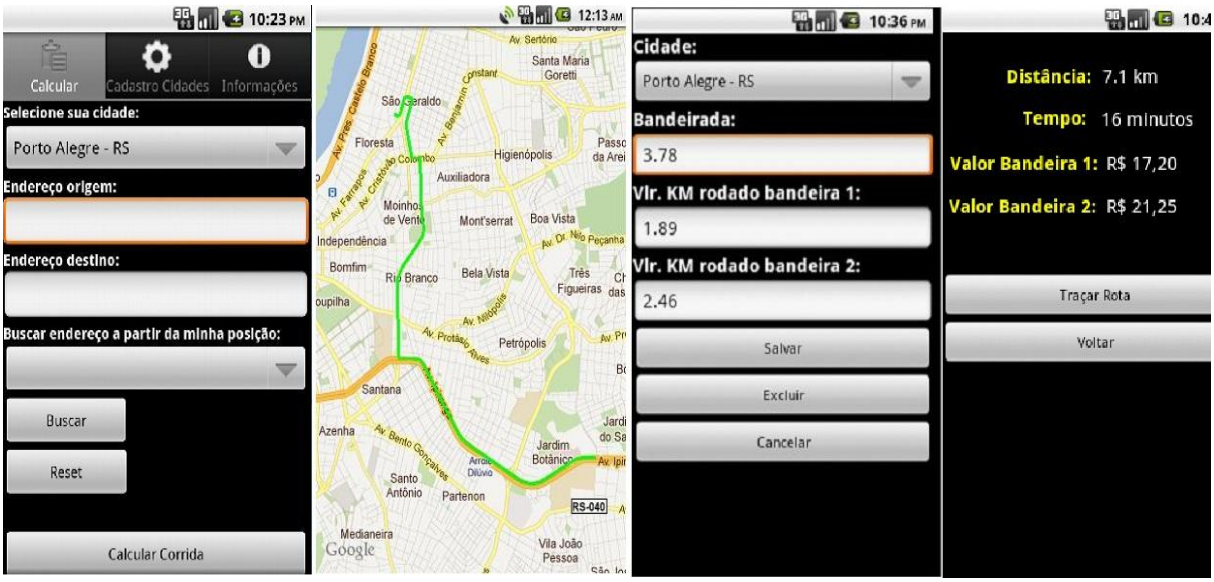

Figura 40: Interface do aplicativo Taximetro Pro

Fonte: [14]

6.4.1 Funcionamento do Sistema

Taximetro Pro é um aplicativo extremamente configurável, pois ele possibilita que o usuário cadastre sua cidade e permite cadastrar/alterar as tarifas de táxi das cidades já cadastradas no sistema. Estando a cidade do usuário cadastrada, ele proporciona ao usuário as seguintes funcionalidades:

- Calcular a tarifa de um determinado trajeto a ser percorrido de táxi.
- Calcular o tempo estimado de um determinado trajeto a ser percorrido de táxi.
- Calcular a distância de um determinado trajeto a ser percorrido de táxi.
- Visualizar, utilizando o Google Maps, o trajeto que a aplicação estipulou para ser percorrido.

Ele permite ao usuário duas maneiras de calcular um trajeto:

- Informar o endereço de origem e de destino.
- Capturar a posição atual do usuário (através do GPS) e usá-la como endereço de origem ou de destino

6.4.2 Relacionamento com o Sistema Desenvolvido

Este aplicativo se insere no contexto das operações com táxis do nosso sistema, mais especificamente nas operações de capturar a posição do usuário. O desenvolvedor desta aplicação contribuiu para melhorar o algoritmo de captura da posição do usuário do nosso sistema.

7 CONCLUSÃO

Neste capítulo, apresentaremos as conclusões obtidas ao longo do desenvolvimento do trabalho.

Este trabalho de conclusão foi caracterizado pela abordagem de muitos dos conceitos estudados no curso de Ciência da Computação, os quais ressaltamos:

- Gerência de prazo e distribuição de tarefas entre os integrantes do projeto.
- Consolidação dos conhecimentos estudados nas disciplinas de bancos de dados e programação.
- Experiência no desenvolvimento de um sistema por completo, passando por todas as fases de um projeto.

Assimilamos que riscos existem e sempre devem ser levados em consideração e que, ao longo do desenvolvimento do trabalho, o escopo inicial pode sofrer diversas alterações. Acreditamos que o trabalho foi um sucesso não tanto pela implementação que foi realizada, mas sim pelo fato de que tivemos uma experiência extremamente satisfatória que nos engrandece como cientistas da computação e como profissionais da informática.

Entre a proposta de trabalho de conclusão e a implementação propriamente dita do sistema, aconteceram diversas mudanças com relação àquilo que era desejado em um primeiro momento. Um exemplo disso foi a alteração na maneira que foram desenvolvidos os serviços (web services) e a aplicação web do sistema. Quando a proposta para o trabalho de conclusão foi entregue, pensamos em construir o sistema em outra linguagem de programação, utilizando outras tecnologias do que as que foram apresentadas neste projeto. A ideia inicial era desenvolver utilizando a linguagem Java, porém, com a realização de alguns exemplos de aplicações e devido a nossa experiência profissional, foi notado que o desenvolvimento seria mais rápido e não perderia em qualidade se feito em tecnologias .NET. Neste momento foi feita a difícil escolha de abdicar daquilo que já estava definido para começar novos estudos sobre outra tecnologia. Ao final do trabalho, essa mudança se mostrou correta, pois apesar de algumas alterações de escopo, conseguimos ter uma versão estável do sistema que corresponde ao objetivo daquilo que foi proposto no início.

7.1 TRABALHOS FUTUROS

Tendo em vista que um software nunca está pronto e sempre está em constante evolução, existem tópicos que serão estudados a fim de agregar ao sistema que está sendo entregue. Seguem abaixo:

- Desenvolvimento de um sistema de cálculo de trajetos: Pretendemos fornecer ao usuário uma funcionalidade de exibir quais linhas ele deve pegar para chegar a um determinado destino, e também realizar o cálculo de quanto o usuário gastaria para chegar ao destino se utilizasse táxi para isso.
- Estudar o “Android Cloud to Device Messaging Framework” [24]: Framework para Android que permite o recebimento de atualizações assíncronas provenientes de um servidor.
- Chamada de táxi via aplicação web: permitir aos funcionários das empresas de transportes realizarem notificações de passageiros a taxistas através do módulo web, possibilitando um futuro passageiro ligar para a empresa pedindo um táxi, e o taxista ser notificado via aplicação.

REFERÊNCIAS

[1] MATEUS, Geraldo R.; LOUREIRO, Antônio A. F.. **Introdução a Computação Móvel**. Rio de Janeiro, RJ: DCC/IM, COPPE/Sistemas, NCE/UFRJ, 1998.

[2] LECHETA, Ricardo R.. **Google Android: Aprenda a criar aplicações para dispositivos móveis com o Android SDK.** 2. ed. São Paulo: Novatec Editora, 2010.

[3] KAPLAN, Elliott D. **Understanding GPS: principles and applications**. Norwood, MA: Artech House, 1996.

[4] GRAHAM, Steve; SIMEONOV, Simeon; BOUBEZ, Toufic; DAVIS, Doug; DANIELS, Glen; NAKAMURA, Yuichi; NEYAMA, Ryo. **Building web services with Java: making sense of XML, SOAP, WSDL and UDDI**. Indianapolis, IN: Sams, 2002.

[5] CHAPPELL, David A.; JEWELL, Tyler. **Java web services**. Sebastopol, CA: O'Reilly, 2002.

[6] IBM, **Web services for J2EE**. Acessado em 04 de setembro de 2011 em: http://huihoo.org/openweb/web_services_for_j2ee/index_eng.shtml.htm

[7] Android Developers - Activities. Acessado em 11 de novembro de 2011 em: http://developer.android.com/guide/topics/fundamentals/activities.html

[8] Cab4me. Acessado em 04 de setembro de 2011 em: http://www.cab4me.com

[9] Google Accounts. Acessado em 04 de setembro de 2011 em: https://accounts.google.com/Login

[10] Google Maps. Acessado em 10 de setembro de 2011 em: http://maps.google.com.br

[11] Google Transit. Acessado em 11 de novembro de 2011 em: http://www.google.com/transit

[12] RECKZIEGEL, Maurício. Artigo sobre o Google Transit. Acessado em 11 de novembro de 2011 em: http://imasters.com.br/artigo/12436/tendencias/minha-cidade-no-google-transit

[13] Poabus. Acessado em 11 de novembro de 2011 em: http://www.poabus.com.br

[14] Taximetro Pro. Acessado em 11 de novembro de 2011 em: https://market.android.com/details?id=br.com.main

[15] Microsoft, **What Is Windows Communication Foundation?**. Acessado em 15 de junho de 2012 em: http://msdn.microsoft.com/en-us/library/ms731082(v=vs.90).aspx

[16] IBM, **RESTful Web Services: The basics**. Acessado em 15 de junho de 2012 em: https://www.ibm.com/developerworks/webservices/library/ws-restful/

[17] GODINHO, Rafael. Artigo: Criando Serviços REST com WCF. Acessado em 15 de junho de 2012 em: http://msdn.microsoft.com/pt-br/library/dd941696.aspx

[18] JSON. Acessado em 15 de junho de 2012 em: http://www.json.org/

[19] Microsoft, **The ADO.NET Entity Framework Overview**. Acessado em 15 de junho de 2012 em: http://msdn.microsoft.com/en-us/library/aa697427(v=vs.80).aspx

[20] Microsoft, **MVC**. Acessado em 15 de junho de 2012 em: http://msdn.microsoft.com/en-us/library/ff649643.aspx

[21] Microsoft, **ASP.NET MVC 3**. Acessado em 15 de junho de 2012 em: http://www.asp.net/mvc/mvc3

[22] Open Handset Alliance. Acessado em 22/06/2012 em: http://www.openhandsetalliance.com/

[23] Dalvik Virtual Machine. Acessado em 22/06/2012 em: http://www.dalvikvm.com/

[24] Android Cloud to Device Messaging Framework. Acessado em 22/06/2012 em: https://developers.google.com/android/c2dm/]ndroid/c2dm

APÊNDICE A – Detalhamento de Casos de Uso da Aplicação do Passageiro.

1. Caso de Uso: Acompanhar Deslocamento de Táxi

1.1. Breve Descrição

O usuário deve ser capaz de acompanhar o deslocamento de um táxi que tenha atendido a uma requisição sua.

1.2. Breve Descrição dos Atores

1.2.1. Usuário

Cliente portador de um dispositivo móvel com sistema operacional Android.

1.3. Pré-condições

Aplicativo EASYT deve estar sendo executado e deve-se ter um táxi em deslocamento para atender o cliente.

1.4. Fluxo Básico de Eventos

1. O caso de uso inicia quando o usuário faz uma requisição de um táxi e a requisição é aceita pelo taxista.
2. O sistema informa ao usuário que sua requisição foi aceita e mostra a localização do taxista no mapa.
3. O sistema passa a acompanhar o deslocamento do taxista no mapa da aplicação até que ele chegue ao destino do cliente.

1.5. Pós-condições

1.5.1. Sucesso em acompanhar o deslocamento de um táxi

Usuário acompanhou o deslocamento do táxi.

2. Caso de Uso: Procurar Táxi

2.1. Breve Descrição

O usuário deve ser capaz de procurar um táxi no mapa da aplicação e requisitá-lo.

2.2. Breve Descrição dos Atores

2.2.1. Usuário

Cliente portador de um dispositivo móvel com sistema operacional Android.

2.3. Pré-condições

Aplicativo EASYT deve estar sendo executado.

2.4. Fluxo Básico de Eventos

1. O usuário clica em “Ver Mapa” na tela inicial do sistema.
2. O usuário clica em “Táxi” para visualizar somente táxis no mapa.

3. O usuário visualiza o mapa da aplicação e procura um táxi que esteja disponível.
4. O usuário encontra um táxi com status disponível e clica nele.
5. O sistema apresenta as informações do taxista.
6. O usuário clica em “Chamar Táxi”.
7. O sistema notifica o usuário que seu chamado foi aceito e inicia o caso de uso “Acompanhar Deslocamento de Táxi”.

2.5. Fluxo Alternativo de Eventos

2.5.1. Se o usuário não encontrar nenhum táxi disponível

1. Se desejar, o usuário pode iniciar o caso de uso “Requisitar Táxi” ou fechar a aplicação.

2.5.2. Se o taxista não responder a chamada do passageiro ou não aceitar

1. O sistema notifica o usuário que sua requisição foi cancelada.

2.6. Pós-condições

2.6.1. Sucesso ao procurar um táxi

Usuário conseguiu encontrar um táxi.

3. Caso de Uso: Requisitar Táxi

3.1. Breve Descrição

O usuário deve ser capaz de requisitar um táxi.

3.2. Breve Descrição dos Atores

3.2.1. Usuário

Cliente portador de um dispositivo móvel com sistema operacional Android.

3.3. Pré-condições

Aplicativo EASYT deve estar sendo executado.

3.4. Fluxo Básico de Eventos

1. O usuário clica em “Requisitar Táxi” na tela inicial do sistema.
2. O sistema mostra ao usuário o endereço obtido através do GPS e pede confirmação.
3. O usuário confirma o endereço.
4. O usuário preenche o campo “Observação” para fornecer mais informações ao taxista.
5. O usuário clica em “Requisitar”.
6. O sistema mostra aos taxistas disponíveis a localização do usuário que deseja um táxi.

7. O sistema notifica o usuário que seu chamado foi aceito e inicia o caso de uso “Acompanhar Deslocamento de Táxi”.

3.5. Fluxo Alternativo de Eventos

3.5.1. Se o usuário clicar em alterar endereço

1. O usuário corrige o endereço capturado pelo GPS e o caso de uso continua no passo 4.

3.5.2. Se nenhum taxista responder a chamada do cliente

1. O sistema notifica o usuário que sua requisição foi cancelada.

3.6. Pós-condições

3.6.1. Sucesso ao requisitar um táxi

Usuário conseguiu requisitar um táxi.

4. Caso de Uso: Editar Informações Pessoais

4.1. Breve Descrição

O usuário deve ser capaz de editar suas informações pessoais no sistema.

4.2. Breve Descrição dos Atores

4.2.1. Usuário

Cliente portador de um dispositivo móvel com sistema operacional Android.

4.3. Pré-condições

Aplicativo EASYT deve estar sendo executado.

4.4. Fluxo Básico de Eventos

1. O usuário clica em “Perfil” na tela inicial do sistema.
2. O usuário preenche os campos (nome e telefone) e clica em “Ok”.
3. O sistema armazena as informações pessoais do usuário.

4.5. Pós-condições

4.5.1. Sucesso ao editar informações pessoais

Usuário conseguiu editar suas informações pessoais.

5. Caso de Uso: Acompanhar Deslocamento de Ônibus e Lotações

5.1. Breve Descrição

O usuário deve ser capaz de acompanhar o deslocamento de um ônibus ou de uma lotação que esteja no seu campo de visão do mapa.

5.2. Breve Descrição dos Atores

5.2.1. Usuário

Cliente portador de um dispositivo móvel com sistema operacional Android.

5.3. Pré-condições

Aplicativo EASYT deve estar sendo executado e deve-se ter algum ônibus ou lotação em circulação.

5.4. Fluxo Básico de Eventos

1. O usuário clica em “Ver Mapa” na tela inicial da aplicação.
2. O usuário clica em “Ônibus/Lotação” para visualizar somente ônibus ou lotações.
3. O usuário seleciona um ônibus ou lotação e clica em “Acompanhar”.
4. O sistema passa a acompanhar o deslocamento do veículo no mapa da aplicação.

5.5. Pós-condições

5.5.1. Sucesso em acompanhar o deslocamento de um ônibus ou lotação

Usuário acompanhou o deslocamento do veículo.

6. Caso de Uso: Visualizar Mapa

6.1. Breve Descrição

O usuário deve ser capaz de visualizar o mapa da aplicação.

6.2. Breve Descrição dos Atores

6.2.1. Usuário

Cliente portador de um dispositivo móvel com sistema operacional Android.

6.3. Pré-condições

Aplicativo EASYT deve estar sendo executado.

6.4. Fluxo Básico de Eventos

1. O usuário clica em “Ver Mapa” na tela inicial da aplicação.
2. O usuário seleciona “Táxi” ou “Ônibus/Lotação” conforme for o seu desejo.
3. O sistema apresenta o mapa para o usuário com o tipo de veículo que ele escolheu.

6.5. Pós-condições

6.5.1. Sucesso ao visualizar o mapa

Usuário conseguiu visualizar o mapa.

APÊNDICE B – Detalhamento de Casos de Uso da Aplicação do Prestador de Serviços de Transporte.

1. Caso de Uso: Login

1.1. Breve Descrição

O prestador de serviço de transporte deve ser capaz de acessar a aplicação.

1.2. Breve Descrição dos Atores

1.2.1. Usuário

Prestador de serviço de transporte portador de um dispositivo móvel com sistema operacional Android.

1.3. Pré-condições

Aplicativo “EASYT Driver” deve estar sendo executado.

1.4. Fluxo Básico de Eventos

1. O sistema apresenta uma tela com os campos “Nome”, “Senha” e “Veículo”.
2. O usuário preenche os campos e clica em “Entrar”.
3. O sistema apresenta a tela inicial da aplicação.

1.5. Fluxo Alternativo de Eventos

1.5.1. Se o usuário não preencher os campos da tela inicial

1. O sistema informará que não foi possível acessar o sistema.

1.6. Pós-condições

1.6.1. Sucesso ao acessar o sistema

Usuário conseguiu acessar o sistema.

2. Caso de Uso: Alterar Status

2.1. Breve Descrição

O prestador de serviço de transporte deve ser capaz de alterar seu status no sistema.

2.2. Breve Descrição dos Atores

2.2.1. Usuário

Prestador de serviço de transporte portador de um dispositivo móvel com sistema operacional Android.

2.3. Pré-condições

Aplicativo “EASYT Driver” deve estar sendo executado e o usuário já deve ter efetuado o Login no sistema.

2.4. Fluxo Básico de Eventos

1. O usuário clica em “Status”.
2. O sistema apresenta uma lista de status para o usuário (tais como disponível, ocupado, invisível para população).
3. O usuário seleciona o status desejado.

2.5. Pós-condições

2.5.1. Sucesso ao alterar o status

Usuário conseguiu alterar seu status.

3. Caso de Uso: Editar Informações do Taxista

3.1. Breve Descrição

O taxista deve ser capaz de editar suas informações no sistema.

3.2. Breve Descrição dos Atores

3.2.1. Usuário

Taxista portador de um dispositivo móvel com sistema operacional Android.

3.3. Pré-condições

Aplicativo “EASYT Driver” deve estar sendo executado e o taxista já deve ter efetuado o Login no sistema.

3.4. Fluxo Básico de Eventos

1. O usuário clica em “Perfil”.
2. O usuário marca as opções que ele desejar (tais como: aceitar fumantes, possuir ar condicionado).
3. O sistema salva as informações do usuário.

3.5. Pós-condições

3.5.1. Sucesso ao editar informações

Usuário conseguiu editar suas informações.

4. Caso de Uso: Encontrar Passageiros

4.1. Breve Descrição

O taxista deve ser capaz de encontrar passageiros no mapa da aplicação.

4.2. Breve Descrição dos Atores

4.2.1. Usuário

Taxista portador de um dispositivo móvel com sistema operacional Android.

4.3. Pré-condições

Aplicativo “EASYT Driver” deve estar sendo executado e o taxista já deve ter efetuado o Login no sistema.

4.4. Fluxo Básico de Eventos

1. O usuário clica em “Ver Mapa”.
2. O usuário visualiza o mapa da aplicação em busca de um passageiro que esteja requisitando um táxi.
3. O usuário encontra um passageiro e aceita sua requisição.

4.5. Pós-condições

4.5.1. Sucesso ao encontrar um passageiro

Usuário conseguiu encontrar um passageiro.

5. Caso de Uso: Aceitar Chamada

5.1. Breve Descrição

O taxista deve ser capaz de aceitar uma requisição de um passageiro ou de uma cooperativa.

5.2. Breve Descrição dos Atores

5.2.1. Usuário

Taxista portador de um dispositivo móvel com sistema operacional Android.

5.3. Pré-condições

Aplicativo “EASYT Driver” deve estar sendo executado e o taxista já deve ter efetuado o Login no sistema e algum passageiro deve ter feito uma requisição para o taxista.

5.4. Fluxo Básico de Eventos

1. O sistema dispara uma notificação para o taxista informando que um passageiro está requisitando os seus serviços.
2. O usuário aceita a requisição.

5.5. Fluxo Alternativo de Eventos

5.5.1. Se o usuário não aceitar a requisição

1. O sistema irá informar ao passageiro que o taxista não aceitou a requisição.

5.6. Pós-condições

5.6.1. Sucesso ao aceitar uma chamada

Usuário conseguiu aceitar uma requisição de um passageiro.

APÊNDICE C – Detalhamento de Casos de Uso da Aplicação da Empresa Prestadora de Serviços de Transporte.

1. Caso de Uso: Visualizar Mapa

1.1. Breve Descrição

A empresa de transporte deve ser capaz de visualizar o mapa na aplicação web.

1.2. Breve Descrição dos Atores

1.2.1. Usuário

Funcionário da empresa de transporte que está acessando a aplicação Web do Sistema.

1.3. Pré-condições

Site do sistema EASYT deve estar sendo acessado e o usuário já deve ter realizado o Login no sistema.

1.4. Fluxo Básico de Eventos

1. O sistema mostra ao usuário a opção de visualizar o mapa.
2. O usuário clica em “Visualizar Mapa”.
3. O sistema mostra o mapa ao usuário.

1.5. Pós-condições

1.5.1. Sucesso ao visualizar o mapa

Usuário conseguiu visualizar o mapa.

2. Caso de Uso: Realizar Login

2.1. Breve Descrição

A empresa de transporte deve ser capaz de realizar o login na aplicação web.

2.2. Breve Descrição dos Atores

2.2.1. Usuário

Funcionário da empresa de transporte que está acessando a aplicação Web do Sistema.

2.3. Pré-condições

Site do sistema EASYT deve estar sendo acessado.

2.4. Fluxo Básico de Eventos

1. O sistema apresenta uma tela com os campos “Nome” e “Senha”.
2. O usuário preenche os campos e clica em “Entrar”.

3. O sistema apresenta a tela inicial da aplicação web.

2.5. Fluxo Alternativo de Eventos

2.5.1. Se o usuário não preencher os campos da tela inicial

1. O sistema informará que não foi possível acessar o sistema.

2.6. Pós-condições

2.6.1. Sucesso ao acessar o sistema

Usuário conseguiu realizar o acesso ao sistema.

3. Caso de Uso: Atualizar Informações das Linhas

3.1. Breve Descrição

A empresa de transporte coletivo deve ser capaz de atualizar/incluir informações de linhas de ônibus e lotações.

3.2. Breve Descrição dos Atores

3.2.1. Usuário

Funcionário da empresa de transporte coletivo que está acessando a aplicação Web do Sistema.

3.3. Pré-condições

Site do sistema EASYT deve estar sendo acessado e o usuário já deve ter realizado o Login no sistema.

3.4. Fluxo Básico de Eventos

1. O usuário clica em “Editar Linhas”.
2. O sistema apresenta uma lista das linhas cadastradas que podem ser atualizadas e um botão para cadastrar uma nova linha.
3. O usuário seleciona a linha que ele deseja atualizar.
4. O sistema apresenta o mapa com o trajeto da linha destacado.
5. O usuário pode alterar/criar o trajeto da linha conforme desejar.
6. O usuário clica em “Avançar” após alterar a linha.
7. O sistema apresenta o resultado das alterações do usuário.
8. O usuário clica em “Salvar”.

3.5. Fluxo Alternativo de Eventos

3.5.1. Se o usuário clicar em “Cadastrar Linha”

1. O usuário deve informar o nome da nova linha.
2. O usuário clica em “Avançar”.
3. O caso de uso vai direto para o quinto passo do fluxo básico.

3.5.2. Se o usuário não preencher todos os campos

1. O sistema informará que todos os campos devem ser preenchidos.

3.6. Pós-condições

3.6.1. Sucesso ao atualizar informações de uma linha

Usuário conseguiu atualizar/incluir uma linha.

4. Caso de Uso: Atualizar Informações das Paradas

4.1. Breve Descrição

A empresa de transporte coletivo deve ser capaz de atualizar/incluir informações de pontos de ônibus no sistema.

4.2. Breve Descrição dos Atores

4.2.1. Usuário

Funcionário da empresa de transporte coletivo que está acessando a aplicação Web do Sistema.

4.3. Pré-condições

Site do sistema EASYT deve estar sendo acessado e o usuário já deve ter realizado o Login no sistema.

4.4. Fluxo Básico de Eventos

1. O usuário clica em “Editar Paradas”.
2. O sistema apresenta um mapa com todas as paradas cadastradas no sistema.
3. O usuário seleciona a parada que ele deseja atualizar.
4. O sistema abre uma janela para o usuário alterar ou excluir a parada.
5. O usuário clica em “Salvar” e as alterações são persistidas (e o sistema permanece na mesma tela).

4.5. Fluxo Alternativo de Eventos

4.5.1. Se o usuário der um clique duplo no mapa, cadastra novo ponto de ônibus

1. O usuário deve confirmar que deseja criar uma parada no ponto que ele clicou.
2. O usuário confirma a operação.

4.6. Pós-condições

4.6.1. Sucesso ao atualizar informações de uma parada

Usuário conseguiu atualizar/incluir uma parada.

5. Caso de Uso: Gerenciar Funcionários

5.1. Breve Descrição

A empresa de transporte coletivo deve ser capaz de gerenciar os funcionários no sistema.

5.2. Breve Descrição dos Atores

5.2.1. Usuário

Funcionário da empresa de transporte coletivo que está acessando a aplicação Web do Sistema.

5.3. Pré-condições

Site do sistema EASYT deve estar sendo acessado e o usuário já deve ter realizado o Login no sistema.

5.4. Fluxo Básico de Eventos

1. O usuário clica em “Funcionários”.
2. O sistema apresenta uma lista com os funcionários do sistema e um botão para cadastrar um novo funcionário.
3. O usuário exclui/atualiza/inclui um funcionário e preenche os campos nome, senha e tipo do funcionário.
4. O usuário clica em “Salvar”.

5.5. Pós-condições

5.5.1. Sucesso ao excluir/alterar/incluir funcionário

Usuário conseguiu excluir/alterar/incluir um funcionário.

6. Caso de Uso: Gerenciar Veículos

6.1. Breve Descrição

A empresa de transporte coletivo deve ser capaz de gerenciar os veículos no sistema.

6.2. Breve Descrição dos Atores

6.2.1. Usuário

Funcionário da empresa de transporte coletivo que está acessando a aplicação Web do Sistema.

6.3. Pré-condições

Site do sistema EASYT deve estar sendo acessado e o usuário já deve ter realizado o Login no sistema.

6.4. Fluxo Básico de Eventos

1. O usuário clica em “Veículos”.
2. O sistema apresenta uma lista com os veículos do sistema e um botão para cadastrar um novo veículo.

3. O usuário exclui/atualiza/inclui um veículo e preenche os campos código e tipo do veículo.
4. O usuário clica em “Salvar”.

6.5. Pós-condições

6.5.1. Sucesso ao excluir/alterar/incluir veículo

Usuário conseguiu excluir/alterar/incluir um veículo.

APÊNDICE D – Requisitos para Implantação do Sistema.

- Servidor:
  - Banco de dados SQL Server 2008 R2 para a camada de dados do sistema.
  - IIS 7.0 com a pool dos aplicativos configurada para .NET Framework 4.0 para a hospedagem dos serviços WCF e da aplicação web.
- Aplicações Móveis:
  - Sistema operacional Android 2.1 ou superior.