UNIVERSIDADE FEDERAL DO PARANÁ
FLÁVIO AUGUSTO CASALI DE SOUZA
JONATHAN COSTA
VALÉRIA PEDRO

TAXI CALLER – UMA PLATAFORMA DE AUTOMAÇÃO PARA CENTRAIS DE TÁXI

CURITIBA
2013

FLÁVIO AUGUSTO CASALI DE SOUZA
JONATHAN COSTA
VALÉRIA PEDRO

TAXI CALLER – UMA PLATAFORMA DE AUTOMAÇÃO PARA CENTRAIS DE TÁXI

Trabalho de Conclusão de Curso apresentado à banca examinadora do curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor de Educação Profissional e Tecnológica da Universidade Federal do Paraná, para a obtenção do grau de Tecnólogo em Análise e Desenvolvimento de Sistemas.

Orientador: Professor Dr. Alessandro Brawerman

CURITIBA
2013

TERMO DE APROVAÇÃO

FLÁVIO AUGUSTO CASALI DE SOUZA
JONATHAN COSTA
VALÉRIA PEDRO

TAXI CALLER – UMA PLATAFORMA DE AUTOMAÇÃO PARA CENTRAIS DE TÁXI

Trabalho de Conclusão do Curso de Tecnologia em Análise e Desenvolvimento de Sistemas, Setor de Educação Profissional e Tecnológica, Universidade Federal do Paraná, pela seguinte banca examinadora:

_____
Prof. Dr. Alessandro Brawerman
Orientador – Setor de Educação Profissional e Tecnológica da Universidade Federal, UFPR.

_____
Prof. Mestre Pedro Rodrigues Torres Júnior
Setor de Educação Profissional e Tecnológica da Universidade Federal, UFPR

_____
Prof. Mestre Mario de Paula Soares Filho
Setor de Educação Profissional e Tecnológica da Universidade Federal, UFPR

Curitiba, 21 de Março de 2013.

RESUMO

É notável nos anos mais recentes o crescimento exponencial do uso de celulares no cotidiano do homem, deixando de ser apenas um meio de comunicação. Os smartfones se tornaram ferramentas multifuncionais e flexíveis, que podem atender aos mais diversos tipos de usuários. Aos desenvolvedores surgiu uma nova oportunidade para se destacar no mercado de trabalho, que é a área de desenvolvimento para aplicativos móveis. Tendo como foco esse novo ambiente que vem ganhando espaço na tecnologia da informação, esse projeto tem como objetivo apresentar uma solução integradora interligando de maneira transparente usuário de táxi, central de táxi e o próprio taxista. A solução é composta por dois aplicativos móveis e um sistema Web central controlador e tem como função principal realizar, de maneira simples e eficiente, o chamado de um táxi, buscando o mais próximo a partir da posição do usuário. Em seu aplicativo, o usuário a partir do GPS localiza sua posição e se necessário complementa as informações e em seguida solicita o táxi. Estas informações são enviadas para um servidor central na nuvem, que através das informações busca o taxista mais próximo e envia o pedido de corrida para o táxi escolhido. O taxista recebe em seu aplicativo o pedido da corrida e confirma o aceite. Finalmente, o usuário recebe uma mensagem informando que o táxi está a caminho, com o número da placa e nome do taxista.

Palavras–chave: Aplicativos móveis, Smartfones, Táxi, integração de sistemas, web services.

ABSTRACT

It is noticeable the exponential cellphone usage in human routine, not only as a communication device. Smartphones became multifunctional and flexible tools that manage to help the majority of users. For software developers a new market opportunity flashed: the mobile application development. Focusing on this new environment that is getting more and more into the information technology area, this project aims on presenting an integrated solution for transparently connecting taxi users, taxi centrals and taxi drivers. This solution presents two mobile applications and a Web Controller system. The objective is to establish, in a simple and efficient way, the cab requisition, which will contact the nearest taxi around the user's location. On the user's front–end, the system uses a GPS–based location and complements the requisition's data, if needed, and then creates a cab requisition. This set of information is sent to a server, located in the Internet cloud, which is able to determine which cab is nearest to the user and then sends the requisition to the driver. The driver receives the requisition at his front–end and then chooses to accept or not the request. In case it is accepted, a message is sent to the user confirming the acceptance informing the cab's license plate and the driver's name.

Keywords: mobile applications, smartphones, taxi, systems Integration, web services.

LISTA DE FIGURAS

LISTA DE TABELAS

LISTA DE SIGLAS

API – Application Programming Interface

BHC – Balanço Hídrico Climatológico

EAP – Estrutura Analítica do Projeto

Embrapa – Empresa Brasileira de Pesquisa Agropecuária

EUA – Estados Unidos da América

GPS – Global Positioning System

HTML – Hyper Text Markup Language

HTTP – Hyper Text Transfer Protocol

IDE – Integrated Development Environment

IETF – Internet Engineering Task Force

IBGE – Instituto Brasileiro de Geografia e Estatística

JSON – JavaScript Object Notation

JSP – Java Server Page

MIME – Multipurpose Internet Mail Extensions

OHA – Open Handset Alliance

OS – Operational System

POC – Prova de conceito

REST – Representational State Transfer

SDK – Software Development Kit

SMS – Short Message Service

SOAP – Simple Object Access Protocol

TCC – Trabalho de Conclusão de Curso

UDDI – Universal Description, Discovery and Integration

URI – Uniform Resource Identifier

WSDL – Web Services Description Language

XML – Extensible Markup Language

SUMÁRIO

1 INTRODUÇÃO

Entre os aparelhos eletrônicos criados pelo homem, o celular é um dos tantos que sofreram evoluções facilmente percebidas pelo usuário. Passando por mudanças que seguem em proporção inversa, o celular teve sua construção física grandemente reduzida, ao ponto em que quase não há botões, apenas uma fina e sensível tela de toque, que cabe na palma da mão. Em contrapartida, as funções contidas em um telefone celular se desenvolveram exponencialmente, não se limitando mais a apenas realizar ligações ou enviar mensagens. Atualmente, os celulares podem acessar a Internet para exibir páginas de todos os tipos, visualizar vídeos e tocar músicas. Diversas outras aplicações como câmera fotográfica, visualização de mapas, agenda, calendário, localização via GPS e muitas outras se tornaram comuns em aparelhos celulares. Junto com o avanço dessa tecnologia portátil, logo chegaria a oportunidade para que desenvolvedores de sistemas pudessem criar suas próprias aplicações que seriam utilizadas por estes chamados smartfones.

Assim teve início a corrida pelo desenvolvimento de aplicações móveis. Este é um setor no mercado que vem ganhando cada vez mais espaço na Tecnologia da Informação. Pesquisa realizada pelo IBGE indica que mais de 60% da população brasileira possui celular (Revista Veja, 2012). Segundo dados obtidos pela Pesquisa Nacional por Amostra de Domicílios em 2011, aproximadamente 70% da população brasileira possui um aparelho de uso próprio (LIMA e SILVA, 2012). A partir disso, percebe–se que a programação para aplicativos móveis é uma tendência que está sendo seguida por desenvolvedores que querem se destacar no mercado, criando aplicações específicas que servem a diferentes tipos de pessoas.

1.1 DESCRIÇÃO DO PROBLEMA

Existe atualmente um alto número de concessionárias, revendedoras e outros tipos de estabelecimentos que trabalham com a venda de automóveis, nem todas as

pessoas podem adquirir um carro ou veículo de uso próprio. Seja por motivos financeiros ou pessoais, existem aqueles que dependem de outros meios de transporte para se locomover entre grandes distâncias, como o ônibus e o táxi. Mesmo aqueles que possuem um automóvel podem se encontrar em uma situação na qual dependam de um meio alternativo. Como por exemplo, um empresário que está de viagem em outro país e acaba de chegar ao aeroporto local. Em casos como esse, a utilização de um táxi é geralmente a opção mais escolhida, devido à sua segurança e eficiência no transporte.

Pessoas que utilizam serviços de táxi com frequência podem, porém, encontrar certos problemas. Em primeiro lugar, há a dificuldade em localizar um táxi próximo e que esteja disponível. É comum para taxistas transitarem ou até mesmo permanecerem parados próximos a um local em que sejam comuns os pedidos de corrida. Ainda assim, há a chance destes táxis estarem ocupados, ou fora do alcance de visão de um possível cliente. Outro problema, mais agravante, são os táxis clandestinos, que disputam clientes com os regulamentados (Revista Veja SP, 2009). Segundo reportagem da revista Veja, esses táxis “piratas” se utilizam de carros disfarçados para aplicar golpes e assaltos, ou então rodando sem taxímetro, com preço definido para uma corrida.

O projeto Taxi Caller visa interligar o usuário de táxi, às centrais de táxi e finalmente aos taxistas de maneira totalmente transparente, automatizando o processo de chamada de um táxi e evitando o uso de táxis clandestinos. No projeto, são utilizados dois aplicativos móveis. Um dos aplicativos é destinado ao cliente que fará um pedido de corrida, enquanto que o outro aos taxistas, que irão receber esses pedidos. Ambos serão desenvolvidos na plataforma Android. O projeto também possui um sistema central em um servidor na nuvem, que é responsável por distribuir os pedidos entre os aplicativos de cliente e taxista. A comunicação entre todos os sistemas se dá via Internet através de ofertas e consumos de Webservices (Serviços Web).

1.2 VISÃO GERAL DA SOLUÇÃO

É possível encontrar aplicativos para celulares que auxiliem pessoas nas mais diferentes tarefas, como encontrar um restaurante ou hotel próximo, acompanhamento de atividades físicas, gerenciamento de tarefas, etc. Tendo isso como base, é possível se aproveitar da praticidade dos aplicativos móveis para oferecer uma solução ao problema dado.

Esse projeto tem como objetivo apresentar uma solução integradora no desenvolvimento de uma aplicação para celular na plataforma Android que atende a uma necessidade específica. A função principal será a de oferecer ao usuário um método simples e eficiente de realizar um pedido de táxi. Após a solicitação do táxi pelo cliente, o pedido é encaminhado para uma central, que fica em funcionamento da nuvem, em um servidor Web. Esta, a partir de um algoritmo de localização e expansão busca o táxi mais próximo e envia a solicitação ao taxista, que também possui um aplicativo e recebe alerta da central. A resposta do taxista, aceite ou não da corrida, volta para o servidor central. Em caso de aceite de corrida, o servidor envia a resposta para o cliente. Por outro lado, em caso negativo, o servidor tenta o segundo táxi mais próximo ao cliente. Esta busca continua até algum taxista aceitar a corrida ou não haver mais táxis disponíveis. Neste último caso, um aviso é enviado ao cliente.

1.3 OBJETIVO GERAL

Desenvolver uma solução integradora de chamadas para táxi, composta por dois aplicativos Android e um sistema na nuvem que atuará como servidor central e que possa atender a demanda dos pedidos de táxis, aprimorando a qualidade nos serviços atuais de táxis.

1.4 OBJETIVOS ESPECÍFICOS

Pretende–se com este projeto:

- Desenvolver o aplicativo do cliente para chamar o táxi;
- Desenvolver o aplicativo do taxista;
- Desenvolver o sistema central que gerencia os pedidos de táxi;
- Desenvolver Web Services que farão a comunicação entre as aplicações dos celulares e o sistema central;
- Desenvolver um algoritmo inteligente para calcular o táxi mais próximo;
- Integrar os aplicativos com o sistema central;

1.5 ESTRUTURA DO PROJETO

Nos próximos capítulos deste projeto será apresentado o desenvolvimento do projeto. O documento foi divido da seguinte maneira, o Capítulo 2 apresenta a fundamentação teórica com o objetivo de posicionar o leitor em relação às tecnologias utilizadas no projeto. No capítulo 3 é detalhado como o sistema foi projetado, a arquitetura do projeto, ferramentas utilizadas como o EAP (Estrutura Analítica do Projeto) e o Diagrama de Gantt. É apresentado também o plano de atividades e o plano de riscos e como foi realizado o desenvolvimento. No capítulo 4 são apresentadas as aplicações, a explicação do projeto e as todas as funcionalidades, juntamente com a validação e os testes. Por fim, o capítulo 5 finaliza o projeto com as considerações finais. Este seguido pelas referências e os apêndices.

2 FUNDAMENTAÇÃO TEÓRICA

Para a melhor compreensão das tecnologias em uso atualmente, nos dispositivos móveis, faz–se necessário abordar os principais conceitos assim como discorrer historicamente sobre o avanço dos mesmos.

2.1 PLATAFORMA ANDROID

Com a evolução da tecnologia, empresas e desenvolvedores buscaram uma melhor plataforma, que fosse ágil e moderna. Por outro lado, para os usuários a preocupação era um visual elegante e moderno, com fácil navegação e uma infinidade de recursos. Tendo em vista que metade da população mundial, aproximadamente três bilhões de pessoas no mundo possui um aparelho celular, as empresas líderes em tecnologia móvel viram o surgimento de um grande mercado a ser explorado (LECHETA, 2011). Ao mesmo tempo, perceberam que haveria a necessidade de uma padronização que permitisse a todas essas empresas atender às expectativas de seus clientes e acompanhar as tendências do mercado. Em união com o Google, empresas como a Sony Ericsson, Toshiba, LG, Samsung e Motorola criaram a Open Handset Alliance (OHA), um grupo com o principal objetivo de criar uma plataforma de desenvolvimento para tecnologia móvel, que fosse poderosa, flexível, de código aberto e de livre acesso aos desenvolvedores (LECHETA, 2011).

O sistema Android em si, contudo, estava em desenvolvimento sem estar diretamente relacionado a então criada OHA. O projeto estava sendo desenvolvido primeiramente pela empresa Android Inc., e foi comprado pela Google no ano de 2005, para então ser inserido no grupo criado pelas empresas de telefonia. Contudo, apenas três anos mais tarde o sistema teria sua primeira versão comercializada, através do celular HTC G1 (LECHETA, 2011), conforme ilustra a Figura 1.

Figura 1 **–** Celular HTC G1 com sistema operacional Android (2008).
Fonte: FayerWayer Brasil

Com a intenção de estimular a inovação, o Google investiu milhões de dólares no patrício de duas edições do “Android Developer Challenges” (IBM Developers, 2009). Meses depois do lançamento do G1, foi então lançando o Android Market (atual Play Store), permitindo assim com que os usuários navegassem e fizessem download de aplicativos diretamente no próprio celular (IBM Developers, 2009).

2.1.1 Características

A plataforma Android possui uma arquitetura muito flexível, na qual pode integrar aplicações nativas com recém–criadas. A construção do Android foi feita com base no sistema de computadores Linux. Baseado no Kernel 2.6, o sistema tem segurança, gerenciamento de memória e controles de rede e drivers (JOBSTRAIBIZER, 2009). A utilização da linguagem Java é um dos destaques no Android, porém não há uma máquina virtual Java (JVM). Há, na verdade, uma máquina virtual otimizada para utilização em dispositivos móveis, a Dalvik. Como o código é aberto e livre, possui um vasto conjunto de ferramentas, como uma rica interface visual, um ambiente flexível, inovador e poderoso e também o acesso à Internet com o uso de tecnologias 3G e wi–fi. (LECHETA, 2011).

Diversas características tornam o Android uma plataforma excepcional. Por possuir código aberto, e de livre acesso, como já foi dito anteriormente, abre–se a possibilidade para diversos programadores criarem seus próprios aplicativos. A possibilidade de livre edição do código também possibilita um maior aperfeiçoamento da plataforma, de modo que programadores possam contribuir com novas funções ou simplesmente corrigir falhas de versões anteriores. Sem qualquer custo adicional e com total liberdade de customização para o uso em seus aparelhos, significando realmente uma grande vantagem (LECHETA, 2011). Outra característica marcante do Android é a sua versatilidade. Há a possibilidade dos fabricantes intervirem no sistema e produzir aparelhos para públicos específicos. Alguns aparelhos com interfaces mais simples são voltados para o público corporativo, enquanto os aparelhos voltados para o público mais jovem têm interfaces mais elaboras e integrações pesadas com as redes sociais (UOL Celular, 2010).

A inserção de produtos no mercado de aplicativos da Google, a Play Store, permite que desenvolvedores enviem seus projetos muito facilmente, sem haver um grande número de condições a serem cumpridas e nem preços elevados a serem pagos. É uma grande vantagem que o Android possui em relação à outra grande empresa que também atua no desenvolvimento de aplicações móveis, a Apple (DEVELOPERS, 2013).

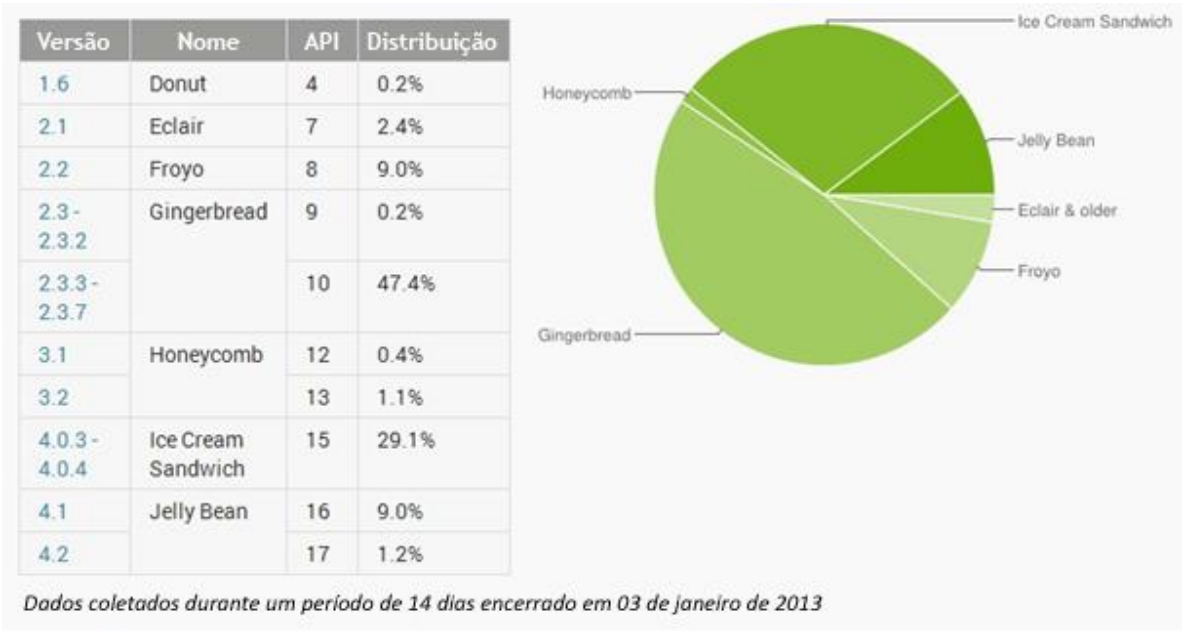
2.1.2 Histórico de versões

Desde o início, o Android foi e continua sendo marcado especialmente por suas diferentes versões desenvolvidas, cada uma possuindo um apelido em especial. Atualmente, o sistema está em sua versão 4.2, presente em smartfones mais atuais e com maior capacidade de hardware (DEVELOPERS, 2013). A TABELA 1 apresenta uma as principais versões oficiais da plataforma Android, junto com suas determinadas nominações e nível de Application Programming Interface (API).

Versão da Plataforma	Nível de API	Codinome
Android 4.2	17	Jelly Bean
Android 4.1	16	
Android 4.0.3	15	Ice Cream Sandwitch MR1
Android 4.0	14	Ice Cream Sandwitch
Android 2.3.4	10	Gingerbread MR1
Android 2.3	9	Gingerbread

Tabela 1 – Principais versões da Plataforma Android (2013).
Fonte: Android Developers.

Outra informação que é de grande importância para os desenvolvedores é em relação à versão mais utilizada pelos clientes. Embora hajam versões mais atualizadas e com maior nível de API, como é caso da Ice Cream Sandwitch e da Honeycomb, a maioria dos usuários ainda possui versões anteriores, em especial, a Gingerbread (DEVELOPERS, 2013), conforme ilustra a Figura 2.



Versão	Nome	API	Distribuição
1.6	Donut	4	0.2%
2.1	Eclair	7	2.4%
2.2	Froyo	8	9.0%
2.3 - 2.3.2	Gingerbread	9	0.2%
2.3.3 - 2.3.7		10	47.4%
3.1	Honeycomb	12	0.4%
3.2		13	1.1%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	29.1%
4.1	Jelly Bean	16	9.0%
4.2		17	1.2%

Dados coletados durante um período de 14 dias encerrado em 03 de janeiro de 2013

Figura 2 – Versões mais utilizadas da plataforma Android (2013).
Fonte: Android Developers.

2.1.3 Comparações de mercado

Além das qualidades descritas acima em relação ao Android, pesquisas realizadas mostram o crescimento e a liderança do sistema operacional da Google nas vendas de smartfones. Em pesquisa realizada pela Gartner, mostra um comparativo do terceiro quarter de 2011 para o de 2012. As vendas dos smartfones Android chegam a 72% do mercado mundial em 2012, um aumento de cerca de 20% em relação ao ano anterior, conforme pode ser visto através da Tabela 2 (Tech Crunch, 2012).

Sistema Operacional	**Fatia de mercado 3Q 2012**	**Fatia de mercado 3Q 2011**
Android	72,4%	52,5%
iOS	13,9%	15%
Research in Motion (BlackBerry)	5,3%	11%
Bada	3%	2,2%
Symbian	2,6%	16,9%
Microsoft	2,4%	1,5%
Outros	0,4%	0,9%

Tabela 2 – Vendas mundiais de celulares por sistemas operacionais móveis (2012).
Fonte: Gartner.

2.2 INTEGRAÇÃO DE SISTEMAS

Webservices é um termo utilizado para referenciar um modelo de integração para aplicações baseadas na web. Através deste, é possível que uma aplicação chame outra para executar tarefas de simples ou até grande grau de complexidade (BEAL, 2010). A integração de sistemas pode ser realizada entre sistemas em servidores distintos, em sistemas que sejam desenvolvidos em linguagens de programação diferentes, não importando se são sistemas Web ou aplicativos móveis.

Os recursos de um serviço web estão disponíveis para que qualquer cliente possa chamar e extrair os dados e soluções oferecidos, funcionando assim, como uma camada de comunicação entre sistemas que desejam disponibilizar suas informações. Existe, porém, algo que diferencie o método de outros, que seria a ausência de uma interface gráfica, deixando esta a cargo de qualquer empresa que deseje utilizar o Webservice (BEAL, 2010).

No projeto Taxi Caller, toda a comunicação entre a central e os aplicativos é realizada através de serviços Web. Os aplicativos acessam os serviços disponibilizados pela central, que faz o gerenciamento dos pedidos de táxi e retorna para os aplicativos as respostas das requisições. Desta forma a central é responsável pela comunicação entre os aplicativos. Esse processo não é visto pelos usuários, sendo a comunicação e os acessos transparente.

Um exemplo de aplicação real de Web services é o sistema de autorização de notas, recentemente implantado pela Secretaria do Estado da Fazenda, em que não é mais necessária a utilização de notas fiscais em papel, mas somente a versão eletrônica emitida pelas empresas e autorizada via Webservices (K19 treinamentos, 2012). O serviço de nota fiscal eletrônica é acessado diretamente através das interfaces de cada empresa, pelo serviço disponibilizado, automatizando o processo e diminuindo os gastos. Além da necessidade de serem acessados diretamente por sistemas e não por pessoas, esse tipo de serviço pode ser disponibilizado através da Internet, ou por uma rede local, criada em uma empresa, por exemplo, para atingir um grande número de sistemas usuários, se necessário.

2.3 Global Positioning System

GPS ou *Global Positioning System* é um sistema de navegação baseado no uso de satélites e criado a partir de uma rede de vinte e quatro satélites colocados em órbita pelo Departamento de Defesa dos EUA (DANA, 1994). Para se computar uma posição exata, são utilizados sinais codificados de quatro satélites distintos,

obtendo assim três dimensões e o tempo marcado no receptor (GARMIN, 2012). A Figura 3 ilustra o funcionamento do GPS através dos quatros satélites.

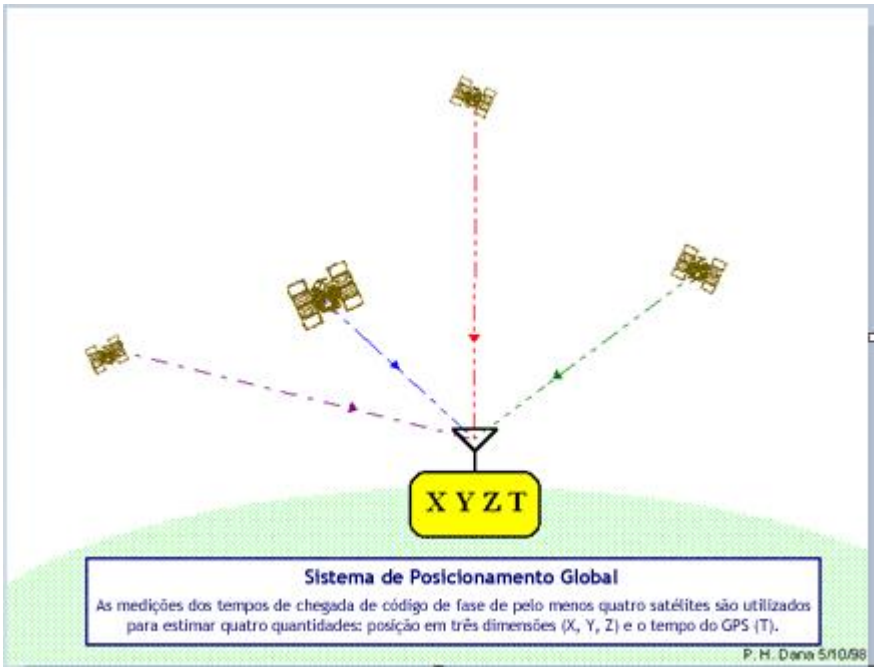


Figura 3 – Funcionamento do GPS (1998).
Fonte: P. H. Dana.

Embora inicialmente desenvolvido para fins militares, o GPS tornou–se uma tecnologia livre a partir dos anos oitenta, e, atualmente, é utilizada amplamente em diversos aparelhos, como navegadores ou celulares. Serviços baseados em GPS também se encontram em aplicações na Internet, como é o caso do Google Maps, um dos mais populares.

No projeto Taxi Caller, o GPS tem um papel fundamental, pois é a partir do posicionamento do usuário e do posicionamento dos táxis que o sistema central seleciona qual veículo pode realizar o transporte do usuário.

2.3.1 Google Maps

Possuindo a própria API, o Google Maps é um serviço conhecido principalmente pela sua ferramenta de traçar rotas. A partir de um ponto de saída e

um de chegada, a aplicação exibe o caminho a ser seguido pelo usuário, passo a passo e diferenciando entre os possíveis veículos (MOBILE, 2011), conforme ilustra a Figura 4.

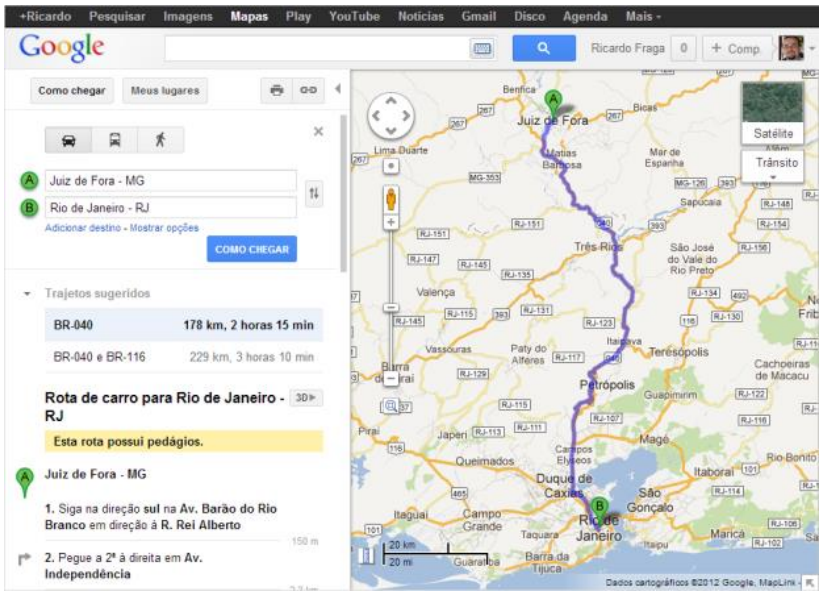


Figura 4 – Tela do Google Maps (2012).
Fonte: Google

Não tardaria para que o Maps se tornasse uma ferramenta desejada em aparelhos móveis, e até o fim de 2008, a Google teria a aplicação disponível para diversos sistemas operacionais móveis. Entre eles o iOS, BlackBerry, Palm, Symbian e Windows Mobile. O Google Maps também está disponível para a plataforma Android, sendo possível à utilização de sua API para o desenvolvimento de aplicativos e no suporte ao GPS, pois nesta plataforma, os aplicativos de GPS que utilizam o Google Maps, a cada atualização de posicionamento, o mesmo vai atualizando o mapa (MOBILE, 2011).

O projeto Taxi Caller utiliza o Google APIs, na qual dentro da mesma existem bibliotecas para a utilização de mapas e também para definir a localização. Através da localização é possível pegar as coordenadas onde estão os usuários.

2.4 TRABALHOS RELACIONADOS

Após ser feita uma pesquisa, foram encontrados trabalhos acadêmicos e aplicativos comerciais que se utilizam dos mesmos conceitos e tecnologias apresentados anteriormente, ou seja, aplicações móveis que trabalham com integração via serviços Web. Alguns destes trabalhos foram selecionados e são apresentados resumidamente nas seções seguintes.

2.4.1 Processamento dirigido de rotas através de texto–fala

Em seu trabalho de conclusão de curso, Adriano Flach de Araújo (2012), desenvolveu um aplicativo para a plataforma Android com o objetivo de atender pessoas com deficiência visual. Através de texto–fala, o aplicativo recebe o local de origem e o de destino do usuário e o orienta em sua trajetória.

Foram utilizadas duas APIs diferentes para o projeto. Uma para converter a fala humana em texto, e a API do Google Maps, que foi usada para trabalhar com a localização e para guiar o usuário.

Este trabalho se assemelha ao Taxi Caller devido à utilização de Webservices com uso de JSON e de tecnologia Google Maps. No caso do trabalho de Araujo, foram utilizados serviços Web para integrar o aplicativo com a parte dos Mapas, para traçar uma rota para o cliente.

2.4.2 Protótipo de sistema de compartilhamento de arquivos e mensagens

O aplicativo, apresentado também como um trabalho de conclusão de curso, por Cesar Augusto Kuehl (2012), permite a troca de mensagens de texto entre usuários que estejam geograficamente próximos. A utilização da plataforma Android

facilitou a obtenção das coordenadas de posicionamento global dos usuários, o que é feito pela própria API do sistema operacional.

Para efetuar o cálculo de distância entre usuários, foi utilizada a fórmula matemática de Haversine (Madden 2011). A comunicação entre os usuários é feita através de serviços Web, acessados pela API da plataforma Android. Para os downloads de arquivos é utilizado o protocolo padrão HTTP.

Vê–se novamente a semelhança, no trabalho proposto por Kuehl e no projeto Taxi Caller, no uso de uma tecnologia que vêm cada vez mais dominando o mercado quando se fala de integração de sistemas, os serviços Web.

2.4.3 Easy Taxi

Aplicativo desenvolvido pela empresa de mesmo nome que oferece diversas funcionalidades para os clientes que desejam chamar um táxi (Gogole Play Store). Possuindo em seu sistema o cadastro de todos os clientes e taxistas, a aplicação possui um método de rastreamento que oferece mais segurança a seus usuários. Também é possível que um cliente acompanhe pelo mapa, em tempo real, o táxi enquanto este faz seu caminho até o endereço marcado.

É possível para o cliente fazer uma ligação ao taxista com apenas um toque enquanto o aplicativo está em funcionamento. Tendo suas principais áreas de funcionamento em São Paulo e Rio de Janeiro, o Easy Taxi é um aplicativo gratuito que é usado a partir da versão 2.1 do Android e atualizado frequentemente.

Com o mesmo objetivo e mais simplicidade o Taxi Caller efetua os pedidos de táxi. A principal característica é a utilização de opções de *Shared Preferences* e telas mais simples, e também sem a visualização da localidade do taxista, já que o sistema encontra o taxista automaticamente para o usuário.

2.4.4 99 Taxis

Aplicativo gratuito desenvolvido por Ariel Lambrecht e Renato Freitas, o 99 Taxis se utiliza da plataforma Android a partir da versão 1.6 e também é disponível para celulares iPhone (Empresa 99 Taxis). Entre suas funcionalidades, há o cadastro de usuários para taxistas e clientes e acompanhamento do táxi em tempo real pelo mapa. Um diferencial da aplicação é o método para encontrar a localização do cliente através de triangularização das antenas de celular, ou seja, é realizado um cálculo de distância comparando a antena dos celulares do cliente, do taxista e do servidor central.

No projeto Taxi Caller, a busca pelo táxi mais próximo é feita através de um algoritmo inteligente, que recebe a posição do cliente e retorna o táxi mais próximo, tornando a resposta mais precisa e rápida.

2.5 Diferenciais do projeto

O projeto Taxi Caller busca ser simples e eficaz. Para chamar o táxi, o cliente precisa apenas apertar um botão. Essa chamada é enviada para a central, que é capaz de localizar o melhor táxi para o usuário, com a possibilidade de um chegada mais rápida. Para realizar a localização do melhor táxi, o servidor possui um algoritmo inteligente, que faz todo processamento de busca e cálculos para encontrar o táxi mais próximo possível. Após a aceitação da corrida pelo taxista, o usuário recebe na tela a informação de qual táxi será o seu transporte, com a placa do carro e nome do taxista. Esse processo é feito de forma transparente para o usuário e o taxista. Outro grande diferencial do projeto é ser adaptável para qualquer cooperativa de táxi, não ficando preso somente a uma.

3 METODOLOGIA

Este capítulo visa apresentar detalhes de especificação do sistema, bem como relatar os passos de desenvolvimento do mesmo. Além destes, são apresentados também o cronograma, toda estrutura do projeto através da EAP (Estrutura Analítica do Projeto) e do Diagrama de Gantt.

3.1 Visão Geral

O projeto Taxi Caller é uma solução integradora composta por três sistemas distintos, que possuem comunicação frequente. São estes, dois aplicativos para a plataforma Android, e um sistema central, que atua como um sistema controlador que realiza as operações de comunicação entre os dois aplicativos e que realiza através de um algoritmo inteligente a busca pelo táxi mais próximo do cliente.

A aplicação móvel desenvolvida especificamente para os clientes do Taxi Caller possui a única função de enviar um pedido de táxi para a central. Uma vez iniciado o aplicativo, o usuário inicia o processo de busca de táxi. O aplicativo do cliente solicita o táxi com apenas um clique. Quando o táxi é solicitado, é enviada uma requisição para a central com a localização do cliente. A central por sua vez recebe também a posição dos táxis constantemente. Cruzando estas informações, o sistema busca o táxi mais próximo que está disponível e envia um pedido ao taxista.

Ao receber o pedido de corrida, em seu aplicativo, o taxista pode aceitá–lo ou rejeitá–lo. Assim que o pedido é respondido, a resposta é enviada para a central que controla as solicitações. Quando o pedido for aceito por um taxista é enviada a resposta para o cliente com o nome do taxista e a placa do carro. O processo descrito acima está representado pela Figura 5.

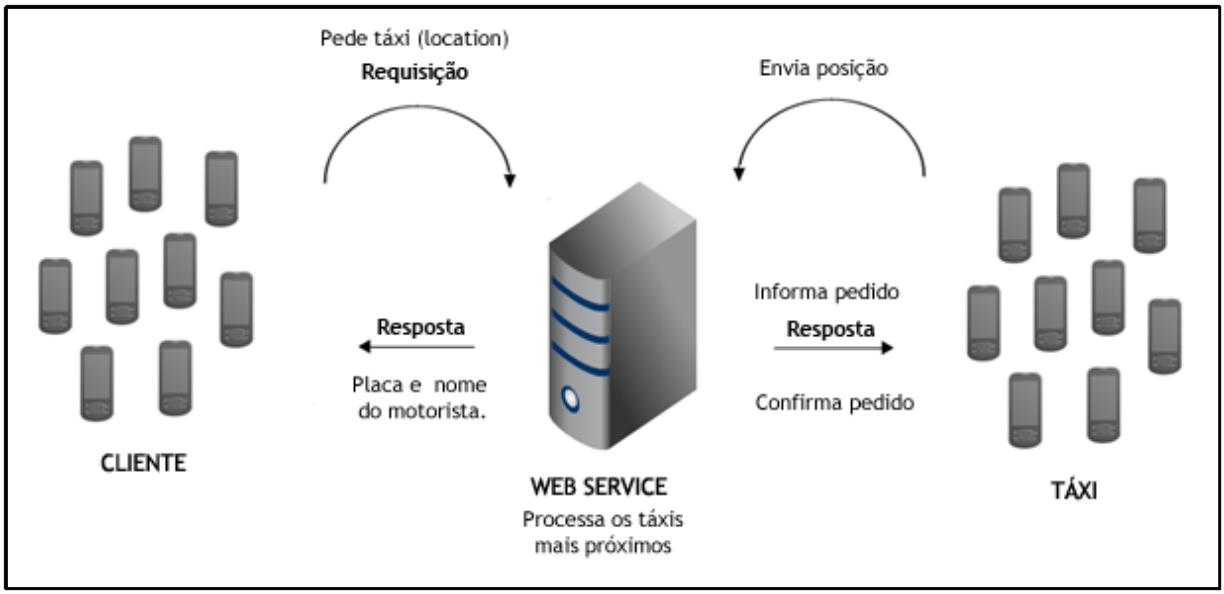


Figura 5 – Arquitetura do projeto (2012).
Fonte: Os autores.

3.2 Engenharia de Software e Análise do Sistema

Durante o levantamento de requisitos nas primeiras etapas, foi necessário modelar os sistemas. Para isso foi utilizada a UML (Linguagem de Modelagem Unificada), baseada no paradigma de Orientação a Objetos. A UML é uma linguagem visual utilizada para modelar sistemas computacionais. O objetivo da mesma é auxiliar na processo de Engenharia de Software, definindo características, comportamentos, estruturas lógicas (GUEDES, 2011).

Com o objetivo de apresentar as estruturas e modelagens do projeto foram utilizados três diagramas da UML:

- Diagrama de casos de uso;
- Diagrama de classes;
- Diagrama de sequência;

A documentação completa com todos os diagramas encontra–se nos apêndices, ao final deste documento.

3.3 Plano de Atividades

Baseado na elaboração da estrutura analítica do projeto (EAP), o plano de atividades apresenta uma visão macro das atividades desenvolvidas durante o projeto. As etapas foram divididas de acordo as definições dos primeiros encontros. A definição do projeto, foram feitas pesquisas sobre o tema e o levantamento de requisitos. Em seguida durante o planejamento feita toda análise para o desenvolvimento do projeto. Na fase do desenvolvimento foram necessários estudos de tecnologias, para isso foram utilizadas as POCs (Prova de Conceitos), dessa forma durante os estudos eram feitos exemplos para ser aplicados diretamente no projeto e somente após então a implementação. Foram realizados testes durante algumas semanas. E na finalização do projeto toda a documentação foi concluída juntamente com a preparação para a apresentação do projeto. A Figura 6 apresenta a EAP do projeto Taxi Caller.

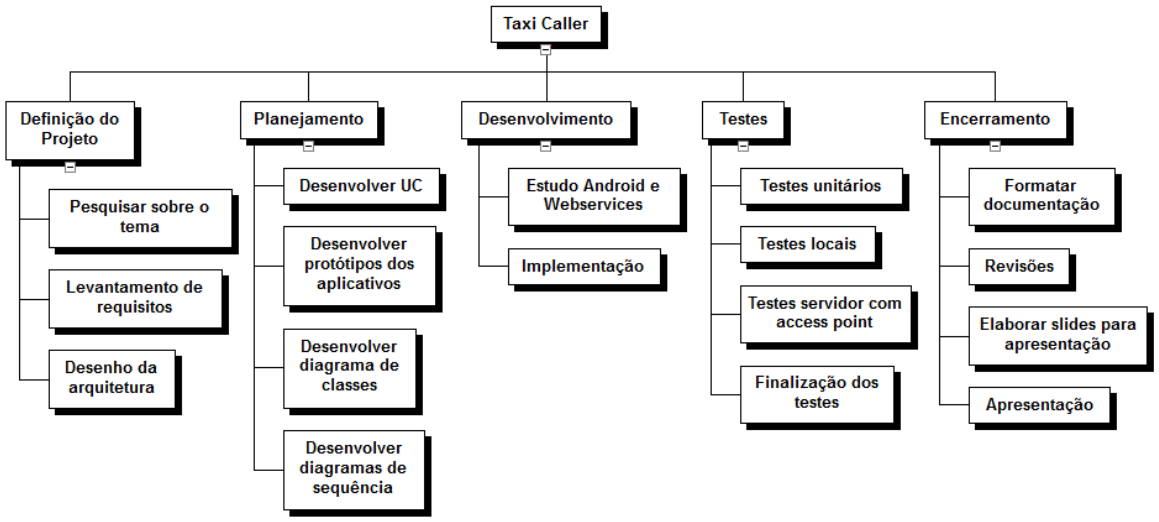


Figura 6 – Estrutura Analítica do Projeto (EAP) (2012).
Fonte: Os autores.

Para elaborar o cronograma e diagrama de Gantt, no qual se esboçam e define as atribuições das atividades de acordo com as datas previstas para entrega de artefatos, foi utilizada a ferramenta Microsoft Project 2007. Este diagrama pode ser visualizado na Figura 7.

	Nome da tarefa	Duração	Início	Término
1	**Taxi Caller**	**98 dias?**	**Sex 02/11/12**	**Ter 19/03/13**
2	**Definição do Projeto**	**10 dias?**	**Sex 02/11/12**	**Qui 15/11/12**
3	Pesquisar sobre o tema	5 dias?	Sex 02/11/12	Qui 08/11/12
4	Levantamento de requisitos	5 dias?	Sex 09/11/12	Qui 15/11/12
5	Desenho da arquitetura	5 dias?	Sex 09/11/12	Qui 15/11/12
6	**Planejamento**	**15 dias?**	**Sex 16/11/12**	**Qui 06/12/12**
7	Desenvolver UC	5 dias?	Sex 16/11/12	Qui 22/11/12
8	Desenvolver protótipos dos aplicativos	5 dias?	Sex 16/11/12	Qui 22/11/12
9	Desenvolver diagrama de classes	5 dias?	Sex 23/11/12	Qui 29/11/12
10	Desenvolver diagramas de sequência	5 dias?	Sex 30/11/12	Qui 06/12/12
11	**Desenvolvimento**	**70 dias?**	**Sex 09/11/12**	**Qui 14/02/13**
12	**Estudo Android e Webservice**	**26 dias?**	**Sex 09/11/12**	**Sex 14/12/12**
13	POC 1 - Aplicativo Calculador (App celular e serviço no PC)	10 dias?	Sex 09/11/12	Qui 22/11/12
14	POC 2 - App celular pegar localização e enviar para servidor via WS JSON	10 dias?	Sex 23/11/12	Qui 06/12/12
15	POC 3 - App celular (táxi) enviando posicionamento via WS JSON e imprimir	10 dias?	Sex 23/11/12	Qui 06/12/12
16	POC 4 - App celular - Fazer Splash Screen e após tela inicial com mapa	5 dias?	Sex 07/12/12	Qui 13/12/12
17	POC 5 - Servidor central recebe posicionamento e imprime	5 dias?	Sex 07/12/12	Qui 13/12/12
18	POC 6 - App do táxi - Implementar conceito de livre/ocupado	5 dias?	Sex 07/12/12	Qui 13/12/12
19	POC 7 - Gerenciamento de múltiplas conexões (threads servidor)	1 dia?	Sex 14/12/12	Sex 14/12/12
20	**Implemantação**	**44 dias?**	**Seg 17/12/12**	**Qui 14/02/13**
21	Aplicativo cliente enviar pedido para central (com todas informações)	24 dias?	Seg 17/12/12	Qui 17/01/13
22	Central simula táxis virtuais e retorna para aplicativo	24 dias?	Seg 17/12/12	Qui 17/01/13
23	Aplicativo táxi enviando posição para central via WS JSON	24 dias?	Seg 17/12/12	Qui 17/01/13
24	Central simula pedidos e retorna para táxi	24 dias?	Seg 17/12/12	Qui 17/01/13
25	Unificar central do táxi e do cliente em uma só central	5 dias?	Sex 18/01/13	Qui 24/01/13
26	Todas as aplicações em funcionamento interligadas	15 dias?	Sex 25/01/13	Qui 14/02/13
27	**Testes**	**20 dias?**	**Sex 15/02/13**	**Qui 14/03/13**
28	Testes unitários	5 dias?	Sex 15/02/13	Qui 21/02/13
29	Testes locais com 1 táxi local e um pedido local	5 dias?	Sex 22/02/13	Qui 28/02/13
30	Testes com mais táxis e mais clientes solicitando com access point	5 dias?	Sex 01/03/13	Qui 07/03/13
31	Finalização dos testes	5 dias?	Sex 08/03/13	Qui 14/03/13
32	**Encerramento**	**24 dias?**	**Qui 14/02/13**	**Ter 19/03/13**
33	Formatar documentação	16 dias?	Qui 14/02/13	Qui 07/03/13
34	Revisões	3 dias?	Sex 08/03/13	Ter 12/03/13
35	Elaborar slides para apresentação	5 dias?	Sex 08/03/13	Qui 14/03/13
36	Apresentação	1 dia?	Ter 19/03/13	Ter 19/03/13

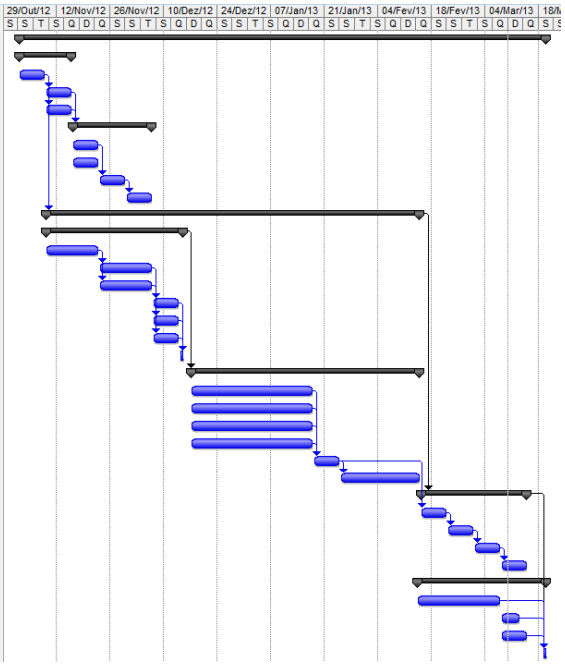


Figura 7 – Cronograma do projeto Taxi Caller (2012).
Fonte: Os autores.

3.4 Plano de Riscos

O pouco conhecimento da plataforma Android pelos membros da equipe e também na integração de sistemas com Web services, causa uma deficiência técnica e a necessidade algumas semanas de estudo até existir a possibilidade de iniciar as implementações.

A falta de precisão nos diagramas e erros de análise também pode ser considerado um risco, já que poderia acarretar problemas no desenvolvimento. Para tentar impedir que aconteçam problemas relacionados, serão elaborados protótipos, para representar os requisitos necessários. O plano de riscos encontra–se no Apêndice E.

3.5 Responsabilidades

Durante o início do projeto Taxi Caller, todos os integrantes do grupo estavam realizando os levantamentos de requisitos e as atividades para conhecer a plataforma Android. Com o início do desenvolvimento da aplicação foi necessário dividir as tarefas para obter um maior controle. Desta maneira Jonathan e Valéria focaram no desenvolvimento e auxiliaram na documentação, enquanto Flávio focou na elaboração da documentação e na análise. A Figura 10 ilustra como foi realizada a divisão das atividades.

Figura 8 – Divisão de atividades (2013).

Fonte: Os autores.

3.6 Materiais

Para o desenvolvimento do sistema completo, foram escolhidas determinadas ferramentas de programação destinadas ao ambiente Android. Primeiramente, foi necessário instalar o Software Development Kit (SDK) da plataforma Android (http://developer.android.com/sdk/index.html). A programação propriamente dita foi feita através do Integrated Development Environment (IDE) Eclipse (http://www.eclipse.org/downloads/packages/eclipse–ide–java–ee–developers/), o qual é destinado especificamente ao desenvolvimento de aplicações Java, incluindo Java Web, Servlets e Java Server Page (JSP). A IDE Eclipse, porém, não possui base para implementação de aplicações Android por conta própria. Foi necessário adicionar ao IDE o plugin Android Development Tools (ADT) (http://developer.android.com/tools/sdk/eclipse–adt.html), que adiciona o módulo de visão para a criação e edição de projetos em sistemas móveis.

O Eclipse foi escolhido como ferramenta principal de desenvolvimento devido a seu grande número de opções, em especial quanto à plataforma Android. A IDE oferece meios para livre edição da interface dos aplicativos desenvolvidos e também uma visão de informações e processos que ocorrem durante a execução de um projeto, seja em um emulador ou em um autêntico celular Android. Há inclusive ferramentas que permitem acessar controles e funções especiais do emulador, como a configuração de posição para o GPS, o que se mostrou muito útil para testes da aplicação desenvolvida.

3.6.1 Webservices

Modelos Webservices funcionam através da conjunção de padrões de quatro diferentes tecnologias, que trabalham sobre um protocolo de Internet principal, sendo *Hyper Text Transfer Protocol* (HTTP) o mais usado (BEAL, 2010).

3.6.1.1 Extensible Markup Language (XML)

Desenvolvida pela World Wide Web Consortium (W3C), comunidade internacional para desenvolvimento de padrões na Web, a XML é uma linguagem semelhante ao *Hyper Text Markup Language* (HTML), porém direcionada à transferência de dados, e não à sua exibição. Ela permite que os desenvolvedores criem as próprias tags personalizadas, permitindo que haja definição, transmissão, validação e interpretação de data entre aplicações (WS3SCHOOLS, 2013).

3.6.1.2 Simple Object Access Protocol (SOAP)

Protocolo de comunicação designado para a Internet, baseado em XML, para codificar informações contidas em mensagens de pedido ou de resposta em serviços web. As mensagens SOAP são independentes de qualquer sistema operacional, e pode ser transmitidas por diferentes protocolos de Internet, como HTTP ou *Multipurpose Internet Mail Extensions* (MIME) (WS3SCHOOLS, 2013).

3.6.1.3 WSDL (Web Services Description Language)

Também baseada em XML, é uma linguagem usada para especificar a localização e as operações ou métodos presentes em um serviço Web. Desenvolvida pela Microsoft e IBM em conjunto, WSDL é também um componente integral de UDDI (WS3SCHOOLS, 2013).

3.6.1.4 UDDI (Universal Description, Discovery and Integration)

Se utilizando de diversos padrões definidos pela W3C e IETF, UDDI é um framework que independe de plataforma de uso, usado para descrever e integrar serviços de negócios através da Internet. A descrição das interfaces aos serviços Web é feita através da linguagem WSDL, enquanto o protocolo SOAP é utilizado para a transferência da informação, que por sua vez é dividida em tags com XML (WS3SCHOOLS, 2013).

Empresas de qualquer tamanho podem se beneficiar do uso de UDDI, podendo descobrir o negócio certo a partir de dada situação e expandir o comércio a partir disso, atraindo novos clientes ao mesmo tempo em que melhora o acesso dos atuais. Através dessas soluções para as necessidades direcionadas a clientes, a empresa facilmente conseguirá sua entrada na economia global da Internet. O UDDI é um projeto realizado com a cooperação de várias empresas provedoras de software, como Dell, HP, Intel, Microsoft e Oracle (WS3SCHOOLS, 2013).

3.6.2 Funcionamento Webservices

Explicados os quatro conceitos, é possível demonstrar o funcionamento de um Webservice ilustrado pela Figura 9. Após interagir com a interface web, usada como *front–end*, o usuário irá enviar os dados, e a aplicação entrará em contato com o provedor UDDI para buscar o método necessário para realizar a conversão. Uma vez estando à mensagem associada ao serviço requisitado e sua locação, o provedor retorna para o cliente um arquivo em WSDL, o qual a aplicação completa como uma mensagem SOAP. Esta, por sua vez, é enviada ao servidor da aplicação que contém o serviço web necessário para efetuar a conversão. Com base nas instruções SOAP, o Web Service executa sua tarefa, de acordo com os parâmetros recebidos, e envia o resultado obtido com a conversão para o cliente (ACUNETIX, 2013).

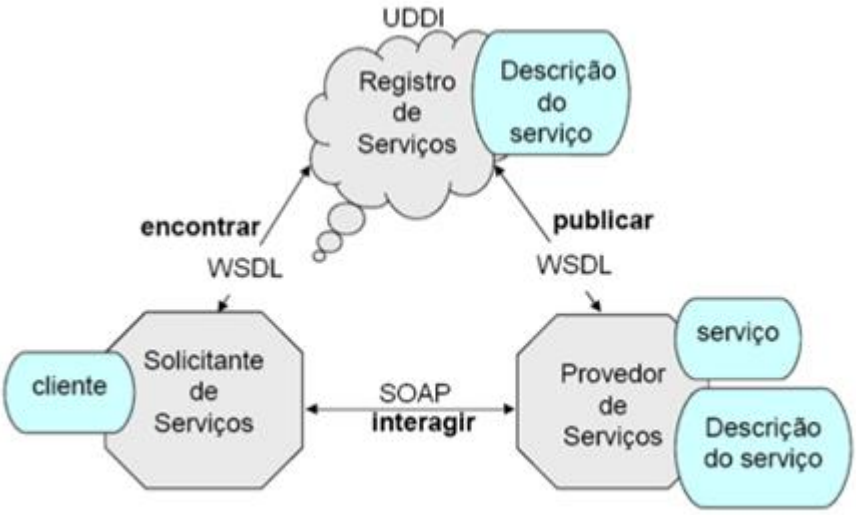


Figura 9 – Funcionamento de um *Web Service* (2011).
Fonte: Universidade Atlântica.

3.6.3 Webservice REST

Representational State Transfer (REST) é um conjunto de princípios de arquitetura pelos quais é possível criar serviços Web que são focados nos recursos de um sistema. No mesmo está incluso como o estado destes recursos estão endereçados e enviados através de HTTP por um vasto alcance de clientes, escritos em linguagens diferentes.

A tecnologia foi introduzida primeiramente no ano de 2000, na Universidade da Califórnia, por Roy Fielding (Fielding, 2000). No início, a ideia não mostrou sinais de que seria adotada por muitos. Entretanto, anos após seu surgimento, REST ganharia vários frameworks em que seria trabalhado. Inclusive, devido à sua simplicidade, os serviços Web RESTful tornaram–se mais utilizados do que aqueles que se utilizam de SOAP e WSDL (RODRIGUEZ, 2008).

Em relação a seu funcionamento, o modelo REST especifica restrições que, quando aplicadas em um serviço Web, induzem propriedades vantajosas, como desempenho, escalabilidade e modificabilidade, permitindo que estes serviços tenham um desempenho muito melhor (TYAGI, 2006).

A arquitetura REST considera informação e funcionalidade como recursos que são acessados através de *Uniform Resource Identifiers* (URI), ou seja, links da Internet. Através do modelo, os clientes e servidores trocam informações usando interface e protocolo padronizados. A arquitetura em si é baseada em uma relação Cliente–Servidor, realizando comunicações por um protocolo sem estado (*stateless*), ou seja, em que cada requisição é uma transação única, sem relação alguma com qualquer outra feita anteriormente.

Serviços Web RESTful usualmente definem os métodos em quatro categorias básicas: GET (obter um recurso), POST (criar um recurso e outras operações), PUT (criar ou atualizar um recurso) e DELETE (deletar um recurso) (ORACLE, 2010), conforme ilustrado pela Figura 12.

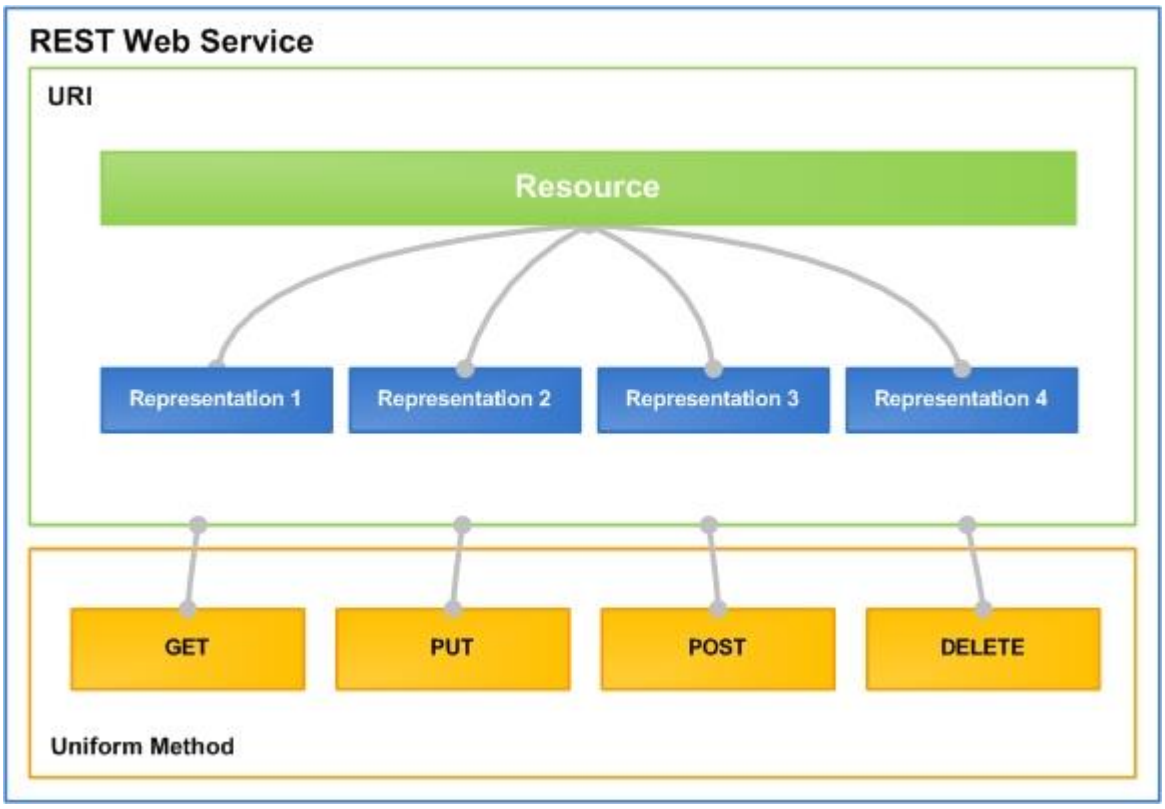


Figura 10 – Arquitetura de Web Service RESTful (2012).
Fonte: CrazyLearner.

Embora tenha seus pontos fortes, o modelo REST também possui suas desvantagens, como por exemplo, sua dependência quanto ao modelo HTTP para a transferência de dados. A Tabela 3 apresenta um comparativo entre os dois tipos de Webservices, dando enfoque aos pontos positivos e negativos de cada um.

	SOAP	REST
Prós	Independe de linguagem, plataforma ou método de transporte.	Independe de linguagem ou plataforma.
	Feito para lidar com ambientes de computação distribuídos	Desenvolvimento mais simples, dependendo menos de ferramentas
	Padrão prevalecente para serviços Web, possuindo melhor suporte de outros padrões (WSDL) e ferramentas de uso.	É conciso, sem a necessidade de uma camada adicional para mensagens.
Contras	Maior dificuldade conceitual e maior peso de execução.	Assume um modelo de comunicação ponto–a–ponto, não utilizável em ambientes de computação distribuídos.
	Mais difícil conceitualmente e de desenvolver, necessitando mais de ferramentas.	Carece de padrões de suporte para serviços que possuem requerimentos mais sofisticados.

Tabela 3 – Comparativo entre os dois tipos de *Web Service* (2008).
Fonte: SPIES.

3.6.4 JavaScript Object Notation (JSON)

JSON é um formato de texto leve e independente de plataforma, derivado dos literais de Java Script, usado para intercâmbio e serialização de dados (ECMA International, 2011). O modelo suporta quatro tipos primitivos de variáveis (inteiro, string, booleano e nulo), e dois tipos de variáveis estruturadas (arrays e objetos). Simplicidade é a maior vantagem de JSON, se mostrando mais eficientes em aplicações Web Ajax do que XML.

Entre outras características que colocam o JSON à frente do XML, está sua maior curva de aprendizado e sua simplicidade quanto a decisões de formatação, uma vez que o formato fornece métodos mais diretos de se mapear aplicações (CROCKFORD, 2006). Entretanto, por ser uma tecnologia razoavelmente inédita, JSON não possui um alto índice de suporte de fornecedores ou de adoção em grandes sistemas. A expectativa, porém, é de que isso mude futuramente (AZIZ, 2007).

JSON vêm se tornando uma das técnicas mais utilizadas para troca de dados entre aplicativos móveis e sistemas Web, via Webservices, já que encapsula a informação trocada em objetos, serializando e deserializando o mesmo de forma transparente ao programados (K19 Artigos, 2011).

No projeto Taxi Caller o JSON é usado para trocar as informações entre os aplicativos e a central. A simplicidade e o pequeno tamanho faz com que o envio do JSON entre as aplicações seja rápido e muito eficiente.

3.7 DESENVOLVIMENTO DO PROJETO

Desde o início com a proposta de realizar um projeto que gerenciaria a central de táxis, o que viria proporcionar maior facilidade para pedir um táxi e também maior agilidade na procura pelo veículo mais próximo.

Durante os encontros semanais da equipe, foram feitas análise, modelagem , protótipos e a arquitetura do projeto que viria a ser desenvolvido. A partir destas reuniões, os requisitos e o escopo do projeto foram sendo elaborados, de forma que a partir de então fosse possível elaborar datas de entrega e os objetivos da equipe.

Para acompanhar as atividades e os prazos de entrega, foi utilizada a EAP (Estrutura analítica de projeto), juntamente com o cronograma e o diagrama de Gantt. A função da EAP era apresentar a divisão das atividades durante cada fase do projeto de acordo com o cronograma, sempre com data para se começar e terminar.

Para o desenvolvimento do projeto optou–se pela plataforma Android, por ser de código aberto e baseado na linguagem de programação Java, a qual toda equipe já estava habituada a utilizar.

Ao final do desenvolvimento das aplicações com todas as interfaces e serviços Web em funcionamento foram realizados vários casos de teste para corrigir falhas que impediriam o bom funcionamento do projeto.

3.8 TESTES E VALIDAÇÃO

Após a implementação do projeto Taxi Caller, foram elaborados alguns cenários para fazer testes e simular o funcionamento por completo.

3.8.1 EXPERIÊNCIA 1 – UM CLIENTE PARA UM TAXISTA

Esta experiência foi realizada utilizando um celular Android com o aplicativo do lado cliente instalado, em conjunto com um emulador Android no qual estava instalado o aplicativo do lado taxista. O objetivo do teste foi realizar o processo completo de envio de pedido, busca do táxi e aceitação do mesmo pedido.

No primeiro momento, a aplicação do cliente busca o endereço através do GPS e conexão a Internet, como mostra a figura 11. Enquanto isso, a aplicação do cliente, com suas informações ajustadas como consta na figura 12, continuamente envia sua posição para a central. É possível visualizar o estado da tela, que está em aguardo de um pedido, na figura 13.

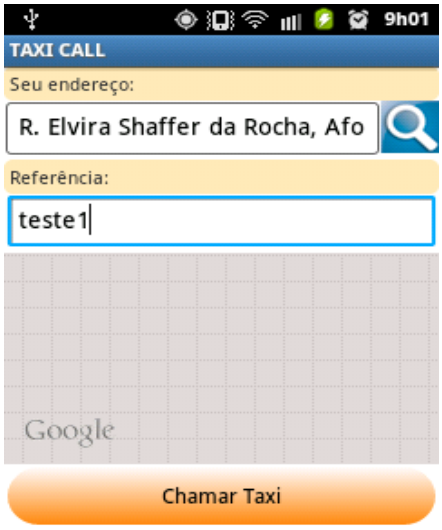


Figura 11 - Tela de início do aplicativo usuário (2013).
Fonte: Os autores

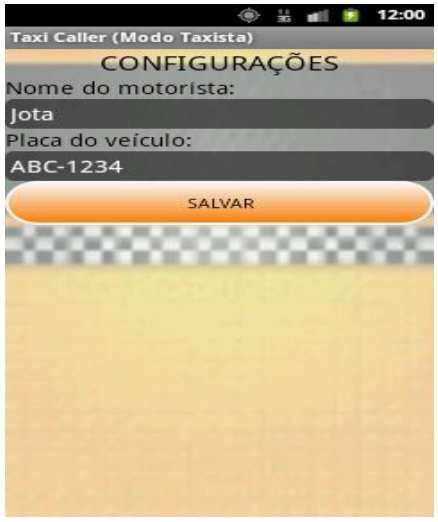


Figura 12 – Tela de configurações do táxi (2013).
Fonte: Os autores

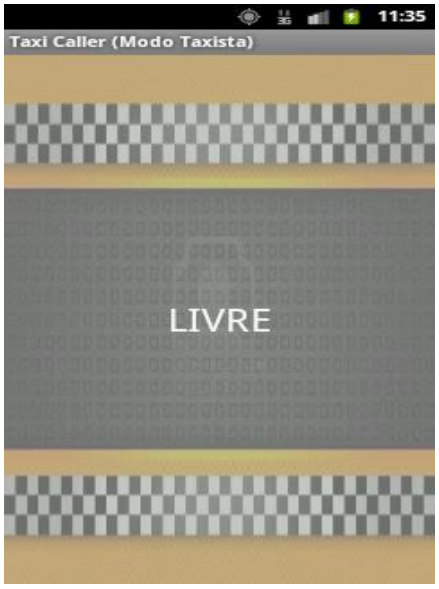


Figura 13 – Tela principal do aplicativo do taxista (2013).
Fonte: Os autores

Em seguida, foi enviado o pedido com as informações constadas no aplicativo do cliente, que automaticamente abre a tela de busca, exibida na figura 14. O taxista logo recebeu o pedido, como é comprovado na tela obtida pela figura 15. O pedido então foi aceito, enviando a confirmação para a central. Essa confirmação foi recebida com sucesso pelo aplicativo do cliente, comprovado pela figura 16.

Figura 14 – Tela de busca do aplicativo do usuário (2013).
Fonte: Os autores

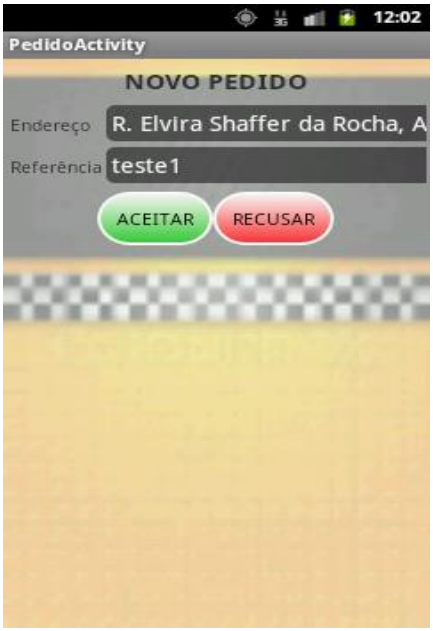


Figura 15 – Tela de pedido do aplicativo do taxista (2013).
Fonte: Os autores

Figura 16 – Tela de pedido aceito do aplicativo do usuário (2013).
Fonte: Os autores

Após aceitar o pedido, o aplicativo do taxista retornou para a tela inicial, porém agora com a mensagem de que o táxi está em serviço, como é exibido na figura 17. Logo, não está mais enviando a própria posição á central e não poderá receber novos pedidos. Através de mensagens emitidas pela central, é possível confirmar que o táxi em primeiro momento enviou sua posição repetidamente, e que então um pedido foi recebido pela central. Esta então calculou o táxi mais próximo,

encontrando o táxi da experiência. A mensagem de pedido aceito também foi emitida pela central, com as informações corretas do taxista. Todo o processo pode ser confirmado pela figura 18, que possui todas as mensagens emitidas pela central durante a experiência.

Figura 17 – Tela principal em serviço do aplicativo do taxista (2013).
Fonte: Os autores

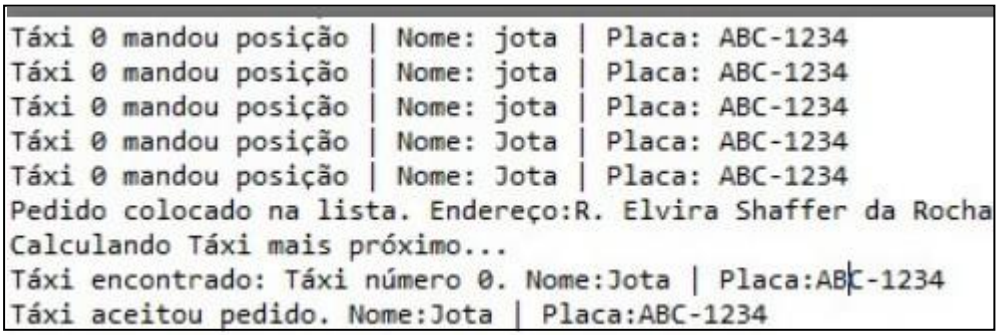


Figura 18 – Log do processo do sistema central (2013).
Fonte: Os autores

3.8.2 EXPERIÊNCIA 2 – UM CLIENTE PARA DOIS TAXISTAS

O objetivo desta experiência foi simular uma situação em que o cliente possui seu primeiro pedido recusado pelo taxista, mas em seguida recebido e aceito pelo próximo taxista na lista. Para isso foram utilizados dois emuladores Android que possuíam instalados o aplicativo do lado taxista, e um celular Android com o aplicativo do lado cliente instalado para o envio do pedido. O endereço e a referência enviados pelo cliente da experiência podem ser vistos na tela inicial, ilustrada na Figura 19. Para diferenciação, cada táxi emulado foi identificado com nomes de motorista e placas diferentes, como mostram as figuras 20 e 21.

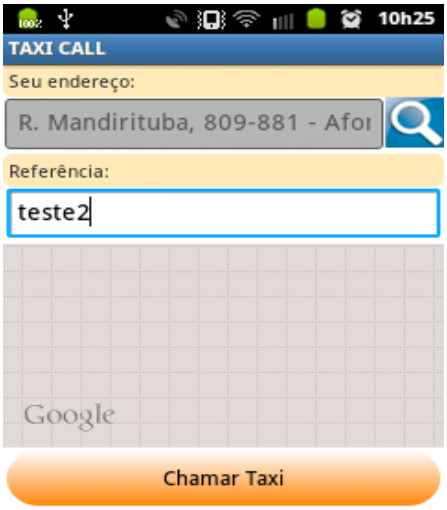


Figura 19 – Tela inicial do aplicativo do usuário (2013).
Fonte: Os autores

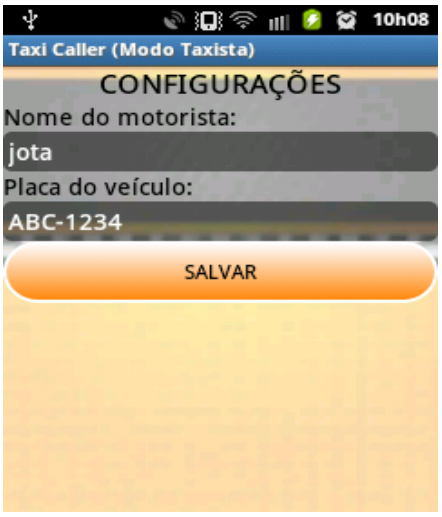


Figura 20 – Tela de configurações do aplicativo do primeiro táxi (2013).
Fonte: Os autores

Figura 21 – Tela de configurações do aplicativo do segundo táxi (2013).
Fonte: Os autores

Em seguida, o pedido foi enviado para a central, a qual o transferiu para o primeiro táxi, como é mostrado na figura 22 enquanto o segundo continua enviando as próprias coordenadas normalmente. Nota–se que mesmo após a recusa, a tela de aguardo do cliente continua executando sem nenhuma interferência, o que pode ser visto na figura 23. Pouco tempo após o primeiro táxi recusar o pedido enviado, o mesmo é recebido pelo segundo táxi, como exibido pela figura 24. E após aceitar este mesmo pedido, a informação é corretamente levada pela central ao aplicativo do cliente, como é provado pela figura 25.

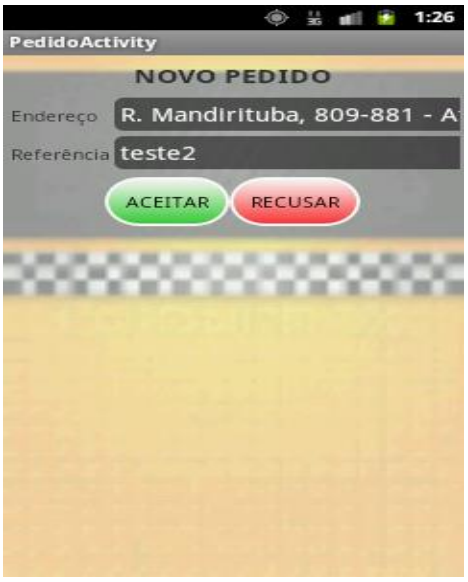


Figura 22 – Tela de novo pedido do aplicativo do primeiro táxi (2013).
Fonte: Os autores

Figura 23 – Tela de aguardo do aplicativo do usuário (2013).
Fonte: Os autores

Figura 24 – Tela de novo pedido do aplicativo do primeiro táxi (2013).
Fonte: Os autores

Figura 25 – Tela de pedido aceito do aplicativo do usuário (2013).
Fonte: Os autores

O processo poder ser observado também pelas mensagens emitidas pela central, que constam na figura 26. A partir dela é possível confirmar que o pedido foi recebido e encaminhado ao primeiro táxi, enquanto o segundo procede com o envio de posição. As mensagens também indicam que o pedido foi recusado, para então ser reenviado e direcionado ao segundo taxista, que o aceita.

```
Táxi mandou posição | Nome: Jota | Placa: ABC-1234
Táxi mandou posição | Nome: Joao | Placa: FGH-9988
Táxi mandou posição | Nome: Jota | Placa: ABC-1234
Pedido colocado na lista. Endereço:R. Mandirituba, 809-881
Calculando Táxi mais próximo...
Táxi encontrado: Táxi de Nome:Jota e Placa:ABC-1234
Táxi mandou posição | Nome: Joao | Placa: FGH-9988
Táxi mandou posição | Nome: Joao | Placa: FGH-9988
Táxi de placa ABC-1234 recusou um pedido
Pedido colocado na lista. Endereço:R. Mandirituba, 809-881
Calculando Táxi mais próximo...
Táxi encontrado: Táxi de Nome:Joao e Placa:FGH-9988
Táxi mandou posição | Nome: Jota | Placa: ABC-1234
Táxi aceitou pedido. Nome:Joao | Placa:FGH-9988
Táxi mandou posição | Nome: Jota | Placa: ABC-1234
Táxi mandou posição | Nome: Jota | Placa: ABC-1234
```

Figura 26 – Log do processo do sistema central (2013).
Fonte: Os autores

3.8.3 EXPERIÊNCIA 3 – DOIS CLIENTES PARA DOIS TÁXIS

Esta experiência teve como objetivo o acesso múltiplo ao servidor, em que dois clientes enviariam pedidos simultaneamente e seriam atendidos por dois táxis diferentes. Para isso foram utilizados três emuladores Android, sendo um com o aplicativo do lado cliente e dois com o lado taxista instalados. O segundo cliente foi simulado por um celular Android com o aplicativo instalado. Os dados de endereço e referência a serem enviados pelos clientes podem ser vistos nas figuras 27 e 28. O nome do motorista e a placa do carro de cada táxi podem ser visualizados nas figuras 29 e 30.

Figura 27 – Aplicativo do primeiro cliente – Tela inicial (2013).

Fonte: Os autores

Figura 28 – Aplicativo do segundo cliente – Tela inicial (2013).
Fonte: Os autores

Figura 29 – Aplicativo do primeiro táxi – Tela de configurações (2013).
Fonte: Os autores

Figura 30 – Aplicativo do segundo táxi – Tela de configurações (2013).
Fonte: Os autores

Ambos os pedidos foram enviados simultaneamente, entrando na tela de busca, já apresentada anteriormente. Após poucos segundos, ambos os aplicativos do lado taxista receberam os diferentes pedidos, como pode ser visto através das figuras 31 e 32.

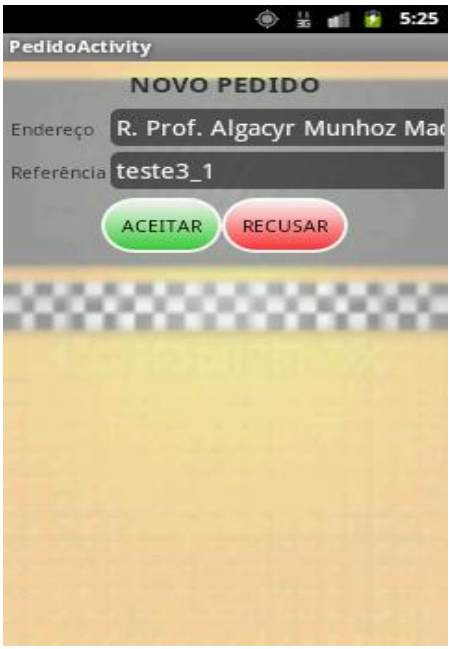


Figura 31 – Aplicativo do primeiro táxi – Tela de novo pedido (2013).
Fonte: Os autores

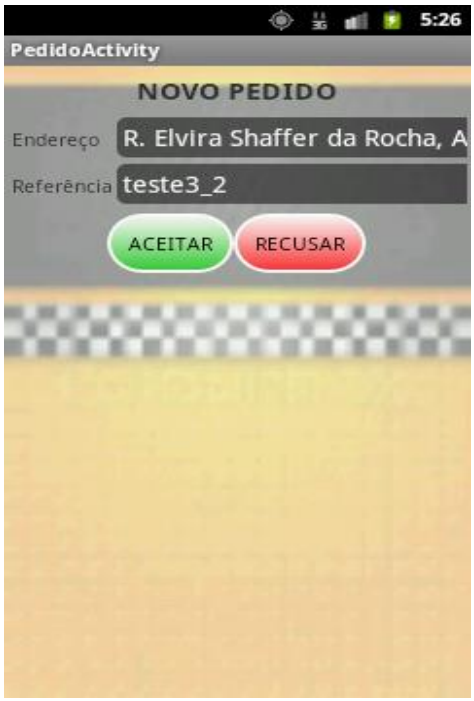


Figura 32 – Aplicativo do segundo táxi – Tela de novo pedido (2013).
Fonte: Os autores

Após receberem os pedidos, como esperado, cada taxista aceitou a corrida, enviando a confirmação para a central. Logo em seguida, cada aplicativo do lado cliente recebeu as informações do taxista que aceitou o pedido. Tais informações podem ser visualizadas nas figuras 33 e 34, que mostram as telas de cada aplicação no momento em que o pedido é aceito. E o processo completo é mostrado nas mensagens do sistema central, como é exibido na figura 35.

Figura 33 – Aplicativo do primeiro cliente – Tela de pedido aceito (2013).
Fonte: Os autores

Figura 34 – Aplicativo do segundo cliente – Tela de pedido aceito (2013).
Fonte: Os autores

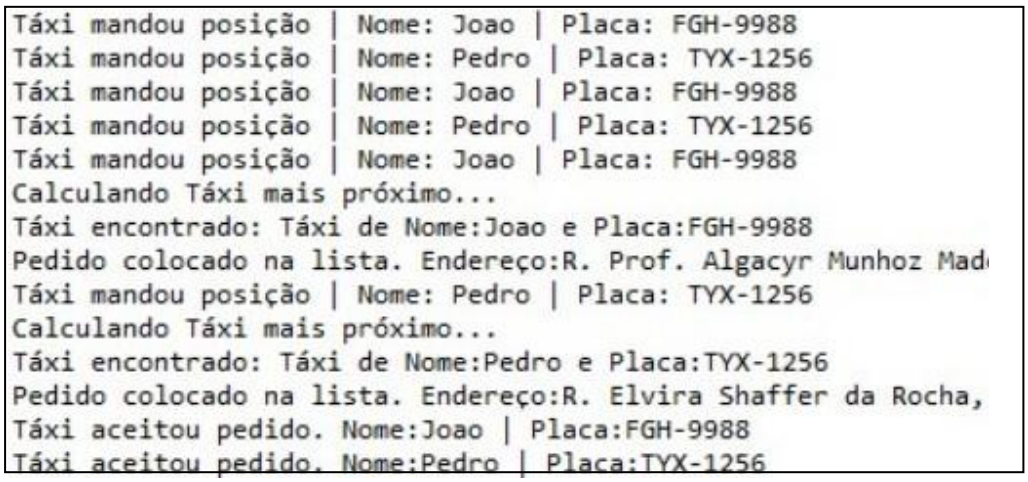


Figura 35 – Log do processo do sistema central (2013).
Fonte: Os autores

3.8.4 EXPERIÊNCIA 4 – DOIS CLIENTES E DOIS TÁXIS COM RECUSA PEDIDO

O objetivo desta experiência foi o de comprovar que, após recusar um pedido, o taxista ainda estará aberto para novas requisições de pedidos diferentes. Assim como na experiência anterior, foram utilizados dois emuladores Android que tivessem a aplicação do lado taxista instalada, e um terceiro emulador com o aplicativo do lado taxista instalado. Um celular Android também foi utilizado para a experiência, funcionando com o aplicativo do lado cliente.

Os dados de endereço e referência de ambos os pedidos enviados podem ser visualizados nas figuras 36 e 37, enquanto as configurações de cada taxista são mostradas nas figuras 38 e 39.

Figura 36 – Aplicativo do primeiro cliente – Tela inicial (2013).
Fonte: Os autores

Figura 37 – Aplicativo do segundo cliente – Tela inicial (2013).
Fonte: Os autores

Figura 38 – Aplicativo do primeiro táxi – Tela de configurações (2013).
Fonte: Os autores

Figura 39 – Aplicativo do segundo táxi – Tela de configurações (2013).
Fonte: Os autores

A experiência procedeu com o primeiro cliente enviando seu pedido primeiro, enquanto ambos os táxis enviaram continuamente suas posições à central. O

primeiro táxi recebeu o pedido, como é mostrado pela figura 40, mas o recusa, retornando à tela principal do aplicativo e enviando a própria posição normalmente.

Em seguida, o segundo táxi recebeu o pedido recusado, ao mesmo tempo em que o segundo pedido foi enviado. Poucos segundos após este envio, o primeiro táxi, que antes havia recusado um pedido, recebeu um novo pedido normalmente, como é exibido na figura 41. O processo completo também pode ser visualizado pelas mensagens emitidas pela central, apresentadas na figura 42.

Figura 40 – Aplicativo do primeiro táxi – Primeiro pedido (2013).
Fonte: Os autores

Figura 41 – Aplicativo do primeiro táxi – Segundo pedido (2013).
Fonte: Os autores

```
Táxi mandou posição | Nome: Jota | Placa: ABC-1234
Táxi mandou posição | Nome: Joao | Placa: FGH-9988
Táxi mandou posição | Nome: Jota | Placa: ABC-1234
Táxi mandou posição | Nome: Joao | Placa: FGH-9988
Táxi mandou posição | Nome: Jota | Placa: ABC-1234
Calculando Táxi mais próximo...
Táxi encontrado: Táxi de Nome:Jota e Placa:ABC-1234
Pedido colocado na lista. Endereço:Travessa Arcy Possebon, 2-168 - Afonso
Táxi mandou posição | Nome: Joao | Placa: FGH-9988
Táxi de placa ABC-1234 recusou um pedido
Calculando Táxi mais próximo...
Táxi encontrado: Táxi de Nome:Joao e Placa:FGH-9988
Pedido colocado na lista. Endereço:Travessa Arcy Possebon, 2-168 - Afonso
Táxi mandou posição | Nome: Jota | Placa: ABC-1234
Calculando Táxi mais próximo...
Táxi encontrado: Táxi de Nome:Jota e Placa:ABC-1234
Pedido colocado na lista. Endereço:R. Elvira Shaffer da Rocha, Afonso Pena
Táxi aceitou pedido. Nome:Jota | Placa:ABC-1234
Táxi aceitou pedido. Nome:Joao | Placa:FGH-9988
```

Figura 42 - Log do processo do sistema central (2013).
Fonte: Os autores

É possível notar, pelas mensagens, que o primeiro pedido foi de fato enviado e então recusado pelo primeiro táxi, que logo em seguida o recusou. O pedido foi enviado novamente, e então transferido para o segundo táxi. Simultaneamente, um novo pedido foi feito, sendo recebido pelo primeiro táxi. Ambos os taxistas aceitaram

seus pedidos, e tiveram seus dados transferidos com sucesso ao cliente respectivo. As confirmações dos pedidos podem ser vista nas figuras 43 e 44.

Figura 43 - Aplicativo do primeiro cliente – Tela de pedido aceito (2013).
Fonte: Os autores

Figura 44 - Aplicativo do segundo cliente – Tela de pedido aceito (2013).
Fonte: Os autores

3.8.5 EXPERIÊNCIA 5 – UM CLIENTE E NENHUM TÁXI DISPONÍVEL

Esta experiência teve o objetivo de realizar um teste de mensagem de erro, em que um cliente tentou realizar um pedido de táxi enquanto não havia nenhum táxi enviando posições para a central. Para isso foi utilizado um celular Android com o aplicativo do lado cliente instalado. As informações de endereço e referência do pedido podem ser visualizadas na figura 45. No momento em que o pedido é enviado, a central tenta buscar algum táxi em sua lista, mas por não encontrar nenhum, retorna um aviso para o aplicativo. A tela do cliente então exibe uma mensagem de aviso ao cliente, mostrada na figura 46.

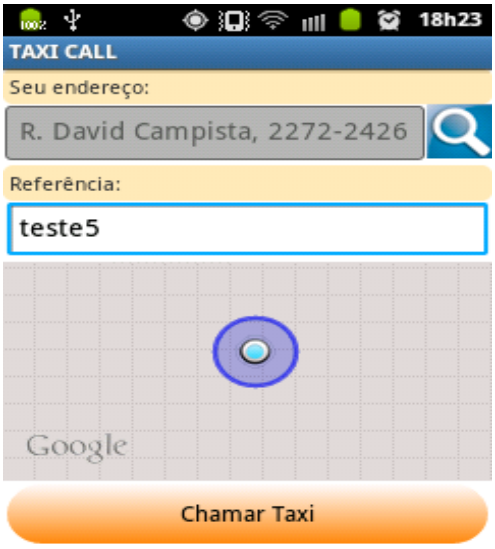


Figura 45 - Aplicativo do cliente – Tela inicial (2013).
Fonte: Os autores

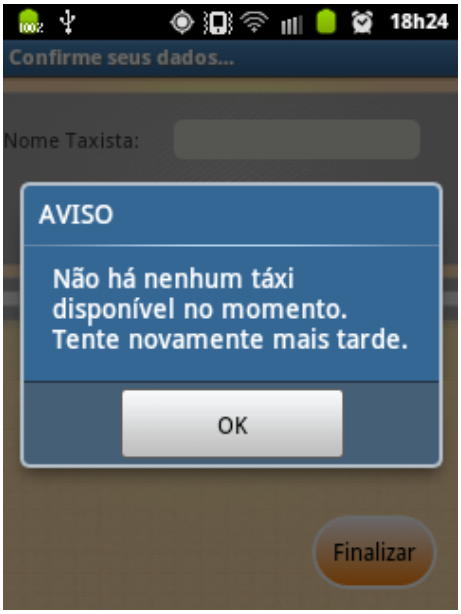


Figura 46 – Aplicativo do cliente – Mensagem de erro (2013).
Fonte: Os autores

3.8.6 EXPERIÊNCIA 6 – UM TAXISTA SEM GPS ATIVADO

Esta experiência teve o objetivo de realizar um teste de mensagem de erro, em que um taxista inicia o aplicativo em seu celular, sem antes ativar o mecanismo GPS do mesmo. Para isso foi utilizado um celular Android com o aplicativo do lado taxista instalado. A aplicação foi inicializada enquanto o GPS estava desativado. Assim que a tela inicial foi exibida, foi exibida a mensagem de erro indicando que o GPS deve ser ativado para que o aplicativo funcione como consta na figura 47.

Figura 47 – Aplicativo do taxista – Mensagem de erro (2013).
Fonte: Os autores

4 APRESENTAÇÃO DO SOFTWARE

Este capítulo descreve o funcionamento dos sistemas, com explicações e imagens ilustrando o funcionamento.

4.1 Instalação

Para o projeto foram desenvolvidas as seguintes aplicações:

- Aplicativo do cliente para Android – neste aplicativo o usuário tem sua localização reconhecida automaticamente pelo aplicativo e pode pedir um táxi com apenas um clique.
- Aplicativo do taxista para Android – neste aplicativo o taxista deve informar seu nome e a placa do táxi e se estiver disponível para corridas, deixar disponível o aplicativo para que dessa forma seja enviada sua posição e aguardando um novo pedido.
- Servidor Web central – o servidor recebe do aplicativo do cliente a solicitação de um pedido de táxi e também recebe do aplicativo do taxista a posição atual e armazena em uma lista. A partir desta lista é que o servidor localiza o táxi mais próximo do cliente e envia o pedido ao taxista. Caso o pedido seja aceito a resposta é enviada ao servidor, que envia para o cliente a confirmação do pedido.
- Estes sistemas encontram–se no CD anexo a este documento. Para executar os sistemas é necessário executar os arquivos .apk referente aos aplicativos em um dispositivo Android, sendo necessário executar em dispositivos diferentes. Para executar o servidor central será necessário montar um servidor de aplicação, adicionando todos arquivos disponíveis.

4.2 Funcionamento

O Taxi Caller é composto por 3 sistemas independentes que se comunicam via serviços Web. Estes sistemas são dois aplicativos móveis, um para o usuário do táxi e o outro aplicativo é para o taxista. O terceiro sistema é a central, responsável por gerenciar os pedidos.

O aplicativo para o usuário de táxi é composto pelas funções de obter a localização e enviar o pedido para a central. Ao iniciar o aplicativo é apresentada a tela principal, na qual é responsável por definir a localização e será exibido na nova tela um mapa, exibindo a atual posição do cliente, com um endereço exato, ou muito próximo. Este mapa é obtido a partir dos cálculos de latitude e longitude, funções específicas da plataforma Android, e que também serão armazenadas para serem enviadas à central. Definida a localização, o usuário poderá confirmar o endereço capturado pelo GPS. Também será possível informar um local de referência, informação que pode ajudar o taxista a encontrar seu cliente mais facilmente, conforme ilustra a figura 48.

Figura 48 – Tela inicial do aplicativo usuário (2013).
Fonte: Os autores.

Após receber a confirmação desses dados, a aplicação irá armazená–los e então enviá–los para o Sistema Central, via Webservice. O aplicativo deverá em seguida exibir ao cliente uma tela de espera, informando que o táxi mais próximo está sendo procurado, ilustrado pela figura 49.

Figura 49 – Tela de busca do aplicativo usuário (2013).
Fonte: Os autores.

Ao encontrar o táxi, e receber uma confirmação, a central responde ao aplicativo, e este apresenta uma nova tela ao cliente contendo os dados do taxista, incluindo o nome do motorista e a placa do táxi, ilustrado pela figura 50. A partir deste ponto, o aplicativo pode ser encerrado. Caso seja novamente executado, irá exibir a tela de início, para que o processo seja iniciado desde o começo.

Figura 50 – Tela de pedido aceito do aplicativo do usuário (2013).
Fonte: Os autores.

A central, um sistema desenvolvido em Java, é responsável por realizar a comunicação entre os dois aplicativos e principalmente encontrar o táxi mais próximo ao local do cliente. Este sistema não possui interface e permanece executando em segundo plano no servidor central.

Suponha que um cliente tenha enviando um pedido de táxi, o aplicativo então acessa o Sistema Central, via Webservice, iniciando um método que busca os táxis próximos, dentro de certo raio de busca. Em seguida, a central calcula qual destes táxis está mais próximo do cliente, a partir das coordenadas que recebeu de ambos. Localizado o veículo mais próximo, o pedido é repassado ao aplicativo do taxista, para que este decida se irá ou não aceitar a corrida. Caso não encontre nenhum taxista veiculando no raio definido, a central deve aumentar este valor de distância e reiniciar a busca. Em caso de recusa, o pedido será redirecionado ao próximo taxista na lista.

Após receber uma confirmação de que o pedido foi aceito pelo taxista, a central repassa a informação para o aplicativo do cliente, junto com os dados do taxista, ilustrado na figura 50. Estando essa corrida em andamento, qualquer outra busca daquele momento em diante deve desconsiderar o táxi que aceitou este pedido.

O Sistema Central foi desenvolvido com a capacidade de lidar com acessos múltiplos, o que foi atestado com a utilização de múltiplas threads para simular várias tentativas de acesso.

O aplicativo do taxista é baseado em dois estados diferentes: quando o veículo está fora de serviço ou quando está livre para atendimento, ilustrado na figura 51. Um taxista, ao iniciar a aplicação, tem sua posição atual enviada automaticamente para a central, se utilizando de métodos padrões da plataforma Android e do sistema GPS do celular. Enquanto o motorista estiver em circulação, livre para atendimento, o aplicativo continua enviando estes dados a cada intervalo de poucos segundos. Isso foi implementado em um método, que é executado continuamente. Ao mesmo tempo, é feita uma verificação na central se há algum pedido ou não.

Figura 51 – Tela principal do aplicativo taxista (2013).
Fonte: Os autores.

Se essa verificação retornar verdadeiro, a execução contínua do método para enviar as coordenadas do táxi deverá ser interrompida. Em seguida, é exibida uma tela com as informações do pedido, incluindo endereço em que se localiza o cliente e a referência informada pelo mesmo. A mesma tela exibe opções para o taxista decidir se aceita ou não a corrida, ilustrada na figura 52. Caso o pedido seja recusado, o aplicativo deve reiniciar a transmissão das coordenadas de localização. E caso a corrida seja aceita, a confirmação é enviada à central, para que seja transferida para o aplicativo do cliente e a localização do taxista para de ser enviada.

Figura 52 – Tela de solicitação de pedido do aplicativo taxista (2013).
Fonte: Os autores.

Outra função implementada para o aplicativo do taxista é a configuração de informações do taxista, ilustrado pela figura 53. Através de uma tela acessível a partir da principal, o usuário visualiza campos onde pode informar o próprio nome e a placa do táxi. Informações estas que são exibidas a qualquer cliente que tenha um pedido aceito. Esses dados são armazenados de maneira que o cadastro necessite ser feito apenas uma vez.

Figura 53 – Tela de configurações do aplicativo taxista (2013).
Fonte: Os autores.

Cada aplicativo do taxista diferente que envia continuamente a posição para a central é armazenado em uma lista de táxis. Quando um novo pedido é enviado para central, com as coordenadas, endereço e referência, a central irá verificar a lista de táxis e efetuar o cálculo da distância entre cada taxista e cliente. Este cálculo da distância é realizado pela fórmula de Haversine. Se a mesma calculada for menor que o raio determinado, o táxi é colocado em outra lista, guardando o valor desta. Após listar os táxis possíveis, um método de ordenação é executado colocando em ordem da menor para maior distância. Dessa forma o primeiro táxi desta lista recebe a solicitação do pedido. Caso o pedido seja recusado por esse taxista, a lista é refeita e o segundo taxista da lista recebe o pedido.

5 CONSIDERAÇÕES FINAIS

O objetivo principal do projeto Taxi Caller foi atingido, que era de desenvolver uma plataforma para automatização de pedidos de táxi, na qual fosse possível pedir um táxi e através dessa solicitação buscar o taxista mais próximo. Todo esse processo é gerenciado por um sistema central na nuvem.

Para isto, no projeto Taxi Caller foram implementadas três aplicações integradas, sendo dois aplicativos para smartfone. Um dos aplicativos permite ao usuário efetuar o pedido de táxi através de um botão. O outro aplicativo é para o taxista que envia sua posição e fica aguardando um novo pedido. O terceiro sistema, o servido central, recebe a posição enviada pelo aplicativo do taxista e armazena em uma lista. A partir desta lista é que o servidor central irá buscar o táxi mais próximo quando um cliente efetuar um pedido. Toda a comunicação entre as aplicações foram realizadas através de serviços Web (Web service).

O projeto foi concluído no tempo determinado seguindo o cronograma. Como as atividades estavam bem dividas entre os integrantes, houve momentos que foi necessário o auxílio da equipe a algum integrante com dificuldades, dessa forma não houve atrasos no cronograma planejado.

A maior dificuldade durante o projeto foi em desenvolver os aplicativos e a integração entre os sistemas com Web services (serviços Web), pois a equipe ainda não tinha o sólido conhecimento, porém com pesquisas e muitos exercícios a equipe conseguiu realizar e desenvolver o projeto.

Aplicações semelhantes ao Taxi Caller, como o Easy Taxi, apresentado anteriormente, possuem uma funcionalidade específica que permite ao usuário acompanhar em tempo real a movimentação do táxi que aceitou seu pedido. Isso oferece um grau maior de confiança ao usuário, uma vez que este tem a certeza de que há um taxista em sua direção. Essa é uma possível funcionalidade a ser adicionada ao projeto do Taxi Caller.

Mesmo entre os aplicativos semelhantes encontrados durante a pesquisa, foi notada a falta de uma funcionalidade para pagamento realizado pelo próprio celular,

embora alguns dos trabalhos citados possuam essa opção como projeto futuro. Esta é outra possibilidade a ser levada em conta para o Taxi Caller, podendo até mesmo se tornar um diferencial do aplicativo em relação a seus semelhantes.

REFERÊNCIAS

ACUNETIX: **Web Services – The Technology and Its Security Concerns.** Disponível em: www.acunetix.com/websitesecurity/web–services–wp/. Acessado em: 18/01/2013.

ARAÚJO, A. F. de: **Processamento dirigido de rotas através de texto–fala.** Disponível em http://campeche.inf.furb.br/tccs/2012–I/TCC2012–7–18–VF–AdrianoFdeAraujo.pdf. Acessado em: 11/02/2013.

AZIZ, A.: MITCHELL, Scott**: An Introduction to JavaScript Object Notation (JSON) in JavaScript and .NET.** Disponivel em: http://msdn.microsoft.com/en–us/library/bb299886.aspx#intro_to_json_topic1. Acessado em: 25/01/2013.

BEAL, V.: **Understanding Web Services.** Disponível em: www.webopedia.com/DidYouKnow/Computer_Science/2005/web_services.asp. Acessado em: 18/01/2013.

CROCKFORD, D.: **RFC 4627.** Disponível em: http://tools.ietf.org/html/rfc4627. Acessado em: 26/01/2013.

DANA, P. H.: **The Geographer's Craft Project.** Disponível em: http://www.colorado.edu/geography/gcraft/notes/gps/gps_f.html. Acessado em: 16/01/2013.

DEVELOPERS: **Platform Versions.** Disponível em: developer.android.com/about/dashboards/index.html. Acessado em: 16/01/2013.

DEVELOPERS: **Publish Checklist for Google Play.** Disponível em: developer.android.com/distribute/googleplay/publish/preparing.html. Acessado em: 16/01/2013.

GARMIN: **What is GPS?** Disponível em: http://www8.garmin.com/aboutGPS/. Acessado em: 16/01/2013.

FINKELSTEIN, C.: **Web Services Evolution.** Disponível em: www.tdan.com/view–articles/5085. Acessado em: 17/01/2013.

KUEHL, C. A.: **Protótipo de sistema móvel na plataforma Android para compartilhamento de arquivos e mensagens entre dispositivos baseado em proximidade geográfica.** Disponível em: http://campeche.inf.furb.br/tccs/2012–I/TCC2012–1–09–VF–CesarAKuehl.pdf. Acessado em 10/02/2013

LECHETA, R. R. Google Android: **Aprenda a criar aplicações para dispositivos móveis.** 2ª edição. São Paulo: Novatec, 2010.

LIMA e SILVA, P.: **Mais de 69% da população brasileira têm celular.** Disponível em: www.veja.abril.com.br/notícia/Brasil/pnad–2011–mais–de–69–da–populacao–brasileira–tem–celular. Acessado em: 12/01/2013.

MELLO JR, J. P.: **Android devices in U.S. face more malware attacks than PCs.** Disponível em: www.pcworld.com/article/2018388/android–devices–in–u–s–face–more–malware–attacks–than–pcs.html. Acessado em 14/01/2013.

GOOGLE MOBILE: **Google Maps para telemóvel.** Disponível em: http://www.google.com/mobile/maps/. Acessado em: 16/01/2013

ORACLE: **Introduction to RESTful Web Services and Jersey.** Disponível em: http://docs.oracle.com/cd/E19776–01/820–4867/ggnyk/index.html. Acessado em: 20/01/2013.

RODRIGUEZ, S.: **RESTful Web Services: The basics.** Disponível em: http://www.ibm.com/developerworks/webservices/library/ws–restful/. Acessado em: 20/01/2013.

SILVA, L. P. da: **BHC Móvel.** Disponível em: http://code.google.com/intl/pt/files/TCC%20–%20Final.pdf. Acessado em: 11/02/2013.

SPIES, B.: **Web Services, Part 1: SOAP vs. REST.** Disponível em: http://www.ajaxonomy.com/2008/xml/web–services–part–1–soap–vs–rest. Acessado em 19/01/2013.

TYAGI, S.: **RESTful Web Services.** Disponível em: http://www.oracle.com/technetwork/articles/javase/index–137171.html. Acessado em: 20/01/2013.

WS3SCHOOLS: **WSDL and UDDI.** Disponivel em: www.w3schools.com/wsdl/wsdl_uddi.asp. Acessado em: 18/01/2013.

GAMA, A. JSON Simples e Prático, Parte I. **K19 Artigos**, 22 dezembro 2011. Sessão Artigos. Disponível em: http://www.k19.com.br/artigos/json–simples–e–pratico–parte–i/. Acessado em: 20/01/2013.

Schek , A. **Lançamento oficial do G1 T–Mobile com HTC e Google**. Disponível em :http://www.fayerwayer.com.br/2008/09/lancamento–oficial–do–g1–t–mobile–com–htc–e–google/. Acessado em 12/01/2013.

Kaswan. **REST API– An Introduction On Rest Web Services.** Disponível em: http://crazylearner.com/rest–api–an–introduction–on–rest–web–services/. Acessado em 20/01/2013.

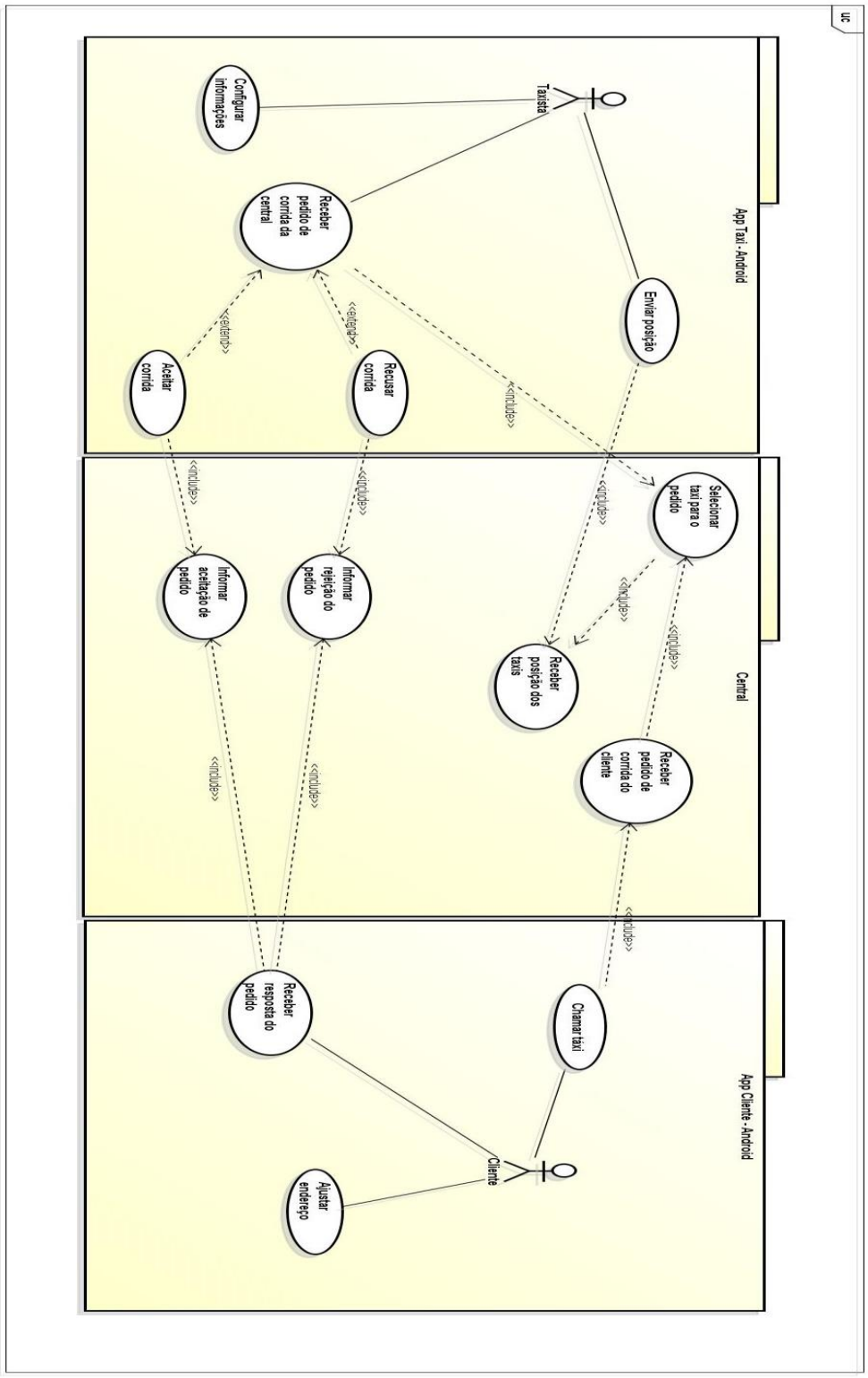
APÊNDICES

APÊNDICE A – PLANO DE RISCOS

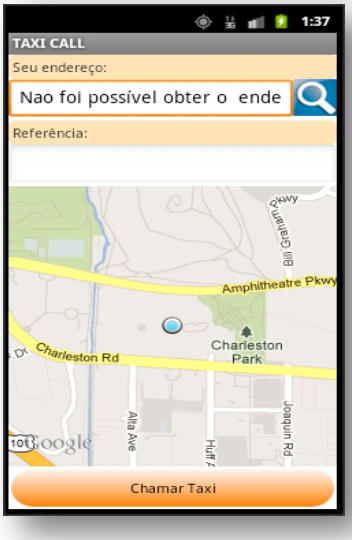
Condição	Consequência	Ação	Probabilidade	Impacto	Classi–ficação
Pouco conhecimento com a plataforma Android.	Não desenvolver os aplicativos	Realizar estudos e fazer exercícios.	Moderado	Moderado	4
Dificuldade com a criação dos Web Services	Os aplicativos não se comunicarão com o sistema central	Realizar estudos e fazer exercícios.	Moderado	Alto	6
Problemas de comunicação	Não execução de tarefas definidas	Reuniões e alinhamentos das atividades.	Moderado	Moderado	5
Falta de precisão nos diagramas e erros de análise	Ocorrer erros no desenvolvimento e recomeçar esta etapa.	Elaboração de protótipos para fazer avaliação da análise	Moderado	Moderado	5
Não cumprimento dos prazos	Atraso nas entregas	E–mails e reuniões para cobrar as tarefas	Moderado	Moderado	5

Tabela 4 – Tabela Plano de riscos

APÊNDICE B – DIAGRAMA DE CASOS DE USO



APÊNDICE C - Especificação dos casos de uso

ID	UC001
Nome	Chamar táxi
Descrição	Este caso de uso serve para que um cliente possa requisitar um pedido de táxi, enviando o próprio endereço, junto com uma referência para a central.
Data View	
DV1 – Tela Splash	
DV2 - Tela principal	

DV3 - Tela buscando táxi	
DV4 - Tela táxi encontrado	
Pré condições	Este caso de uso pode iniciar somente se: 1. O cliente tiver o aplicativo instalado em seu celular 2. O celular do cliente tiver função GPS 3. A função GPS do celular do cliente estiver habilitada ou o celular conectado na internet.
Pós condições	Ao fim normal deste caso de uso o aplicativo deve: 1. Exibir a tela DV4 com os dados do taxista a caminho.
Ator primário	Cliente
Fluxo de Eventos principal	
1.	O ator inicia o aplicativo;
2.	O aplicativo exibe a tela DV1;
3.	O aplicativo exibe a tela DV2;
4.	O ator preenche o campo "Referência" e confirma o seu endereço; (A1)
5.	O ator clica em "Chamar táxi";
6.	O aplicativo captura os dados de localização do cliente e envia à central; (E1) (RN2)
7.	O aplicativo exibe a tela DV3 com um diálogo de progresso da busca de um táxi; (A2)
8.	O aplicativo encontra um táxi e mostra os dados do taxista, conforme a tela DV4; (RN1)
9.	O cliente finaliza o aplicativo

Fluxos Alternativos	
A1:	O endereço encontrado pelo gps não é preciso e o cliente deseja atualizar.
	1. O caso de uso UC002 – Ajustar endereço é iniciado.
A2:	Tempo para encontrar um taxi atinge 1 minuto.
	1. O aplicativo exibe uma mensagem. (M1)
	2. O caso de uso é reiniciado.
Fluxos de Exceção	
E1:	Posição não foi capturada pelo GPS corretamente.
	1. O aplicativo exibe mensagem de erro. (M2)
	2. O caso de uso é reiniciado.
Mensagens	
M1:	Táxi não encontrado.
M2:	Ocorreu um erro de locallização.
Regras de Negócio	
RN1:	O aplicativo retornará um táxi, caso a distância entre o cliente e o táxi esteja em um raio mínimo de alcance igual a 3000m.
RN2:	Para execução correta do aplicativo é necessário que a Central do sistema esteja habilitada e funcional.

Tabela 5 - Caso de uso Chamar Táxi

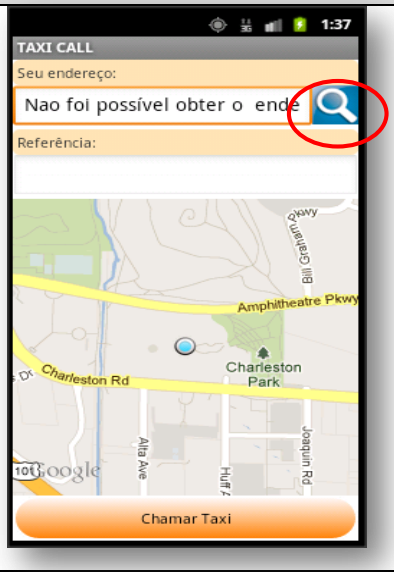
ID	UC002
Nome	Ajustar endereço
Descrição	Este caso de uso serve para que um cliente possa ajustar o endereço encontrado pelo gps do celular.
Data View	
DV1 - Tela ajuste do endereço	
Pré condições	Este caso de uso pode iniciar somente se: 1. O ator quiser ajustar o endereço e clicar no botão apropriado.
Pós condições	Ao fim normal deste caso de uso o aplicativo deve: 1. Atualizar o campo do endereço, conforme indica a imagem em DV1.
Ator primário	Cliente
Fluxo de Eventos principal	
1.	O ator observa que o endereço não está correto.
2.	O ator clica no botão da lupa
3.	O sistema atualiza as informações de endereço, reprocessando as coordenadas. (E1) (E2)
4.	O sistema atualiza o campo endereço da tela DV1.
Fluxos Alternativos	
Não se aplica.	
Fluxos de Exceção	
E1:	As coordenadas não foram capturadas pelo GPS corretamente.
	1. O aplicativo exibe mensagem de erro no campo endereço. (M1)
	2. O caso de uso é finalizado.
E2:	O sistema não pode converter as coordenadas em um nome de endereço.
	1. O aplicativo exibe mensagem de erro no campo endereço. (M1)
	2. O caso de uso é finalizado.
Mensagens	
M1:	Não foi possível obter o endereço.
Regras de Negócio	
RN1:	O aplicativo retornará um táxi, caso a distância entre o cliente e o táxi esteja em um raio mínimo de alcance igual a 3000m.

Tabela 6 - Caso de uso Ajustar Endereço

ID	UC003
Nome	Receber resposta do pedido
Descrição	Este caso de uso serve para que o cliente visualize as informações do taxista que confirmou o seu pedido.
Data View	
DV1 - Tela ajuste do endereço	
Pré condições	Este caso de uso pode iniciar somente se: 1. O caso de uso UC001 – Chamar táxi foi executado; 2. Esse pedido deve ter sido processado na central e um taxista confirmado que aceita o pedido.
Pós condições	Ao fim normal deste caso de uso o aplicativo deve: 1. Atualizar os campos com os dados do taxista que confirmou o seu pedido na tela DV1.
Ator primário	Cliente
Fluxo de Eventos principal	
1.	O ator requisitou um pedido de táxi através do caso de uso UC001 – Chamar taxi;
2.	O ator tem o pedido aceito por um taxista através do caso de uso UC005 – Aceitar corrida
3.	Central transfere a confirmação para o aplicativo do cliente através do caso de uso “UC007 – Informar aceitação de pedido”
4.	Aplicativo do cliente exibe tela DV1 com Mensagem apropriada. (A1) (E1) (M1) (RN1)
5.	O caso de uso é encerrado
Fluxos Alternativos	
A1:	Botão "Finalizar" é pressionado.
	1. O aplicativo é encerrado.
Fluxos de Exceção	
E1:	O aplicativo não retorna nenhum dado do taxista.
	1. O aplicativo exibe mensagem de erro no campo endereço. (M1)
	2. O caso de uso é finalizado.
Mensagens	
M1:	Não foi encontrado um táxi.
Regras de Negócio	
RN1:	O aplicativo retornará um táxi, caso o limite de tempo da busca não seja atingido. O tempo é igual a 1 minuto.

Tabela 7 - Caso de uso Receber resposta do pedido

ID	UC004
Nome	Configurar informações
Descrição	Este caso de uso serve para que o taxista possa armazenar as informações que serão enviadas ao cliente após aceitar um pedido de corrida.
Data View	
DV1 - Tela principal taxi	
DV2 - Tela configura-ções taxi	
Pré condições	Este caso de uso pode iniciar somente se: 1. O taxista usar o aplicativo pela 1° vez ou deseja modificar as suas configurações.
Pós condições	Ao fim normal deste caso de uso o aplicativo deve: 1. O aplicativo deve retornar a tela principal do aplicativo. (DV1)
Ator primário	Taxista
Fluxo de Eventos principal	
1.	O ator inicia o aplicativo;
2.	Aplicativo exibe tela DV1 (A1);
3.	O ator clica em Menu -> Configurações;
4.	Aplicativo exibe a tela DV2;
5.	O ator insere ou altera informações nos campos do formulário;
6.	O ator pressiona botão “Salvar”;
7.	Aplicativo salva as informações;

8.	Aplicativo retorna para a tela DV1;
9.	O caso de uso é encerrado.
Fluxos Alternativos	
A1:	Caso for o primeiro acesso:
	1. O aplicativo mostra diálogo para o taxista (M1) direciona para a tela DV2.
Fluxos de Exceção	
Não se aplica.	
Mensagens	
M1:	Você ainda não configurou o seu táxi.
Regras de Negócio	
Não se aplica.	

Tabela 8 - Caso de uso Configurar informações

ID	UC005
Nome	Enviar posição
Descrição	Este caso de uso serve para que um taxista envie a própria posição por GPS, para que a central possa localizá-lo e assim enviar um pedidos de cliente.
Data View	
DV1 - Tela principal taxi	
Pré condições	Este caso de uso pode iniciar somente se: 1. O celular do taxista possuir função GPS 2. A função GPS estiver ativada ou conectado na internet.
Pós condições	Não se aplica.
Ator primário	Taxista
Fluxo de Eventos principal	
1.	O ator inicia o aplicativo;
2.	O aplicativo exibe a tela DV1; (E1) (RN1)
3.	O aplicativo automaticamente envia a coordenada atual pelo GPS para a central;
4.	O aplicativo verifica se há algum pedido enviado (A1)
5.	O caso de uso reinicia a partir no passo 2;
Fluxos Alternativos	
A1:	Taxista recebe um pedido:
	1. O caso de uso é interrompido
	2. Inicia-se o caso de uso “UC005 – Receber pedido de corrida”
Fluxos de Exceção	
E1:	O sistema não inicializa corretamente:
	1. O caso de uso é interrompido
	2. O aplicativo é encerrado
Mensagens	
M1:	Você ainda não configurou o seu táxi.
Regras de Negócio	
RN1:	Para inicialização correta do aplicativo é necessário que a Central do sistema esteja habilitada e funcional.

Tabela 9 - Caso de uso Enviar posição

ID	UC006
Nome	Receber pedido de corrida da central
Descrição	Este caso de uso serve para o taxista receber um novo pedido de cliente, após ter sido processado na central.
Data View	
DV1 - Tela de novo pedido	
Pré condições	Este caso de uso pode iniciar somente se: 1. Um cliente requisitou um táxi e o caso de uso UC001 – Chamar táxi foi executado.
Pós condições	Após o término desse caso de uso, o sistema deve: 1. Enviar o pedido com as informações do cliente para o taxista escolhido
Ator primário	Taxista
Ator secundário	Cliente
Fluxo de Eventos principal	
1.	Ator secundário envia pedido de táxi através do caso de uso “UC001 - Chamar táxi”
2.	A central recebe o pedido do cliente, com a localização do mesmo. (RN1)
3.	A central busca o táxi mais próximo a partir dessa localização (E1) (RN2)
4.	A central envia os dados desse pedido para o aplicativo do taxista encontrado. (DV1)
5.	O ator primário recebe uma notificação de pedido e faz a confirmação, conforme as possibilidades em DV1. (A1) (A2)
6.	O caso de uso é encerrado.
Fluxos Alternativos	
A1:	Taxista clica em "Aceitar pedido"
	1. O caso de uso UC007 - Aceitar pedido de corrida é executado;
A2:	Taxista clica em "Recusar pedido"
	1. O caso de uso UC008 - Recusar pedido de corrida é executado;
Fluxos de Exceção	
E1:	Táxi não encontrado
	1. O sistema aumenta o raio de alcance da busca;
	2. O caso de uso se reinicia a partir do passo 3;
Mensagens	
Não se aplica	
Regras de Negócio	

RN1:	Para inicialização correta do aplicativo é necessário que a Central do sistema esteja habilitada e funcional.
RN2:	O sistema calcula a distância entre o cliente e o táxi utilizando a Fórmula de Haversine, que fornece distâncias entre dois pontos de uma esfera a partir de suas latitudes e longitudes: $$= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$$

Tabela 10 - Caso de uso Receber pedido de corrida da central

ID	UC007
Nome	Aceitar pedido de corrida
Descrição	Caso de uso em que o taxista aceita a requisição de um pedido e envia a confirmação que será transferida para o aplicativo do cliente.
Data View	
DV1 - Tela pedido aceito táxi	
Pré condições	Este caso de uso pode iniciar somente se: 1. O caso de uso “UC006 - Receber pedido de corrida” foi iniciado.
Pós condições	Após o término desse caso de uso, o sistema deve: 1. Retornar a tela DV1, informando que o taxi está em serviço.
Ator primário	Taxista
Fluxo de Eventos principal	
1.	O ator pressiona botão “Aceitar”
2.	Sistema envia dados do taxista com a confirmação à central; (RN1)
3.	O sistema retorna a tela principal do taxi mostrando uma mensagem apropriada. (DV1) (M1)
4.	O caso de uso é encerrado.
Fluxos Alternativos	
Não se aplica	
Fluxos de Exceção	
Não se aplica	
Mensagens	
M1:	Em serviço
Regras de Negócio	
RN1:	Para inicialização correta do aplicativo é necessário que a Central do sistema esteja habilitada e funcional.

Tabela 11 - Caso de uso Aceitar pedido de corrida

ID	UC008
Nome	Recusar pedido de corrida
Descrição	Caso de uso em que o taxista recusa o pedido de um cliente, e a central possa enviar o pedido ao próximo táxi .
Data View	
DV1 - Tela pedido aceito táxi	
Pré condições	Este caso de uso pode iniciar somente se: 1. O caso de uso “UC006 - Receber pedido de corrida” foi iniciado.
Pós condições	Após o término desse caso de uso, o sistema deve: 1. Retornar a tela DV1, informando que o taxi está livre.
Ator primário	Taxista
Fluxo de Eventos principal	
1.	O ator pressiona botão "Recusar"
2.	Sistema envia dados do taxista com confirmação à central; (RN1)
3.	O sistema retorna a tela principal do táxi mostrando uma mensagem apropriada. (DV1) (M1)
4.	A central envia o pedido para o próximo táxi ;
5.	O caso de uso é encerrado.
Fluxos Alternativos	
Não se aplica	
Fluxos de Exceção	
Não se aplica	
Mensagens	
M1:	Livre
Regras de Negócio	
RN1:	Para inicialização correta do aplicativo é necessário que a Central do sistema esteja habilitada e funcional.

Tabela 12 - Caso de uso Recusar pedido de corrida

ID	UC009
Nome	Selecionar taxi para o pedido
Descrição	Este caso de uso serve para que a central processe o táxi mais adequado para um determinado pedido de corrida recebido.
Data View	
Não se aplica	
Pré condições	Este caso de uso pode iniciar somente se: 1. Um cliente requisitou um táxi e o caso de uso UC001 foi executado.
Pós condições	Após o término desse caso de uso, o sistema deve: 1. Enviar o pedido com as informações do cliente para o taxista escolhido
Ator primário	Cliente
Ator primário	Taxista
Fluxo de Eventos principal	
1.	O cliente envia um pedido de corrida para a central e o caso de uso UC001 é executado;
2.	A central recebe o pedido do cliente, com a localização do mesmo. (RN1)
3.	A central busca o táxi mais próximo a partir dessa localização (E1) (RN2)
4.	A central envia os dados desse pedido para o aplicativo do taxista encontrado.
5.	O caso de uso é encerrado.
Fluxos Alternativos	
Não se aplica	
Fluxos de Exceção	
E1:	Táxi não encontrado
	1. O sistema aumenta o raio de alcance da busca;
	2. O caso de uso se reinicia a partir do passo 3;
Mensagens	
Não se aplica	
Regras de Negócio	
RN1:	Para inicialização correta do aplicativo é necessário que a Central do sistema esteja habilitada e funcional.
RN2:	O sistema calcula a distância entre o cliente e o táxi utilizando a Formula de Haversine, que fornece distâncias entre dois pontos de uma esfera a partir de suas latitudes e longitudes: $= 2r \arcsin\left(\sqrt{\sin^2\left(\frac{\phi_2 - \phi_1}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\lambda_2 - \lambda_1}{2}\right)}\right)$

Tabela 13 - Selecionar taxi para o pedido

ID	UC010
Nome	Receber posição dos táxis
Descrição	Este caso de uso serve para que a central receba os táxis que estão enviando as suas respectivas posições atuais e armazená-los em uma lista de táxis disponíveis.
Data View	
Não se aplica	
Pré condições	Este caso de uso pode iniciar somente se: 1. Servidor da Central estiver habilitado e um táxi enviar a sua posição.
Pós condições	Não se aplica.
Ator primário	Taxista
Fluxo de Eventos principal	
1.	O taxista inicia o seu aplicativo e envia uma posição;
2.	A central recebe a posição do táxi e armazena em uma lista específica; (RN1)
3.	O caso de uso é executado continuamente. (RN2)
Fluxos Alternativos	
Não se aplica	
Fluxos de Exceção	
E1:	Táxi não encontrado
	1. O sistema aumenta o raio de alcance da busca;
	2. O caso de uso se reinicia a partir do passo 3;
Mensagens	
Não se aplica	
Regras de Negócio	
RN1:	Para inicialização correta do aplicativo é necessário que a Central do sistema esteja habilitada e funcional.
RN2:	Enquanto houver envio de posições dos táxis.

Tabela 14 - Receber posição dos táxis

ID	UC011
Nome	Receber pedido de corrida do cliente
Descrição	Este caso de uso serve para a central receber um novo pedido de cliente e processá-lo afim de encontrar um táxi mais próximo.
Data View	
DV1 - Tela aguardando resposta	
Pré condições	Este caso de uso pode iniciar somente se: 1. Um cliente requisitou um táxi e o caso de uso UC001 – Chamar táxi foi executado.
Pós condições	Após o término desse caso de uso, o sistema deve: 1. O caso de uso UC006 - Selecionar táxi para o pedido deve ser iniciado.
Ator primário	Cliente
Fluxo de Eventos principal	
1.	O cliente envia pedido de táxi através do caso de uso “UC001 - Chamar táxi”
2.	A central recebe o pedido do cliente, com a localização do mesmo. (RN1)
3.	A central executa o caso de uso UC009 - Selecionar táxi para o pedido
4.	O cliente visualiza a tela DV1, após central iniciar o processamento. (E1)
6.	O caso de uso é encerrado.
Fluxos Alternativos	
Não se aplica	
Fluxos de Exceção	
E1:	O cliente não visualiza a tela DV1.
	1. O caso de uso UC001 é reiniciado.
Mensagens	
Não se aplica	
Regras de Negócio	
RN1:	Para inicialização correta do aplicativo é necessário que a Central do sistema esteja habilitada e funcional.

Tabela 15 - Receber pedido de corrida do cliente

APÊNDICE D – DIAGRAMA DE CLASSES

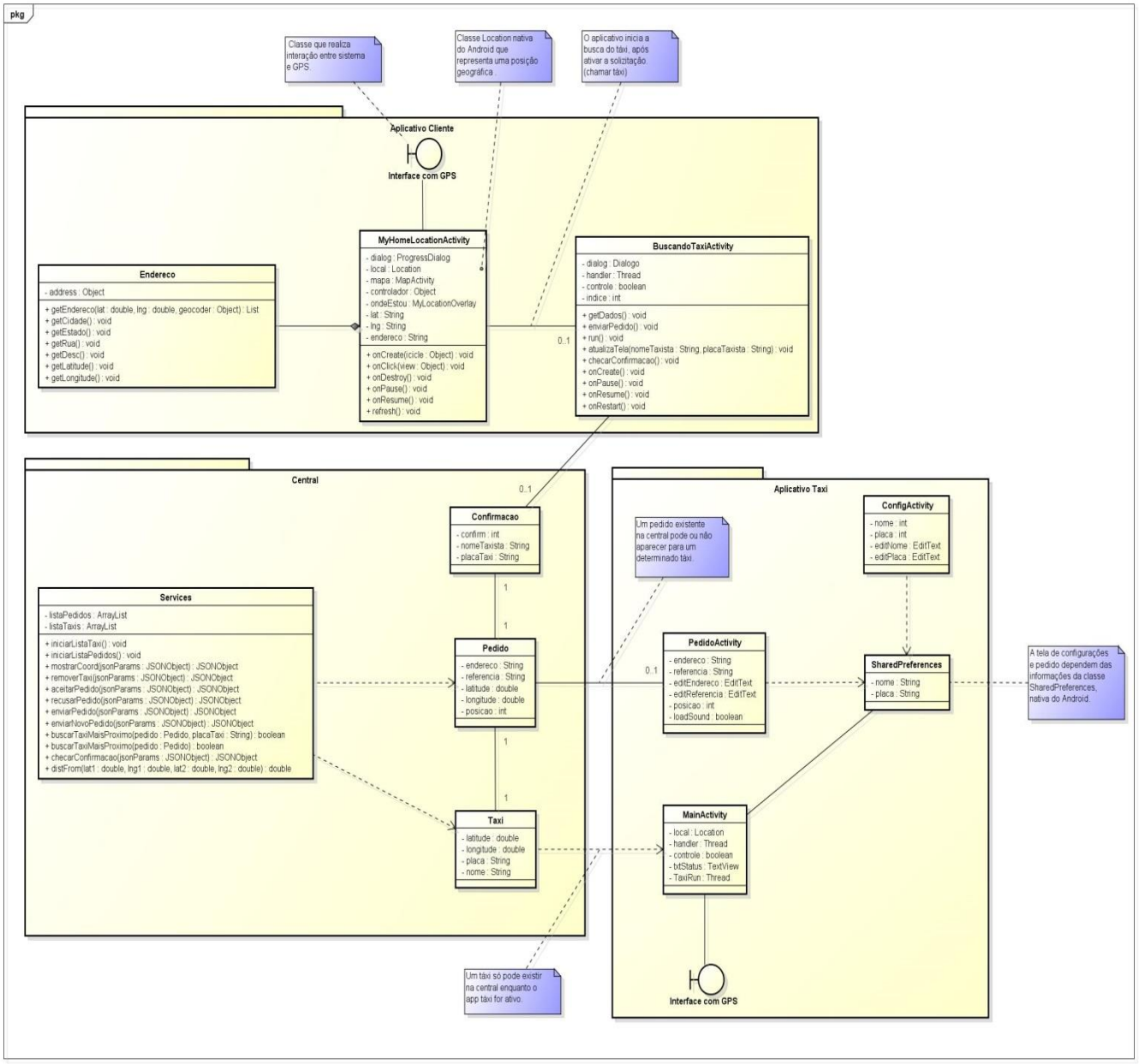


Figura 55 - Diagrama de classes

APÊNDICE E – DIAGRAMA DE SEQUÊNCIA

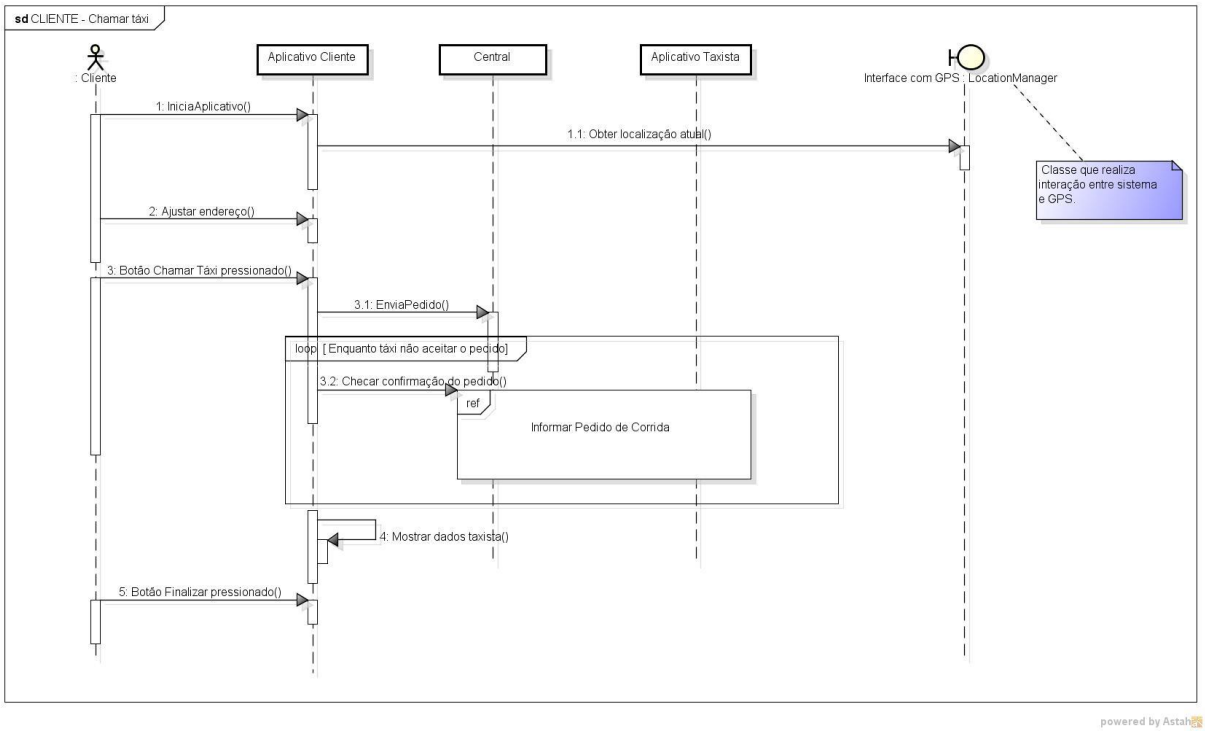


Figura 56 - Diagrama de sequência Chamar táxi

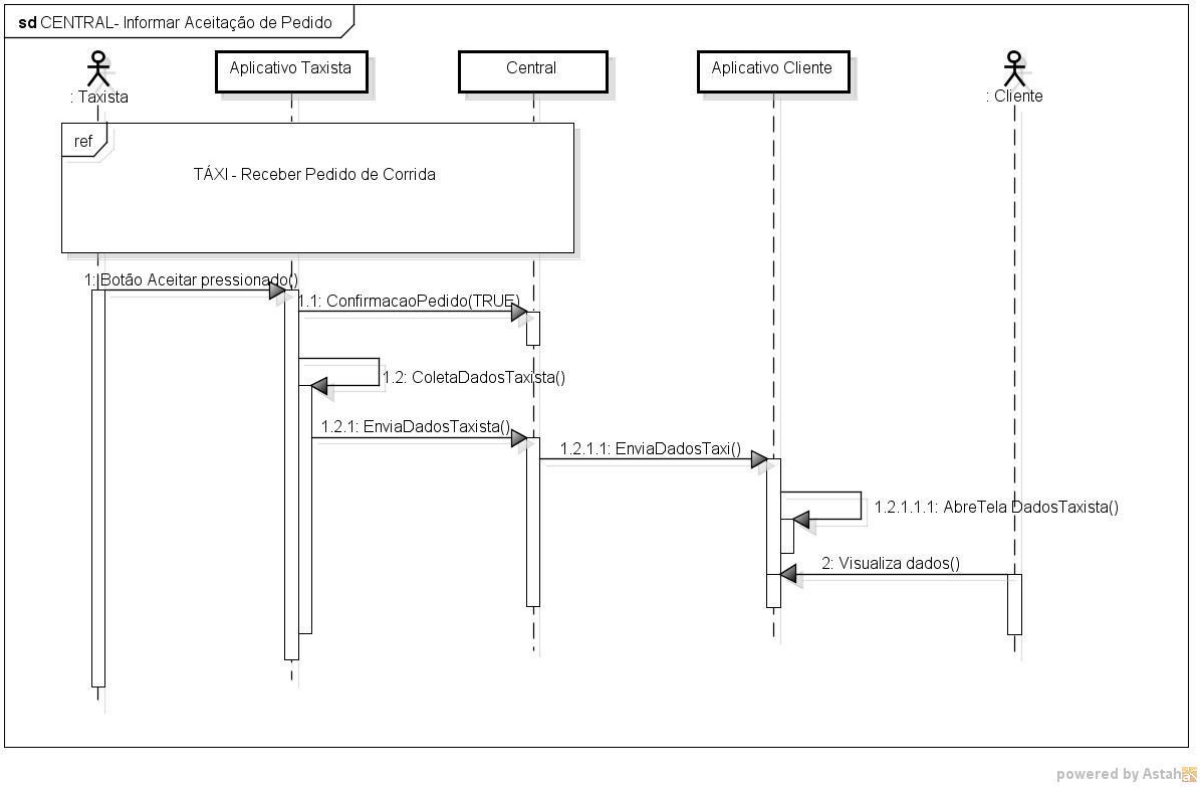


Figura 57 - Diagrama de sequência Informar aceitação de pedido

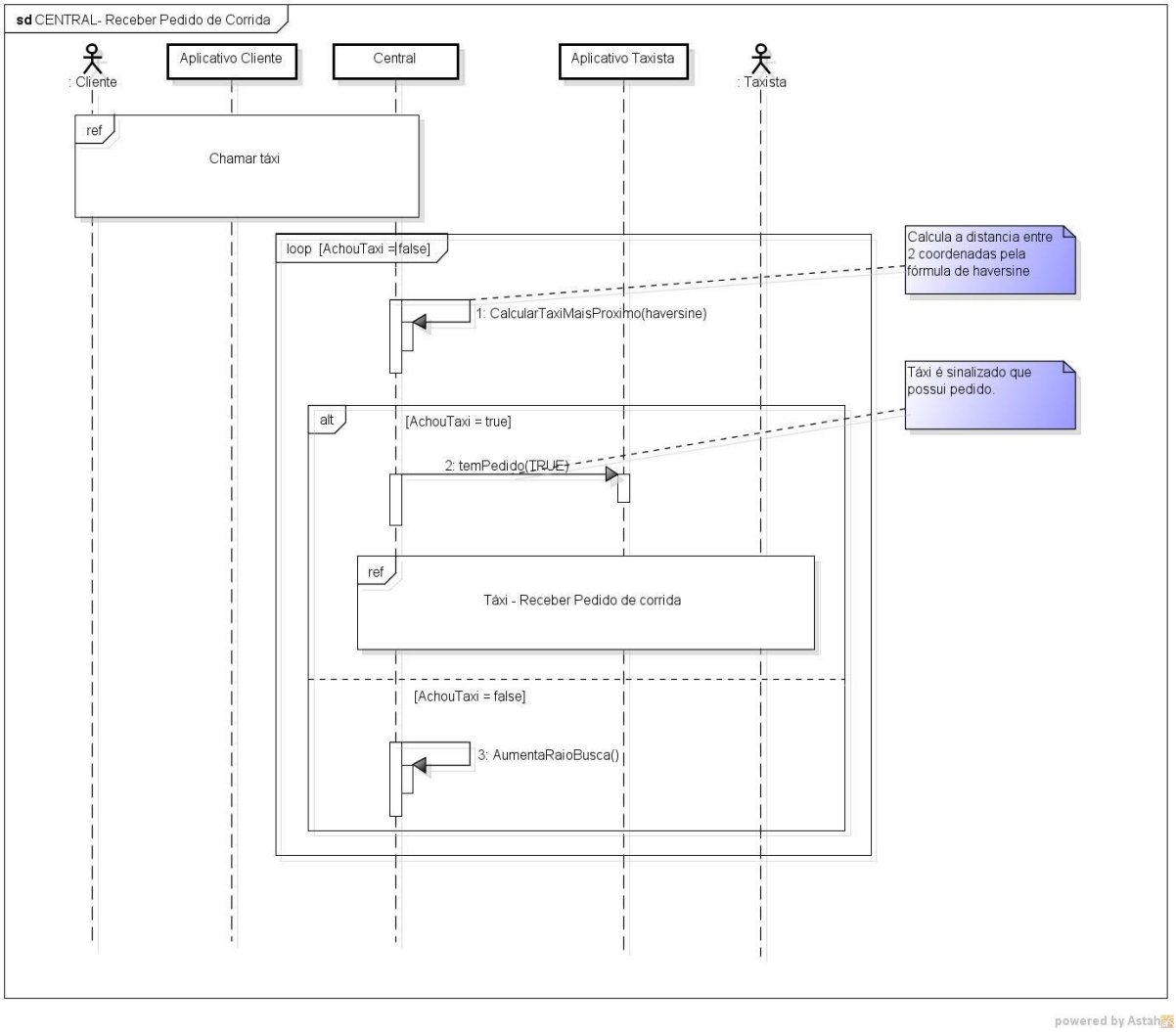


Figura 58 - Diagrama de sequência Receber pedido de corrida

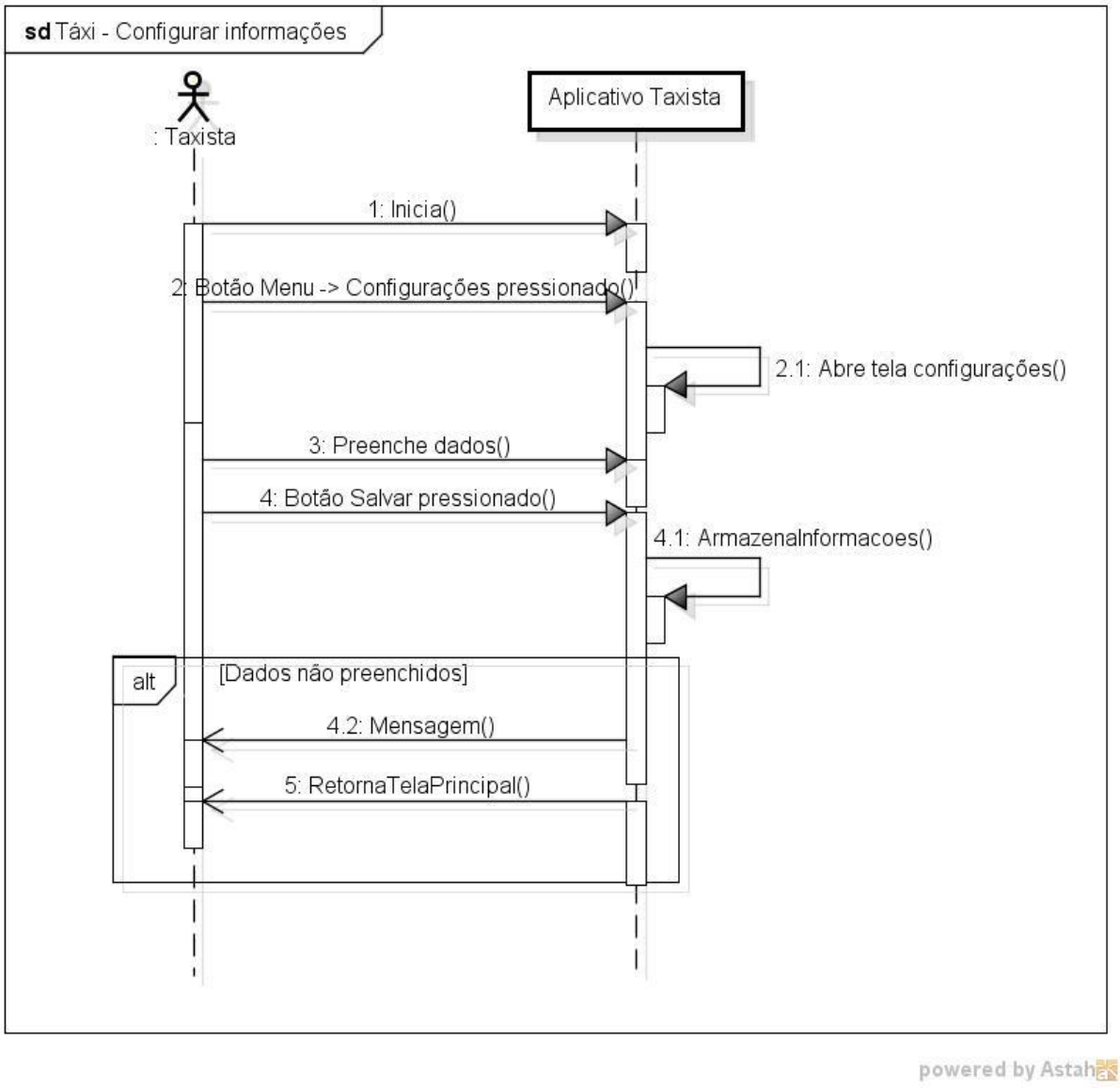


Figura 59 - Diagrama de sequência Configurar informações

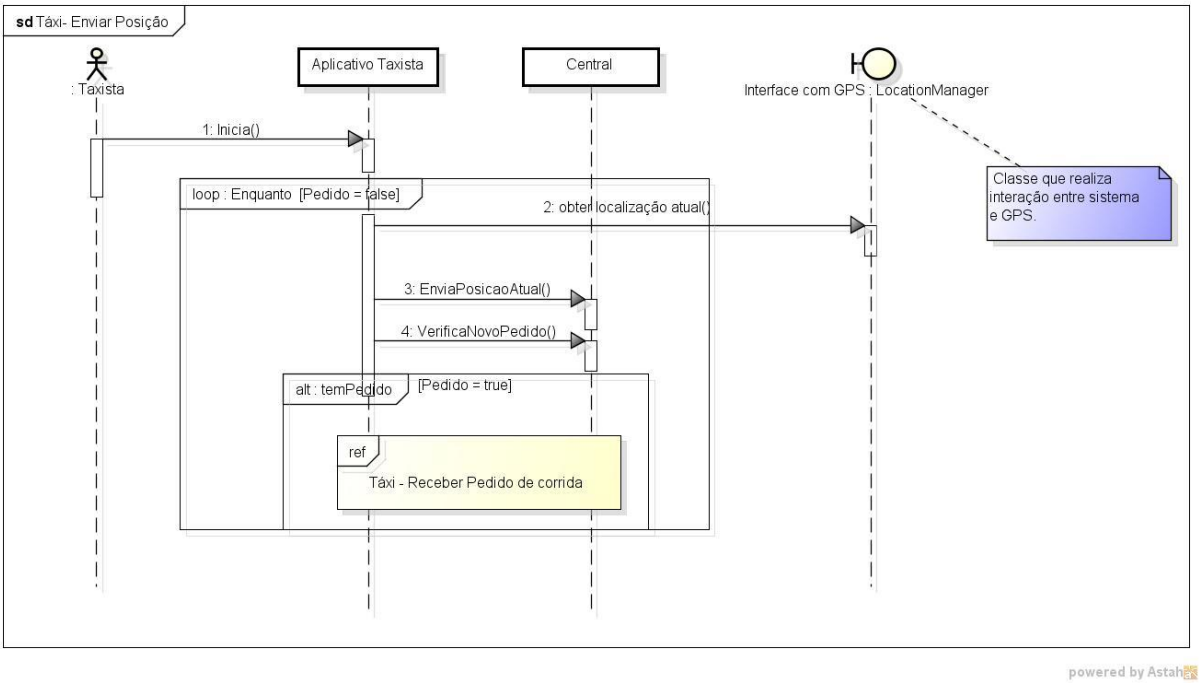


Figura 60 - Diagrama de sequência Enviar posição

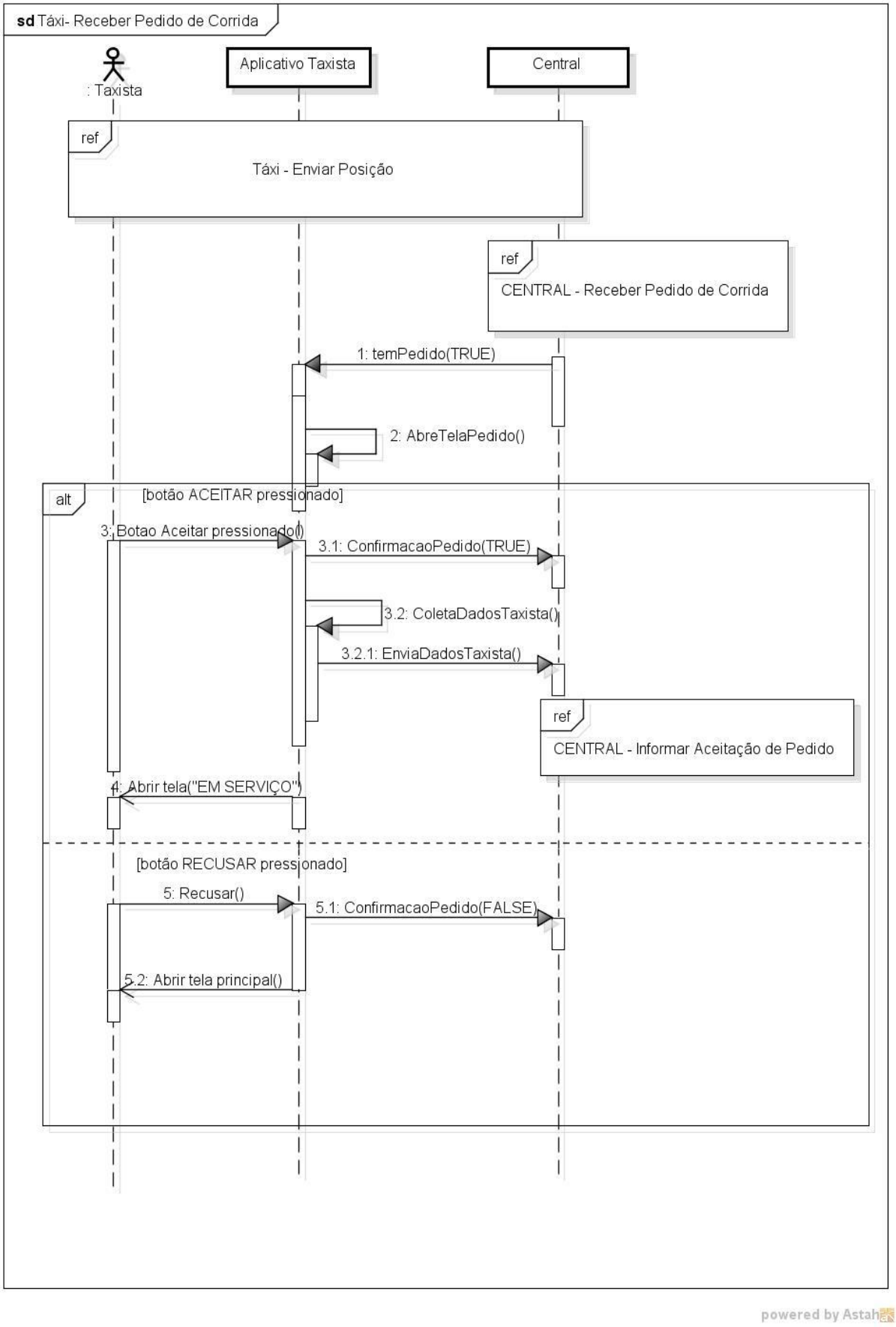


Figura 61 - Diagrama de sequência Receber pedido de corrida

APÊNDICE F – DIAGRAMA DE MÁQUINA DE ESTADOS

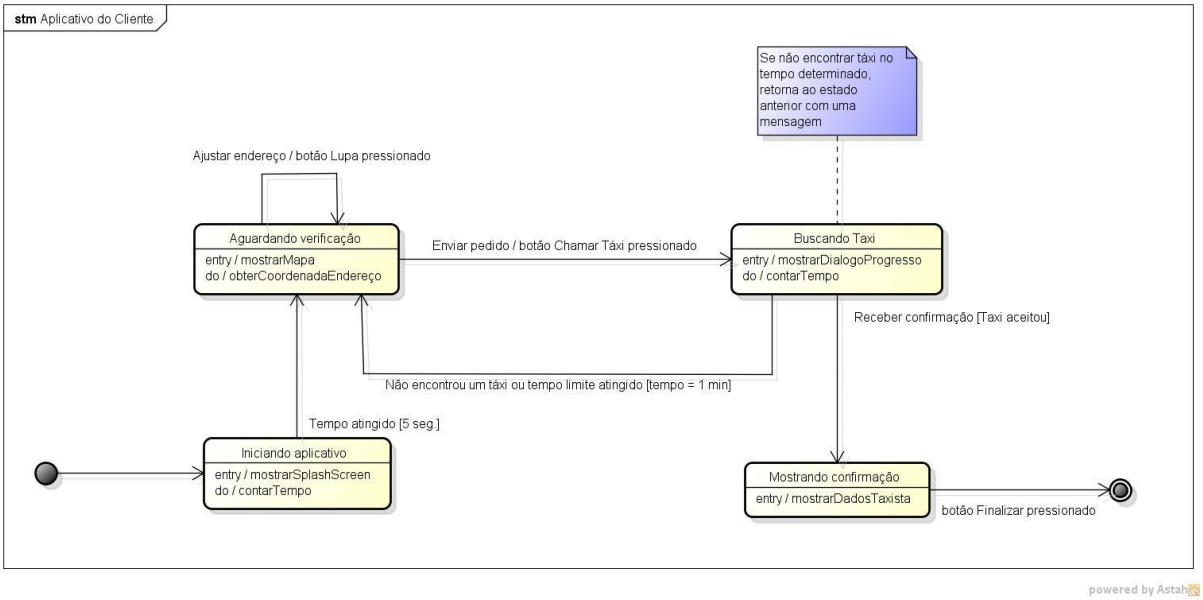


Figura 62 - Aplicativo cliente

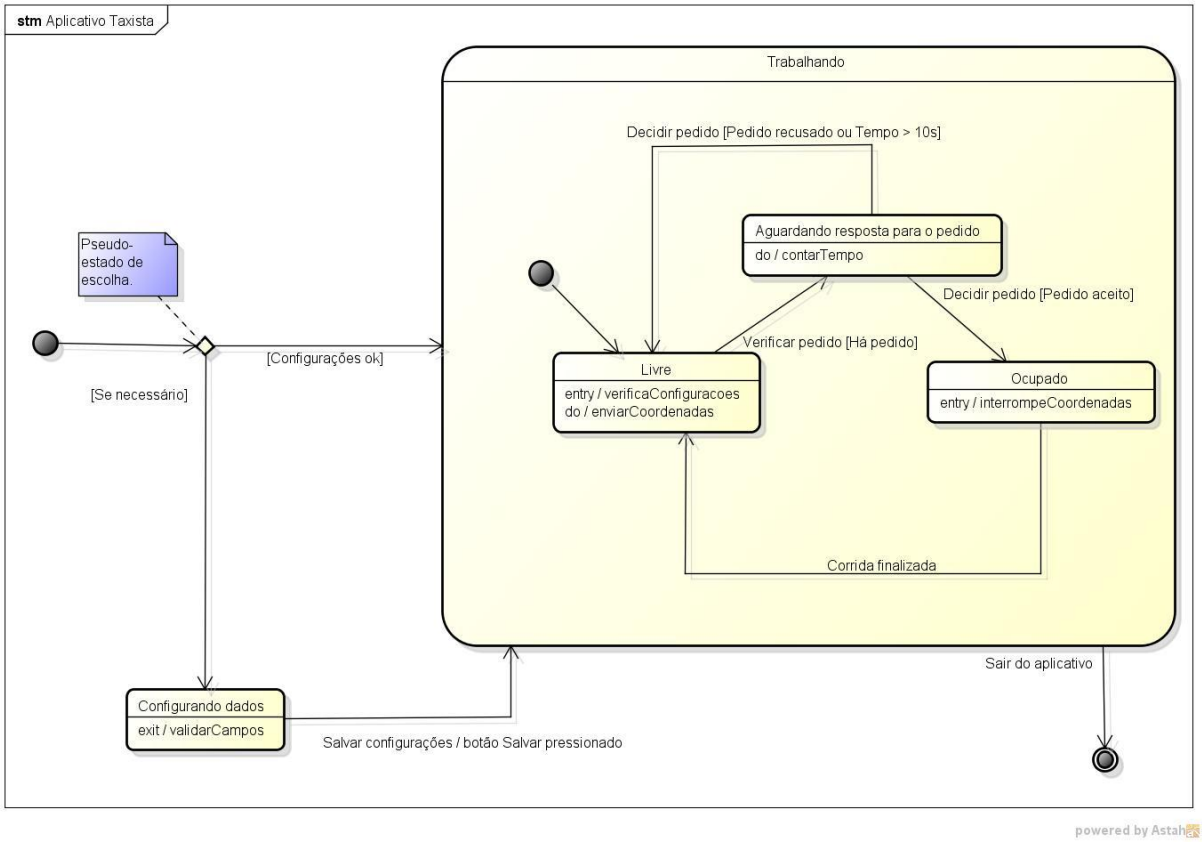


Figura 63 - Aplicativo táxi