FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Análise e Desenvolvimento de Sistemas

GUSTAVO ALMEIDA DOS SANTOS

PROTÓTIPO DE DESENVOLVIMENTO DE UM APLICATIVO MOBILE PARA LOCALIZAÇÃO DE ESTACIONAMENTOS

Americana, SP
2016

FACULDADE DE TECNOLOGIA DE AMERICANA

Curso Análise e Desenvolvimento de Sistemas

GUSTAVO ALMEIDA DOS SANTOS

PROTÓTIPO DE DESENVOLVIMENTO DE UM APLICATIVO MOBILE PARA LOCALIZAÇÃO DE ESTACIONAMENTOS

Trabalho de Conclusão de Curso desenvolvido em cumprimento à exigência curricular do Curso Análise e Desenvolvimento de Sistemas, sob a orientação do Prof. Me. Wladimir da Costa

Área de concentração: Aplicativo Móvel

Americana, SP
2016

FICHA CATALOGRÁFICA – Biblioteca Fatec Americana - CEETEPS
Dados Internacionais de Catalogação-na-fonte

Santos, Gustavo Almeida dos
S235p Protótipo de desenvolvimento de um aplicativo mobile para localização de estacionamentos. / Gustavo Almeida dos Santos. – Americana: 2016.
53f.

Monografia (Graduação em Tecnologia em Análise e Desenvolvimento de Sistemas). - - Faculdade de Tecnologia de Americana – Centro Estadual de Educação Tecnológica Paula Souza.
Orientador: Prof. Dr. André de Lima

1. Dispositivos móveis - aplicativos I. Lima, André de II. Centro Estadual de Educação Tecnológica Paula Souza – Faculdade de Tecnologia de Americana.

CDU: 681.519

Gustavo Almeida dos Santos

DESENVOLVIMENTO DE UM APLICATIVO MOBILE PARA LOCALIZAÇÃO DE ESTACIONAMENTOS

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Analise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – Fatec/ Americana.
Área de concentração: Aplicativo Móvel

Americana, 22 de Junho de 2016.

Banca Examinadora:

Wladimir da Costa
Mestre
FATEC Americana

Antônio Alfredo Lacerda
Especialista
FATEC Americana

André de Lima
Doutor
FATEC Americana

GUSTAVO ALMEIDA DOS SANTOS

PROTÓTIPO DE DESENVOLVIMENTO DE UM APLICATIVO MOBILE PARA LOCALIZAÇÃO DE ESTACIONAMENTOS

Trabalho de graduação apresentado como exigência parcial para obtenção do título de Tecnólogo em Análise e Desenvolvimento de Sistemas pelo CEETEPS/Faculdade de Tecnologia – FATEC/ Americana.

Área de concentração: Aplicativo Móvel

Americana, 26 de Junho 2016.

Banca Examinadora:

_____
Wladimir da Costa
Mestre
FATEC Americana

_____
Antônio Alfredo Lacerda
Especialista
FATEC Americana

_____
André de Lima
Doutor
FATEC Americana

AGRADECIMENTOS

Agradeço primeiramente aos meus pais pelo incentivo e apoio incondicional.

A todos os professores deste curso por serem essências no desenvolvimento desta monografia e na minha vida acadêmica.

Agradeço imensamente ao meu orientador e professor Wladimir da Costa por me aconselhar ao longo deste projeto.

E a todos os meus amigos que fizeram parte da minha formação.

DEDICATÓRIA

Dedico este trabalho aos meus pais, pelo apoio e paciência, aos meus amigos que aqui fiz e que eternamente estarão em minhas lembranças.

RESUMO

Atualmente os usuários comuns estão procurando cada vez mais celulares com diversos recursos como câmeras, músicas, *bluetooth*, ótima interface visual, jogos, *GPS*, acesso à internet e e-mails, e agora ainda temos a TV digital (LECHETA, 2015). Este trabalho consiste no desenvolvimento de um aplicativo usando a plataforma *Android* compatíveis com a versão 4.4 ou superior, neste trabalho será feita uma abordagem das tecnologias envolvidas para o um melhor entendimento do que será utilizado no desenvolvimento do aplicativo. Usando a tecnologia GPS, esta aplicação móvel tem como objetivo analisar as localizações dos estabelecimentos comerciais de estacionamento mais próximos do usuário, que irá usar o aplicativo quando estiver em ambientes de grande volume de carros como, por exemplo, centro de cidades grandes, eventos festivos, etc.... O aplicativo também irá comparar os preços, analisar avaliações de usuários e por fim listar as melhores alternativas na tela do smartphone, tornando, assim, de forma mais rápida, acessível e econômica encontrar um local para estacionar seu veículo.

Palavras Chave: *Bluetooth; GPS; Android*.

ABSTRACT

Today ordinary users are increasingly looking for mobile phones with various features like cameras, music, bluetooth, great visual interface, games, GPS, internet access and e-mails, and now we still have a digital TV (LECHETA, 2015). This work consists in developing an application using the Android platform compatible with version 4.4 or higher, this work will make an approach to the technologies involved for a better understanding of what will be used during the application development. Using GPS technology, this mobile application aims to analyze the locations of parking commercial establishments closer to the user, which will use the application while in high volume cars environments such as, for example, center of major cities or festive events, etc.... The application will also compare the prices and list the best alternatives on the smartphone screen, thus making faster, affordable and cost effective way to find a place to park your vehicle.

Keywords: Bluetooth; GPS; Android.

SUMÁRIO

LISTA DE FIGURAS E DE TABELAS

LISTA DE ABREVIATURAS E SIGLAS

SDK	*Software Development Kit*
GPS	*Global Positioning System*
SO	Sistema Operacional
IDE	*Integrated Development Environment*
API	*Application Programming Interface*
JAR	Java *Archive*
HTML	*Hypertext Markup Language*
XML	*Extensible Markup Language*
UML	*Unified Modeling Language*
MER	Modelo de Entidade Relacionamento
DER	Diagrama de Entidade Relacionamento
SQL	*Structured Query Language*
UI	*User Interface*

1. INTRODUÇÃO

Atualmente os usuários comuns estão procurando cada vez mais celulares com diversos recursos como câmeras, músicas, *bluetooth*, ótima interface visual, jogos, GPS, acesso à internet e e-mails, e agora ainda temos a TV digital (LECHETA, 2015). O mercado de smartphones cresce cada vez mais, chegando até 275,89 milhões de celulares no Brasil. Com a constante evolução de suas tecnologias, os smartphones continuam a revolucionar o cotidiano das pessoas. Tirar fotos, ouvir músicas, assistir TV, se comunicar, usar GPS, entre outros, tudo isso dentro de um aparelho extremamente portátil. Com tantas funcionalidades foi necessário o desenvolvimento de um sistema operacional, o *SO Android*, adquirido pela empresa Google em 2005 e lançado em 2007 é atualmente é a plataforma mais utilizada entre os smartphones.

De acordo com Darwin (2012), o desenvolvimento para o *SO Android* não é tão diferente do desenvolvimento para qualquer kit de ferramentas de UI (User Interface), incluindo Microsoft Windows, X Windows, Java Swing ou Adobe Flex. O que diferencia de outras plataformas é seu design arrojado e simples pois o objetivo é focar na facilidade de utilização do usuário. O *Android* é projetado para uma vida ativa, no qual um usuário está envolvido em várias tarefas: Atendendo ligações, verificando e-mail, enviando mensagens SMS, relacionando-se em redes sócias, tirando fotos, acessando a internet, executando aplicativos e mais, talvez até realizando um pouco de trabalho. Conclui-se que com o avanço da plataforma *Android,* a sociedade precisa andar lado a lado com as novas tecnologias para que cada vez mais melhore o seu dia a dia.

Este protótipo toma como referência o aplicativo LetsPark, que utiliza a tecnologia GPS para localizar comércios de estacionamentos dentro do Brasil, disponibilizado para a plataforma móvel Android e iOS, este aplicativo informa os estabelecimentos ao redor do usuário, junto com suas informações como nome, preço por hora e avaliação.

1.1. PROBLEMA

Principalmente no centro de cidades grandes, foi observado que grande parte dos motoristas têm dificuldade e passam muito tempo a procura de um lugar para

estacionar seu veículo por conta do grande volume de automóveis e falta de vagas de estacionamentos. Por conta disso, foram criados diversos estabelecimentos de estacionamentos, porém, a pouca organização de tantos comércios traz como consequência a falta de confiança de seus consumidores que tem como principais reclamações a ausência de segurança e a discrepância de preços entre os estabelecimentos.

1.2. OBJETIVOS

Este trabalho tem como objetivo desenvolver aplicativo protótipo usando a plataforma *Android* compatíveis com a versão 4.4 ou superior, utilizando o serviço de geoprocessamento Google Maps API, recursos como mapas, localização e navegação por GPS e obtenção de endereços. Esta aplicação móvel tem como objetivo analisar as localizações dos estabelecimentos comerciais de estacionamento mais próximos da localização atual do usuário, que irá usar o aplicativo quando estiver em ambientes de grande volume de carros como, por exemplo, centro de cidades grandes, eventos festivos, etc.... O aplicativo também irá comparar os preços, analisar as avaliações de usuários e por fim listar as melhores alternativas na tela do smartphone.

Outros aplicativos como o *Waze* e o *iGo* usam a navegação GPS e disponibilizam funcionalidades similares ao projeto a ser construído, porém estes aplicativos tem o foco de fazer localizações de lugares não específicos, informando o nome e a localização geográfica de algum ponto ou comercio desejado. O iParking traz como característica própria de analisar os preços e avaliações dos estacionamentos localizados na cidade de Americana.

1.2.1. OBJETIVOS ESPECIFICOS

• Utilizar o levantamento bibliográfico contendo ideias e teorias para promover um melhor entendimento das tecnologias implantadas para o desenvolvimento deste aplicativo.

• Desenvolver um protótipo de aplicativo móvel na plataforma *Android* usando os conhecimentos adquiridos em sala de aula e nos materiais online.

• Analisar o uso do *Android* junto com a tecnologia GPS, usando a ferramenta Google Maps API (Application Programming Interface) para localizações geográficas e obtenção de endereços.

1.3. JUSTIFICATIVA

Por conta dos problemas observados, este trabalho visa desenvolver um protótipo de aplicativo usando recursos e métodos acessíveis, utilizando a tecnologia de GPS para identificar os locais geográficos dos comércios e estacionamentos na cidade de Americana, comparando os preços ao consultar os dados de diversos estabelecimentos diferentes. Tornando, assim, de forma mais rápida, acessível e econômica encontrar um local para estacionar seu veículo.

2. SISTEMA OPERACIONAL ANDROID

Segundo Lecheta (2015), mais de 3 bilhões de pessoas possuem um telefone celular, Diante da constante evolução de suas tecnologias, os smartphones continuam a revolucionar o cotidiano das pessoas. Tirar fotos, ouvir músicas, assistir TV, se comunicar, usar GPS, entre outros, tudo isso dentro de um aparelho extremamente portátil. Com tantas funcionalidades foi necessário o desenvolvimento de um sistema operacional, o SO *Android*, adquirido pela empresa Google em 2005 e lançado em 2007 é atualmente a plataforma mais utilizada entre os smartphones.

O funcionamento do sistema operacional (SO) *Android* tem como principal função gerenciar todos os processos do aplicativo e do hardware para que o dispositivo funcione adequadamente, seu funcionamento é semelhante a outros SO como Debian, Windows, Mac Os, entre outros. Neste trabalho será usado o *Android* Studio, esta ferramenta provê um ambiente de desenvolvimento, debug e testes para *Android.*

De acordo com Monteiro (2013), a facilidade de desenvolver utilizando uma linguagem de programação (Java) bastante disseminada, a simplicidade e baixo custo para a publicação de aplicativos na loja Google Play e a quantidade de dispositivos Android em uso no mundo só fazem aumentar a popularidade da plataforma.

Para Bordin (2015), o *Android* é mais do que um sistema operacional, ele é na verdade um software *stack* composto por cinco camadas, a base do *Android* é uma versão modificada do *kernel* Linux, que prove vários serviços essenciais, como segurança, rede e gerenciamento de memória e processos, além de uma camada de abstração de hardware para as outras camadas de software. Cada camada da pilha agrupa programas que utilizam funções do sistema operacional. No começo do *kernel* do Linux, existem os middleware, bibliotecas e APIs escritos em C em uma estrutura de aplicativo que inclui bibliotecas compatíveis com Java. Entretanto, o desenvolvimento do *kernel* do Linux continua independentemente de outras bases de código fonte do *Android*.

Na Figura 1 estão descritos os dados sobre o número de dispositivos que utilizam atualmente uma determinada versão da plataforma *Android* com base em dispositivos que acessam o *Google Play Store* em um período de sete dias que termina em 07 de março de 2016.

Figura 1 – Distribuições das Versões da plataforma *Android*

Version	Codename	API	Distribution
2.2	Froyo	8	0.1%
2.3.3 - 2.3.7	Gingerbread	10	2.6%
4.0.3 - 4.0.4	Ice Cream Sandwich	15	2.3%
4.1.x	Jelly Bean	16	8.1%
4.2.x		17	11.0%
4.3		18	3.2%
4.4	KitKat	19	34.3%
5.0	Lollipop	21	16.9%
5.1		22	19.2%
6.0	Marshmallow	23	2.3%

Fonte: *ANDROID DEVELOPERS* (2016)

Conforme evidenciado na Figura 1, a versão 4.4 *KitKat* é atualmente a mais usada entre os dispositivos móveis. As versões mais recentes oferecem, muitas

vezes, melhores APIs para o aplicativo mas é recomendável continuar apoiando versões anteriores para ter uma rede maior de compatibilidade.

2.1. BIBLIOTECAS DO AMBIENTE ANDROID

Bibliotecas são funções já disponibilizadas na plataforma que servem para solucionar determinado problema. Bibliotecas são instruções que dizem ao dispositivo como se comportar com os diferentes tipos de dados recebidos. No Java também é possível exportar bibliotecas de terceiros para que possam ser reusadas em uma outra aplicação, compartilhando seus recursos.

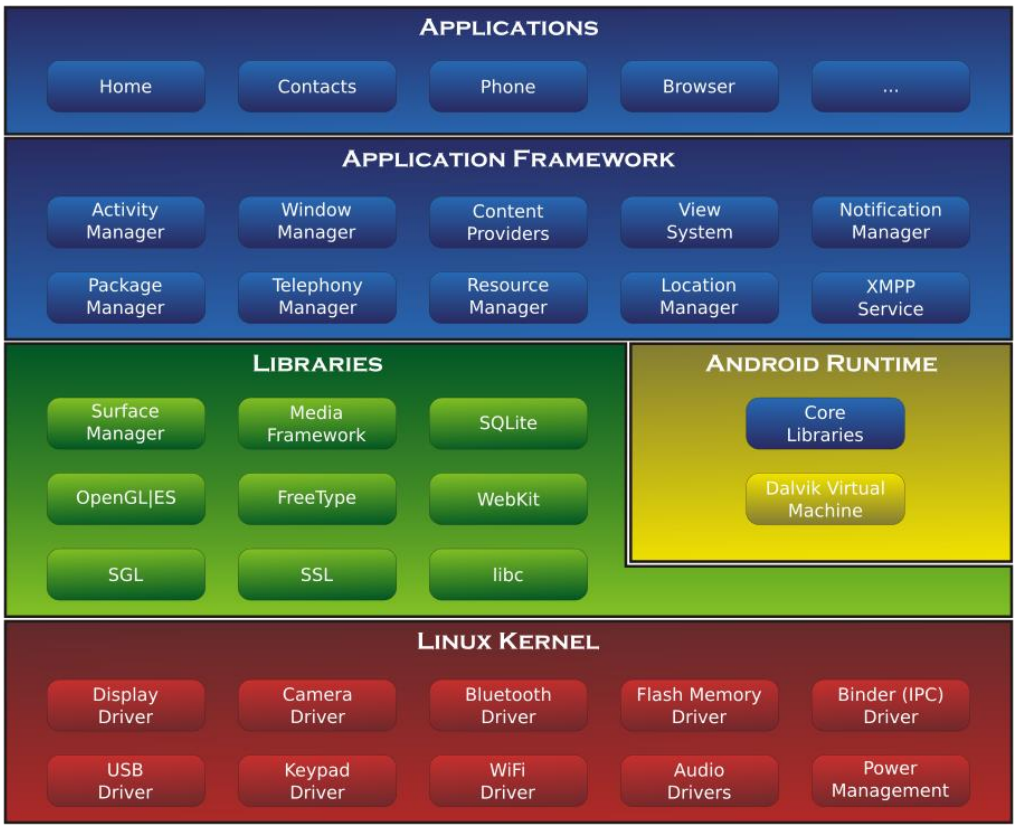
> É muito comum criarmos um JAR com um conjunto de classes que podem ser reutilizados em outros projetos. Em outras palavras, criar uma biblioteca Java. Por sinal, o grande número de bibliotecas gratuitas em Java é um dos fatores que fazem a linguagem ser tão bem aceita no mercado. Precisa gerar relatórios, manipular XMLs, conectar ao banco de dados, gerar códigos de barras etc.? Você já tem tudo isso pronto! O ecossistema do Java é enorme. (TURINI; RODRIGO, 2015, p. 139).

É muito comum achar bibliotecas Java em vários sites diferentes, que cobrem tipos de problema de programação onde um desenvolvedor Java pode encontrar. Um desenvolvedor ocasionalmente se depara com a necessidade de interagir com outros programas, daí vem a necessidade de utilizar Bibliotecas externas em projetos para que possam reutilizar outras funcionalidades.

2.2. FRAMEWORKS NO AMBIENTE *ANDROID*

Frameworks são conceitos que ajudam no desenvolvimento de uma atividade ou de um processo. Em sistemas de computador, um *Framework* é muitas vezes uma estrutura em camadas indicando que tipo de programas podem ou devem ser construídos e como eles irão se relacionar internamente. Um *Framework* pode ser para um conjunto de funções dentro de um sistema e como eles se relacionam, como a comunicação deve ser padronizado em algum nível de uma rede e assim por diante (ROUSE, 2015, tradução nossa). Mais detalhes são ilustrados abaixo no diagrama da Figura 2.

Figura 2 – Diagrama da arquitetura do Android



Fonte: Google I/O (2014)

O *Framework* faz parte da pilha dentro da arquitetura do *Android* em que as aplicações interagem diretamente. Estes programas gerenciam as funções básicas do sistema, como gestão de recursos, gerenciamento de chamadas de voz, gerenciador de janelas, gerenciador de atividades, etc. O *Framework* faz parte de uma camada do *stack Android*, é um conjunto de ferramentas que fornecem vários serviços para os aplicativos, visando o reuso de componentes (Android Developers, 2012 apud BORDIN, 2012).

2.3. APLICAÇÕES ANDROID

Com as aplicações Android podemos fazer praticamente qualquer coisa. Comunicar com amigos e familiares, passar o tempo com um jogo, registrar os momentos importantes em nossas vidas e compartilhar esses momentos quase instantaneamente com qualquer pessoa do outro lado do mundo. Apps nos ajudam a encontrar nosso caminho, eles nos guiam, e o mais importante, nos auxiliam na maioria das atividades do cotidiano.

Os Aplicativos do *Android* são desenvolvidos na linguagem Java, usando o sistema de desenvolvimento do software *Android SDK*. O SDK contém um conjunto de ferramentas para desenvolvimento que incluem bibliotecas, depuradores, documentações, etc.

> A plataforma *Android* desfruta hoje de um papel de destaque no mercado, tanto pela quantidade significativa de dispositivos produzidos como também por oferecer uma API rica, disponibilizando fácil acesso a vários recursos de hardware, tais como Wi-Fi e GPS, além de boas ferramentas para o desenvolvedor. (MONTEIRO, 2013, p. 22).

O melhor jeito de distribuir um aplicativo *Android* é pela Google Play. Carvalho, G. (2012) explica que publicar um aplicativo não é difícil, o desenvolvedor pode fazer isso em alguns passos e disponibilizar seu aplicativo para download em pouco tempo. O Google Play é o painel dedicado do Google para desenvolvedores que permite gerenciar e disponibilizar todos os aplicativos para os usuários. O Google só exige uma taxa de inscrição que será paga uma vez antes que fazer upload de um aplicativo.

De acordo com Chahoud (2014), Há várias maneiras de ganhar dinheiro com aplicativos no Google Play, se o objetivo é monetizar o aplicativo, o desenvolvedor deverá planejar uma estratégia, é recomendável usar um ou mais dos principais modelos de negócios detalhados nos itens que seguem:

Publicidade: É importante colocar anúncios no seu aplicativo, é importante lembrar que muitas pessoas vão estar usando seu aplicativo em smartphones. Para os jogos, um bom lugar para colocar anúncios é após o usuário concluir um nível onde eles veem o anuncio antes de prosseguir, e como muitos outros *apps* fazem, você pode colocar um pequeno banner.

Taxa de inscrição: É preciso analisar seu aplicativo e ver se há uma maneira de fazer um modelo de assinatura pois nem todos são compatíveis para tal modelo. Cobrar dos usuários uma certa quantidade de dinheiro em um mês ou até mesmo por ano por uma assinatura para receber conteúdo exclusivo.

Freemium: O modelo de negócio “*Freemium*” é uma junção de grátis e pago ao mesmo tempo, Alguns usuários não gostam de *apps* que usam desses tipos de modelo de negócio, *Freemium* basicamente significa que os usuários podem fazer o download do aplicativo gratuitamente, mas alguns recursos serão pagos. Uma das maneiras mais comuns de usar *Freemium*, é quando o usuário paga para remover anúncios permanentemente da aplicação, o desenvolvedor também pode adicionar alguns recursos adicionais.

Dependendo do tipo de aplicação que será desenvolvida, é recomendável considerar outros tipos modelos de negócios. Estudar diversos modelos mais afundo é um passo importante para o sucesso na monetização de um *app*.

2.4. ANDROID STUDIO

Android Studio é um ambiente de desenvolvimento integrado (IDE) para a plataforma *Android*. Carvalho, S. (2013) acrescenta que no Android Studio, a interface é bem versátil, podendo utilizar diversos estilos diferentes, Também é possível customizar os atalhos de teclado, de forma que sejam iguais a outras *IDEs*, igual ao o seu concorrente, Eclipse. Desde seu lançamento, o *Android* Studio vem atraindo muitos desenvolvedores por conta das suas novas funcionalidades e uma melhor integração com o sistema. O recurso que mais se destacou nesta ferramenta é a criação de layout, seu recurso de *preview* torna quase desnecessário executar um emulador para o teste do aplicativo.

Para Lecheta (2015), o *Android Studio*, apresenta boas práticas de desenvolvimento de interface no padrão Google usando o estilo Material Design, e a vantagem principal é que essa IDE foca diretamente no desenvolvimento de aplicativos o que é um facilitador para muitos usuários.

Segundo Carvalho, S. (2013), Um dos principais motivos para que muitos desenvolvedores estão abandonando a *IDE Eclipse* é que ela está sendo descontinuada para o desenvolvimento de aplicativos móveis, entre outros motivos o Android Studio também apresenta uma nova e melhorada interface de design onde você pode visualizar a interface que está trabalhando e seus componentes. Quando você usa o modo de exibição de projeto no *Android Studio*, você percebe que a

estrutura do projeto parece diferente do que você pode ter usado no Eclipse. Cada instância do Android Studio contém um projeto com um ou mais módulos de aplicativos e cada módulo de aplicativo contém os conjuntos de código fonte para esse módulo.

Ao finalizar, o desenvolvedor poderá compilar todo seu trabalho e então disponibilizar suas aplicações *Android* de várias maneiras diferentes. Normalmente as aplicações são disponibilizadas através do Google Play, mas também é possível publicar as aplicações em um site próprio ou enviando um pedido diretamente a um usuário.

2.5. GOOGLE MAPS API

O *Google Maps API* (Interface de Programação de Aplicações) possibilita usar a funcionalidade de mapas geográficos no desenvolvimento de um aplicativo *Android*. Os métodos da API serão usados para adicionar os locais de estacionamentos deste projeto, esta função ajuda a fornecer informações adicionais para os locais, e permite a interação do usuário com o mapa.

Esse serviço é disponibilizado gratuitamente para a maioria dos casos, dependendo para o que o desenvolvedor irá utilizar, para usar este recurso, é gerado uma chave de API obtida para o uso do *Google Maps*, cada chave de *API* é única pois cada desenvolvedor tem sua própria *keystore*. Os processos necessários para integrar o *Google Maps* com *apps Android* são relativamente complexos e não recomendado para iniciantes (ANDROID DEVELOPERS, 2016). É possivel escolher entre o plano padrão (grátis) ou o premium (pago), suas diferenças são listadas na Figura 3.

Figura 3 – Diferenças entre o plano Padrão e Premium do Google Maps API

Padrão	Premium
Implementações gratuitas externas publicamente disponíveis e para dispositivos móveis e web	Contratos anuais com termos empresariais
Preços e limites de uso individuais da API	Suporte técnico 24 horas
Uso ilimitado e gratuito de Google Maps	Acordo de nível de serviço (ANS)
Pague por aumentos acima dos limites de uso	Recursos avançados da API e sem publicidade

Fonte: Android Developers (2016)

3. ENGENHARIA DE SOFTWARE

De acordo com Pressman (2011), a engenharia de software envolve um conjunto de práticas usando metodologias que envolve comunicação, planejamento, modelagem, construção e emprego. Ao longo da construção de um software, essas práticas deverão estar presentes dês do começo ao fim do projeto, visando a organização dos processos para que o sistema possa ser desenvolvido e finalizado de acordo com o prazo tendo em vista a qualidade.

Os engenheiros de software praticam uma abordagem sistemática e organizada que é na maioria dos casos a maneira mais eficaz de desenvolver um software de alto nível, porém a engenharia procura selecionar o método mais apropriado para um conjunto de circunstancias, uma abordagem mais criativa e menos formal pode ser mais eficaz em outras situações. O desenvolvimento menos formal é particularmente conveniente para sistemas baseados na Web, que requerem uma combinação de design gráfico e em desenvolvimento de *software*. (SOMMERVILLE; IAN, 2007).

Devido ao avanço das tecnologias, o software, nos últimos anos, ultrapassou o hardware como a chave para o sucesso de muitos sistemas de computação. Um software fará muita diferença em qualquer tipo de negócio, seja para gerenciar uma loja, controlar um produto, capacitar um sistema, etc. A qualidade do serviço oferecido pelo sistema diferenciam uma empresa de suas concorrentes. A habilidade de um programa ser produtivo e ao mesmo tempo prático o diferencia dos produtos concorrentes que tenham função idêntica em outros aspectos.

O software certamente fará a diferença nos negócios do cliente, a inteligência e a função oferecidas pelo software muitas vezes diferenciam dois produtos ou industrias idênticas. (PRESSMAN; Roger S., 2011). O engenheiro de software analisa o funcionamento de um sistema por meio de uma avaliação, onde ele define os problemas, necessidades e soluções. Ele irá documentar e demonstrar um meio de ajudar a resolver os problemas através das documentações, gráficos e diagramas, sempre com a finalidade de melhorar a qualidade do gerenciamento do projeto.

A engenharia de software é uma área da engenharia onde o objetivo é gerenciar o processo de desenvolvimento de sistemas de software. Um software é algo intangível e não limitado por materiais ou processos de manufatura, simplificando assim os procedimentos da engenharia de software pois não existem limitações físicas, porém, a falta de organização pode comprometer o software tornando-o mais complexo e muito difícil de ser assimilado (SOMMERVILLE; IAN, 2007).

Neste tópico será abordado as práticas relacionadas a engenharia de software que serão implementadas no desenvolvimento do projeto. A engenharia de software tem como objetivos a aplicação de teorias, modelos e técnicas da ciência da computação ou desenvolvimento de software. Todos os processos aplicados aqui serão explicados de forma que o leitor entenda a importância de cada procedimento.

3.1. REQUISITOS

A engenharia de requisitos refere-se ao processo de definição, documentação e manutenção de requisitos do cliente e usuário. Para o engenheiro de software, entender os requisitos de um problema é uma das tarefas mais complicadas, ao pensar sobre essa engenharia na primeira vez pode soar como um processo simples, porém muitas vezes os usuários finais não tem um bom entendimento das características e funções ou o cliente não sabe exatamente o que é necessário. Mesmo que o cliente e usuários fossem específicos quanto as necessidades, elas mudariam ao decorrer do projeto. (PRESSMAN; Roger S., 2011).

> Talvez o maior problema que enfrentamos no desenvolvimento de sistemas de software grandes e complexos seja o da engenharia de requisitos. Ela está relacionada com a definição de que o sistema deve fazer, suas propriedades emergentes desejáveis e essenciais e as restrições quanto a operação do sistema e quanto aos processos de desenvolvimento de software. Você pode, portanto pensar na engenharia de requisitos como um processo de comunicação entre os clientes e usuários de software e os desenvolvedores de software. (SOMMERVILLE; IAN, 2007, p.78).

O uso da engenharia de software nos requisitos proporciona os conhecimentos que serão usados para entender o que o cliente quer, analisando as necessidades, soluções, viabilidade, especificações e também gerenciando as necessidades de forma que são transformadas em um sistema. Segundo Pressman (2011), esse

processo é separado em sete partes diferentes, mas sempre são adaptadas de acordo com a necessidade do projeto:

Concepção: É o primeiro passo para iniciar todo o processo do desenvolvimento, após identificar a necessidade de um novo negócio o engenheiro irá estabelecer um entendimento do problema, quais os interessados na solução, a raiz do problema e a comunicação entre os envolvidos.

Levantamento: Nessa parte o engenheiro pergunta para o cliente por meio de entrevistas quais os problemas que o sistema deverá solucionar, geralmente há confusões durante este processo por que o cliente não tem uma visão do sistema como o desenvolvedor, durante a especificação dos detalhes técnicos são muitas vezes abordadas informações desnecessárias ou até mesmo podem ofuscar problemas que acharem óbvios.

Elaboração: Aqui serão coletadas todas as informações obtidas na parte de concepção e levantamento para que esses dados possam ser filtrados e refinados com o objetivo de identificar como o usuário irá interagir com o sistema. Serão elaborados os diagramas de acordo com a definição do cenário do usuário e todos atributos de sua classe.

Negociação: Clientes podem requisitar funções que não podem ser atingidas, essas divergências devem ser abordadas na negociação. É necessário que as partes interessadas façam uma análise de seus requisitos e decidam as prioridades, levando em conta os custos e riscos. Após a negociação, os requisitos podem ser modificados, de forma que todos os lados interessados fiquem satisfeitos.

Especificação: A especificação pode ser uma combinação de um documento escrito, conjunto de modelos gráficos, modelo matemático formal, conjunto de cenários de casos de uso ou um protótipo, o importante é que essa especificação seja clara e objetiva nos olhos do cliente.

Validação e gestão: Nesta parte é analisado todo o documento de especificação para detectar possíveis inconsistências ou omissões de requisitos onde

serão corrigidos de acordo com o padrão estabelecido pelo cliente. Ao longo do projeto os requisitos podem mudar, na gestão de requisitos é utilizado algumas atividades que ajudam a manter as necessidades dessas mudanças ao longo do projeto.

Na área de Engenharia de Software, os requisitos são denominados em classes de definições: O requisito funcional e o requisito não funcional.

3.1.1. REQUISITOS FUNCIONAIS

De acordo com Sommerville (2007), os Requisitos Funcionais são os tipos de serviços que o sistema deve fornecer, como o sistema deve reagir a tipos de entradas e como o sistema deve se comportar em situações diferentes, os requisitos funcionais também podem estabelecer o que o sistema não deve fazer.

Esse tipo de requisito descreve um comportamento do sistema e como ele se relaciona com a funcionalidade, geralmente é descrita de forma abstrata detalhando suas entradas, saídas, exceções, etc.... Um exemplo de requisito funcional seria, por exemplo, neste TCC: “A função de avaliar um estacionamento que foi rastreado será opcional, sendo escolhido pelo usuário”.

3.1.2. REQUISITOS NÃO FUNCIONAIS

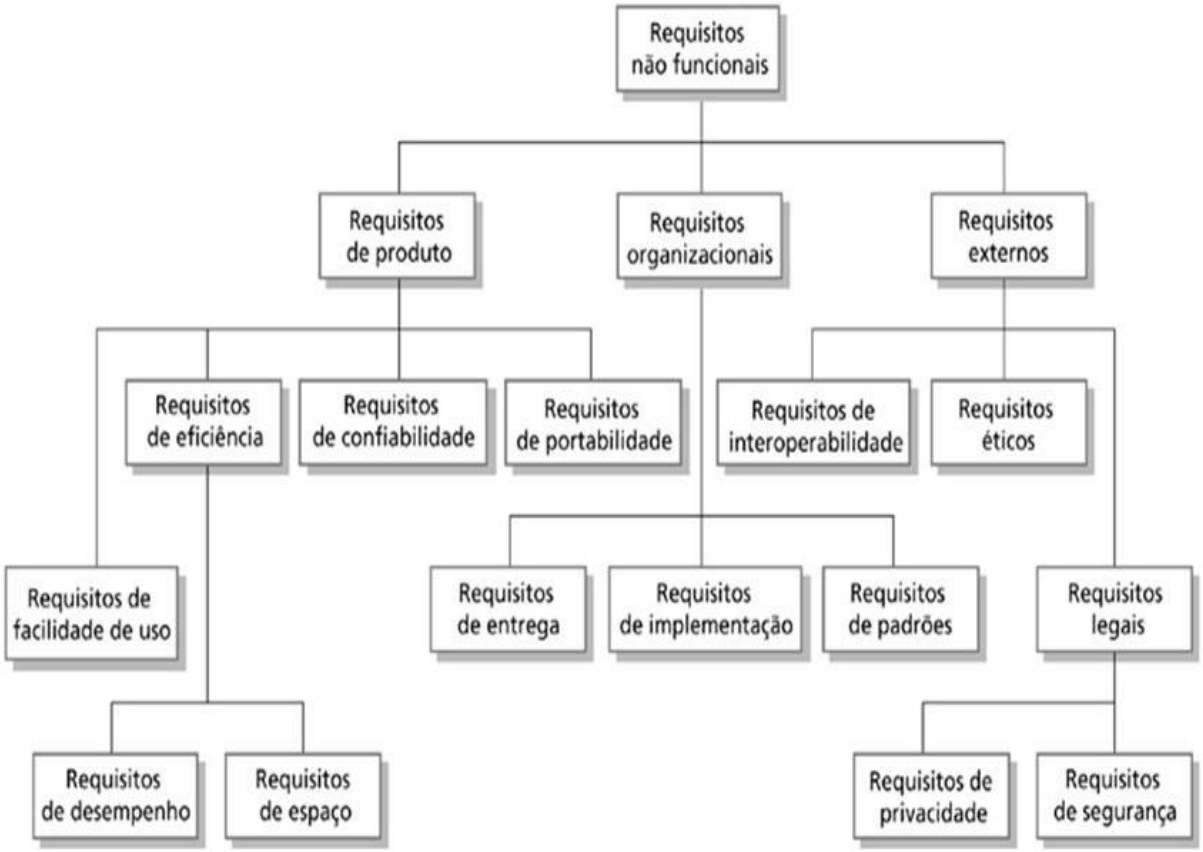
Para Engholm Junior (2010), no quesito qualidade de software, é apresentado uma série de objetivos da construção de software, conhecidos como requisitos não-funcionais, onde são denominados como extensibilidade, capacidade de manutenção, reutilização de código, desempenho, escalabilidade, usabilidade e confiabilidade nos dados apresentados pela aplicação.

> Os requisitos não funcionais surgem devido as necessidades do usuário, as restrições de orçamento, as políticas organizacionais, a necessidade de interoperabilidade com outros sistemas de software ou hardware ou a fatores externos como regulamentos de segurança ou legislação a respeito de privacidade (SOMMERVILLE, 2007, p.82).

Conforme ilustrado na Figura 4 a seguir, observa-se que os requisitos não funcionais podem seguir o caminho de **requisitos de produto** onde ficam as características necessárias do software, **requisitos organizacionais**, estabelecendo

políticas e procedimentos vindos da organização que desenvolve o *software* ou **requisitos externos** que são provenientes de fontes externas.

Figura 4 – Tipos de requisitos não funcionais



Fonte: Sommerville (2007, p.82)

3.2. LINGUAGEM DE MODELAGEM UNIFICADA

O *Unified Modeling Language (UML)* é uma linguagem de modelagem que tem o objetivo de representar visualmente uma parte do sistema em vários tipos de diagramas diferentes, onde cada um tem sua função de esclarecer atividades especificas, este processo é fundamental na construção de um sistema porque ele auxilia a entender como funciona a comunicação dos objetos e entidades do projeto. Serão abordados no estudo de caso alguns diagramas de modelagem baseados no sistema a ser desenvolvido neste trabalho de graduação.

> Cada diagrama de UML analisa o sistema, ou parte dele, sob uma determinada óptica. É como se o sistema fosse modelado em camadas, sendo que alguns diagramas enfocam o sistema de forma

> mais geral, apresentando uma visão externa do sistema, como é o objetivo do Diagrama de Casos de Uso, enquanto outros oferecem uma visão de uma camada mais profunda do software, apresentando um enfoque mais técnico ou ainda visualizando apenas uma característica específica do sistema ou um determinado processo. A utilização de diversos diagramas permite que falhas sejam descobertas, diminuindo a possibilidade da ocorrência de erros futuros. (GUEDES; GILLEANES, 2011, p.30)

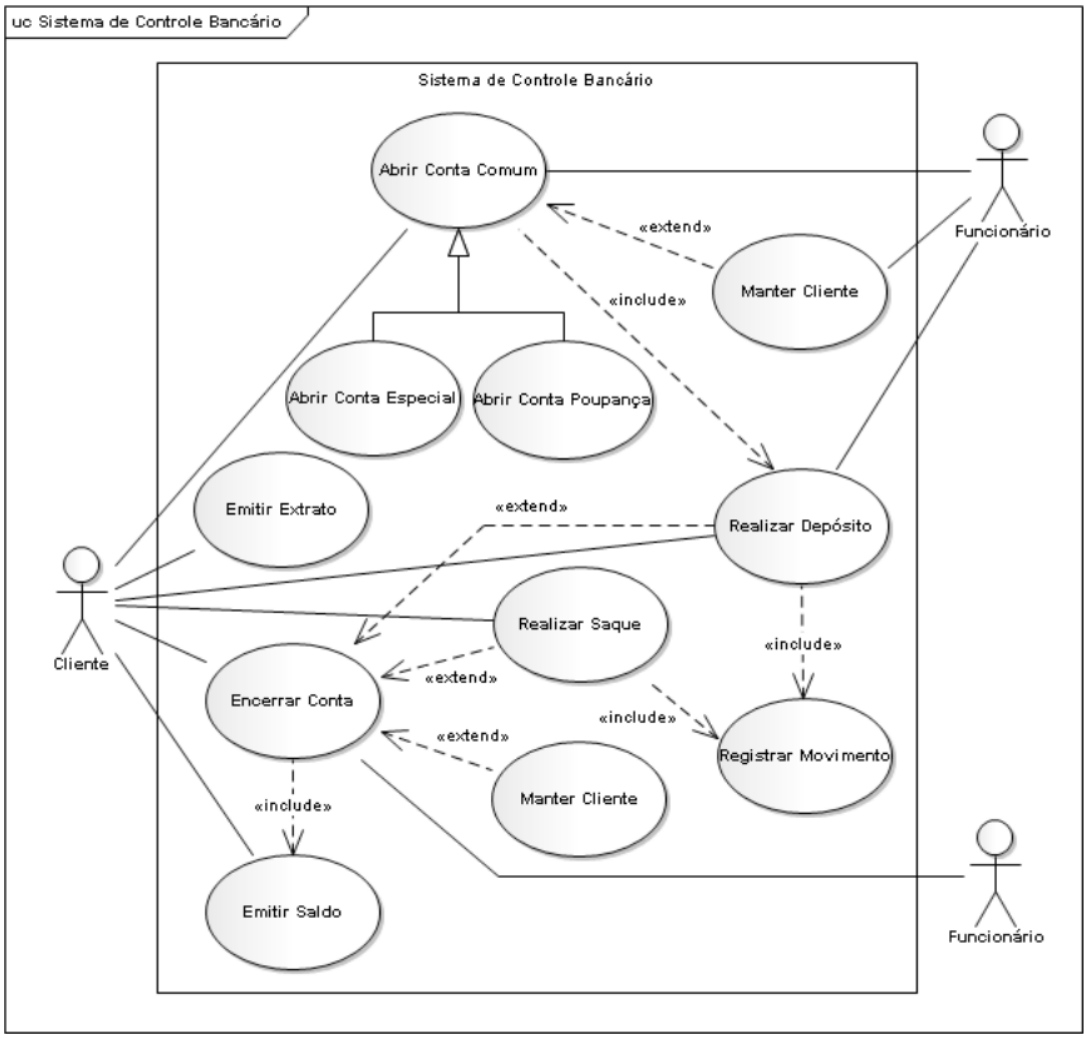
Para Guedes (2011), o que faz um sistema produtivo ao usar o UML é o auxílio que isso provem aos desenvolvedores, determinando as necessidades do sistema com mais facilidade, modelar um sistema também é uma forma bastante eficiente de documentá-lo. Para garantir a qualidade e reduzir defeitos, um sistema de informação precisa ter uma documentação muito detalhada, sendo constantemente atualizado para que possa ser mantido com facilidade sem ocasionar novos erros ao corrigir os antecessores.

3.2.1. CASO DE USO

O diagrama de caso de uso é usado na fase de levantamento e análise de requisitos, esse diagrama é importante porque serve de base para os outros. O caso de uso apresenta uma linguagem simples e mostra as funcionalidades principais do sistema. Guedes (2011) acrescenta que os casos de uso têm o objetivo de dar uma visão externa e geral das funcionalidades do sistema sem entrar em detalhes de como elas irão funcionar.

No exemplo da Figura 5 observa-se um sistema de controle bancário, onde o cliente pode abrir ou encerrar contas, depositar ou sacar dinheiro e emitir saldos ou extratos.

Figura 5 – Diagrama de Casos de Uso – Sistema de Controle Bancário



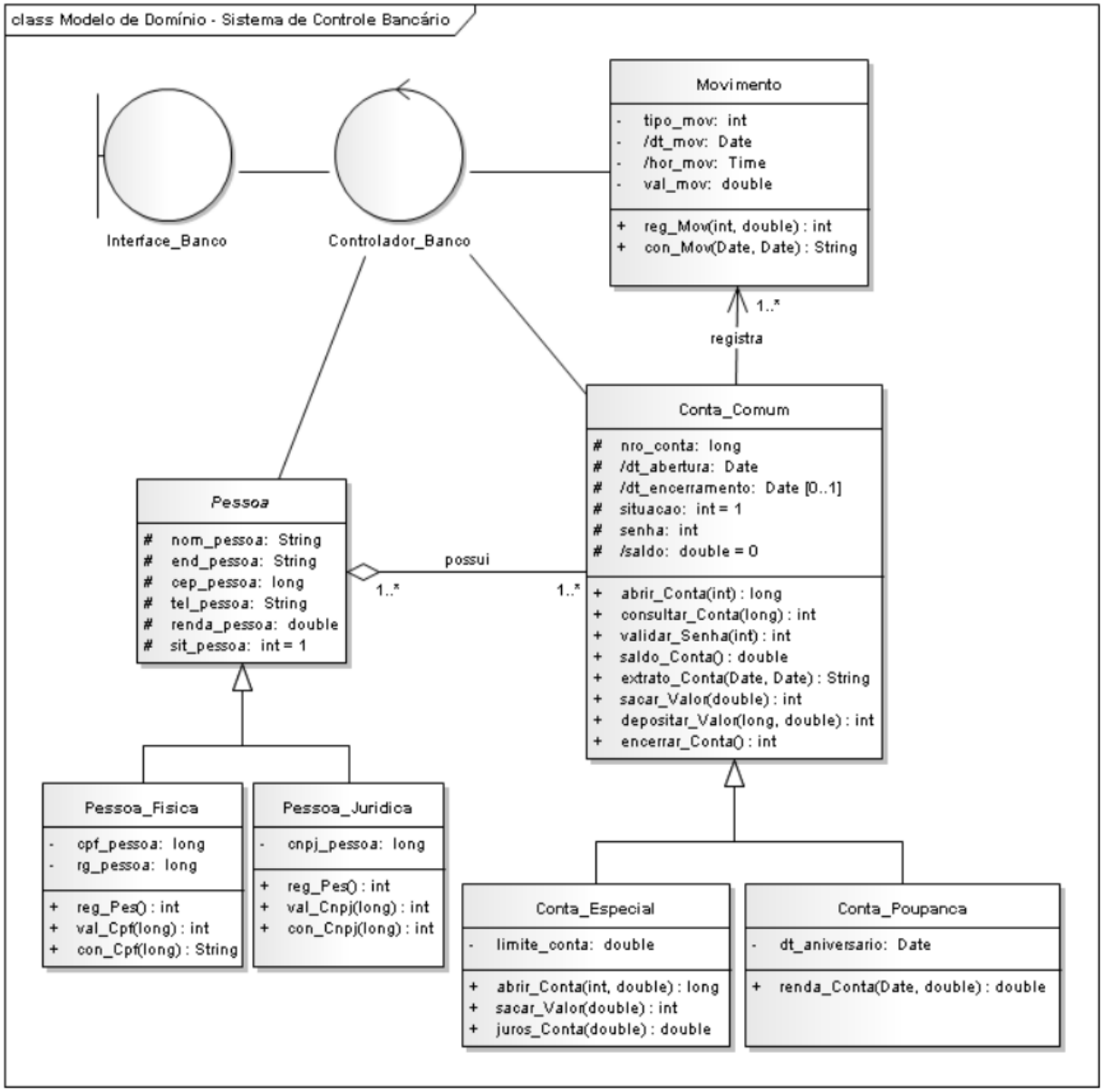
Fonte: Guedes (2011, p.73)

3.2.2. DIAGRAMA DE CLASSES

De acordo com Guedes (2011), o diagrama de classes é o mais usado e provavelmente o mais importante, ele tem o objetivo de representar a estrutura de todas as classes do projeto e de mostrar os atributos e métodos usados por cada uma delas, detalhando como se comunicam entre si, também serve de base para outros diagramas.

Continuando com o mesmo exemplo, a Figura 6 detalha como seria o diagrama de classes para um sistema de controle bancário.

Figura 6 – Diagrama de Classes – Sistema de Controle Bancário



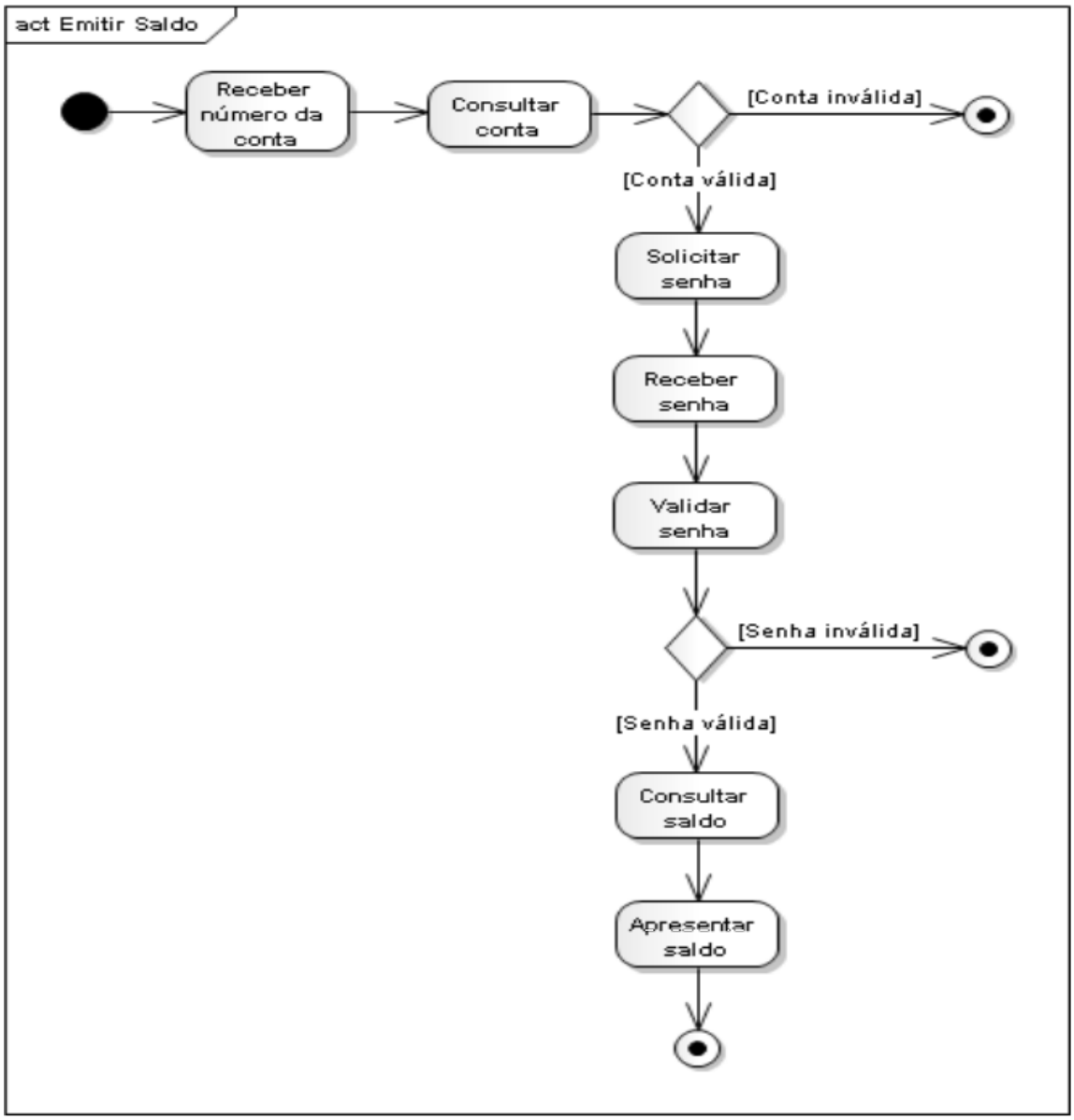
Fonte: Guedes (2011, p.138)

3.2.3. DIAGRAMA DE ATIVIDADES

O diagrama de atividades informa todo o processo que uma atividade persegue dês do início até o fim de uma operação, ele analisa todas as opções possíveis de uma atividade e quais resultados serão apresentados. Guedes (2011) diz que este diagrama modela mais do que uma atividade, ele abrange todas atividades do sistema indicando o processo de cada uma, que é composto por uma sequência de ações.

Segue o exemplo na Figura 7, para atividade “Emitir Saldo” do sistema de controle bancário.

Figura 7 – Diagrama de Atividade – Processo de Emissão de Saldo



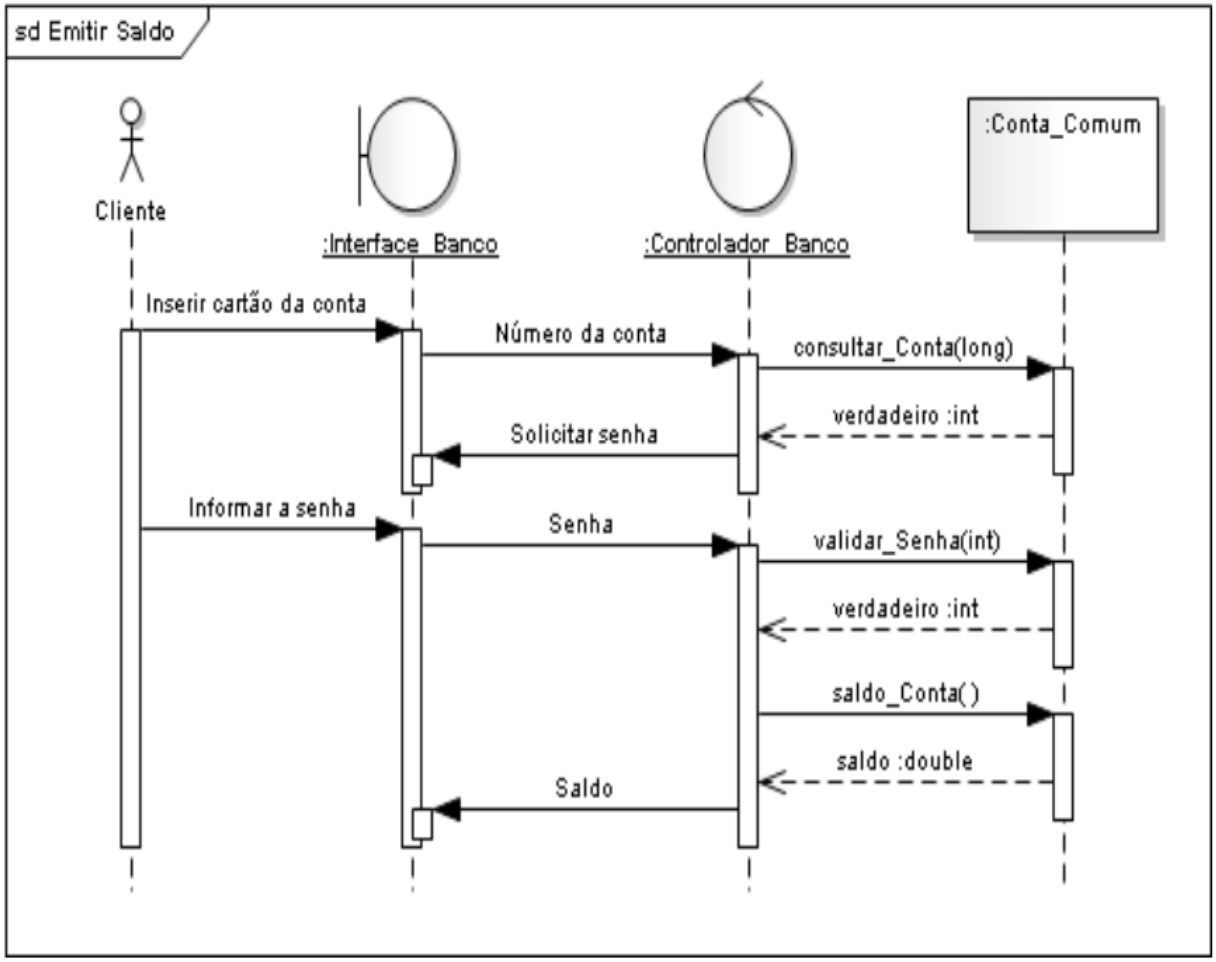
Fonte: Guedes (2011, p.290)

3.2.4. DIAGRAMA DE SEQUÊNCIA

De acordo com Guedes (2011), o diagrama de sequência é do tipo comportamental, ou seja, ele informa a sequência de atividades de um processo, determina a ordem em que eles acontecem, as mensagens enviadas, os métodos que são chamados e como os objetos se comunicam. O diagrama de sequência tem como base o diagrama de caso de uso, porém é mais especifico quanto as atividades, geralmente é usado um diagrama para cada processo usado por um ator.

Observa-se no exemplo da Figura 8 o processo que o cliente corre ao emitir o saldo num sistema de controle bancário.

Figura 8 – Diagrama de Sequência – Processo de Emissão de Saldo



Fonte: Guedes (2011, p.211)

3.3. TESTE E QUALIDADE

Pressman (2011) diz que a necessidade de qualidade na área de TI começou quando essas tecnologias estavam sendo utilizadas no cotidiano das organizações, em 1990, empresas começaram a perceber que bilhões de dólares estavam sendo perdidos por conta de erros e defeitos em seus sistemas. Nos tempos atuais ainda se vê essa dificuldade de garantir um software de qualidade, estes problemas ocorrem por conta de falhas em ambos os lados, tanto do cliente quanto ao desenvolvedor.

> Os defeitos de software que levam ao crash do programa são certamente bastantes inconvenientes [...] há outros fatores, como o preço, que não devem ser desprezados quando se busca determinar a qualidade. Erros de software já foram responsáveis por prejuízos milionários e mesma a perda de vidas humanas. A análise de falhas que tenham sido identificadas e documentadas abre a possibilidade para que sejam estudadas técnicas para evitar erros no futuro. (KOSCIANSKI; SOARES, 2007, p.33-34)

Delamaro (2007) acrescenta que o desenvolvimento de um sistema é de grande complexidade, ainda mais para um projeto de grande dimensão que contém vários detalhes, por conta disto pode ocorrer tantos problemas durante a construção que o produto final saia de uma forma diferente do esperado pelo cliente. Antes do software ser finalizado, são utilizadas várias atividades denominadas de Validação, Verificação e Teste (VV&T), essas atividades devem garantir que as especificações feitas pelo cliente estejam de acordo com o que foi desenvolvido.

Do ponto de vista de um analista de teste, detectar um defeito em um sistema é um ponto positivo, pois será menos um problema para o usuário, porém é importante enfatizar que a qualidade deve ser compreendida principalmente pelos desenvolvedores para que possa primeiramente evitar esse defeito. Diante disso, pode-se afirmar que a fase de Teste e Qualidade dentro da Engenharia de Software tem uma grande influência na funcionalidade do produto final.

4. WEB SERVICES

No final dos anos 90 começou a ser desenvolvido, em grande escala, sistemas que utilizavam HTTP e XML em suas tecnologias, porém cada empresa os manipulavam de sua maneira. Essa situação causou o início de várias complicações por conta da incompatibilidade entre os sistemas, por isso foi preciso criar uma

tecnologia para manter um padrão de linguagem que sejam entendidos por ambos os lados de um sistema. (MORO; DORNELES; REBONATTO, 2011)

Durães (2005) acrescenta que o Web Service nasceu com o objetivo de ajudar o mercado que tinha a necessidade de integração entre os diferentes ambientes existentes e que não podiam se comunicar por falta de um padrão comum. Os Web Services usam a internet e o XML para troca de informações, você aciona um método que dispara o processamento no servidor remoto que então retorna uma resposta no seu sistema, esse processo é feito por meio de uma leitura do WSDL (*Web Services Description Language), que é* um documento em XML onde contém o padrão do Web Service.

Rebonnato, Dornelles e Moro (2011) explica que esta tecnologia permite que aplicações e dispositivos diferentes interajam entrei si, o Web Service faz com que um sistema entenda e envie mensagens de diversos formatos, uma aplicação pode ter sua própria linguagem e enviar uma mensagem para outra com uma estrutura completamente diferente e ainda assim ela irá captar a mensagem que será traduzida para a linguagem universal, o XML.

De acordo com Fidel (2015), está cada vez mais comum em sistemas corporativos a necessidade entre sistemas se comunicarem através da internet, mesmo que seja para o uso de uma simples função, os Web Services são uma das tecnologias mais usadas na linguagem Java e fazem parte do cotidiano dos desenvolvedores. Existem protocolos para os Web Services que servem para definir o formato de dados usado para o envio de estrutura de dados por meio dos serviços, entre eles os mais usados são o SOAP (*Simple Object Access Protocol*) e o REST (*Representational State Transfer*), que serão detalhados mais afundo ao decorrer deste capitulo

4.1. DEFINIÇÃO DE SOAP E REST

Para Rozlog (2013), há uma grande variedade de tecnologias que um desenvolvedor pode escolher, desde simples ferramentas como um gerenciador de banco de dados a um medidor de volume sanguíneo, por conta disto, se torna difícil escolher uma abordagem especifica para a construção de um projeto. Se tratando de

Web Services, o protocolo SOAP e REST estão dentro desta disputa entre qual deve ser escolhido, apesar de as duas terem os mesmos objetivos, cada uma tem suas características distintas e são manejadas de formas diferentes.

> O protocolo SOAP [...] é uma especificação para a troca de informação entre sistemas, ou seja, uma especificação de formato de dados para envio de estruturas de dados entre serviços, com um padrão para permitir a interoperabilidade entre eles. Seu design parte do princípio da utilização de XMLs para a transferência de objetos entre aplicações, e a utilização, como transporte, do protocolo de rede HTTP. Os XMLs especificados pelo SOAP seguem um padrão definido dentro do protocolo. Esse padrão serve para que, a partir de um objeto, seja possível serializar o mesmo para XML e, também, deserializá-lo de volta para o formato original. (FIDEL, 2015)

O SOAP contém algumas funções adicionais não encontrada no REST, a principal é que o SOAP depende do XML em três maneiras: **O envelope**, define o conteúdo e como processa-lo, normalmente enviado por meio de HTTP; **regras de codificação**, definindo os tipos de dados e o **layout** usado nos procedimentos de chamadas e respostas. Após este processo é executado uma chamada de procedimento que retorna o XML formatado com as informações do documento. (Rozlog, 2013)

De acordo com Rebonnato, Dornelles e Moro (2011), REST é um termo que foi usado primeiramente por Roy Fielding, um dos criadores do HTTP. Este estilo de arquitetura não é exclusivo para o uso em *Web Services*, geralmente o REST é usado para caracterizar interfaces que transmita dados de um domino especifico sobre HTTP, sem usar uma camada adicional de mensagem ou "*cookies"* como utilizados pelo SOAP. Os sistemas que utilizam dos métodos REST também são denominados de "*RESTful*".

Rozlog (2013) acrescenta que a maioria dos desenvolvedores utilizam o REST para o gerenciamento do Web Service por fazer uso do padrão URI (*Uniform Resource Identifier*), que são caracteres usados para identificar um método ou recurso. REST é uma arquitetura simples, onde pode ser utilizado em vários tipos de situações ou

qualquer servidor com suporte HTTP. A principal vantagem para um desenvolvedor é o pequeno esforço necessário para dominar esta tecnologia.

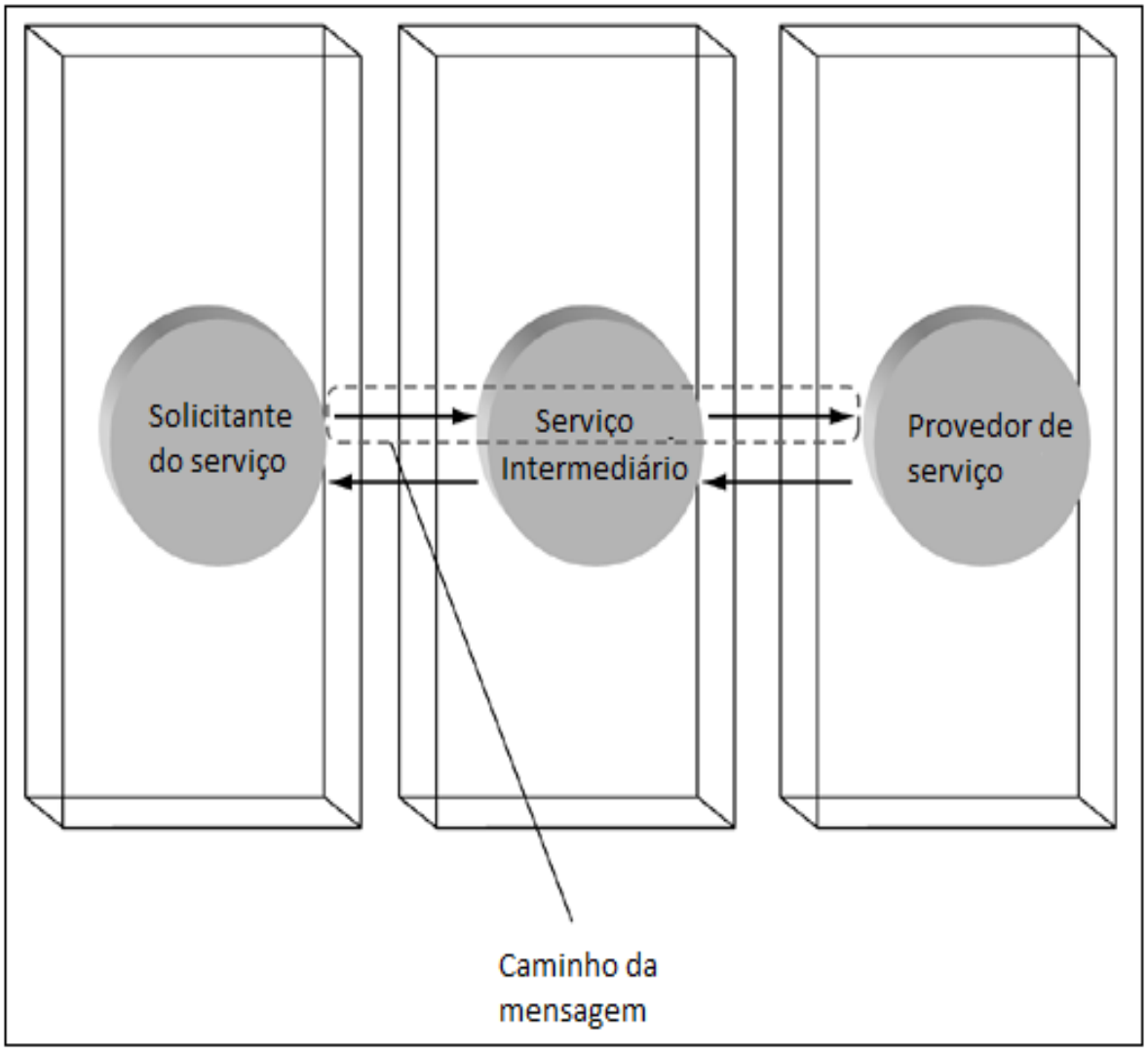
4.2. COMUNICAÇÃO ENTRE WEB SERVICES

Segundo ERL (2009), Os serviços podem se comportar de maneiras diferentes dependo da situação. Dependendo da tarefa em que o é designado, o mesmo *Web Service* pode ser apontado a fazer outra atividade ou até varias simultaneamente. Essas tarefas são denominadas em: **Provedor de serviços**, que também pode ser descrito como uma organização que provê o *Web Service*, ele pode assumir o papel tanto do solicitante quanto ao provedor de serviços, **Solicitante de serviço**, é o programa que aguarda uma mensagem resposta do Web Service, **Intermediário** que tem o papel de assumir uma mensagem e repassar para outro Web Service, **Remetente inicial** responsável por iniciar a transmissão da mensagem e o **Receptor final** que é simplesmente o ultimo Web Service a receber a mensagem.

O funcionamento começa com uma requisição, que é colocado num envelope SOAP onde é transmitido em HTTP para o *Web Service* alvo, o documento WSDL é usado para armazenar todos os dados que o serviço espera, logo a requisição que contém o envelope chega ao serviço que então o processa montando um outro envelope SOAP de resposta. Esta resposta é enviada para o consumidor do serviço, por fim o cliente abre o envelope retornado e analisa os dados. (REBONNATO; DORNELLES; MORO, 2011)

ERL (2009) diz que o caminho pelo qual a mensagem percorre deve se constituir de um remetente inicial e um remetente final e pode também conter nenhum ou vários intermediários, como ilustrado na Figura 9.

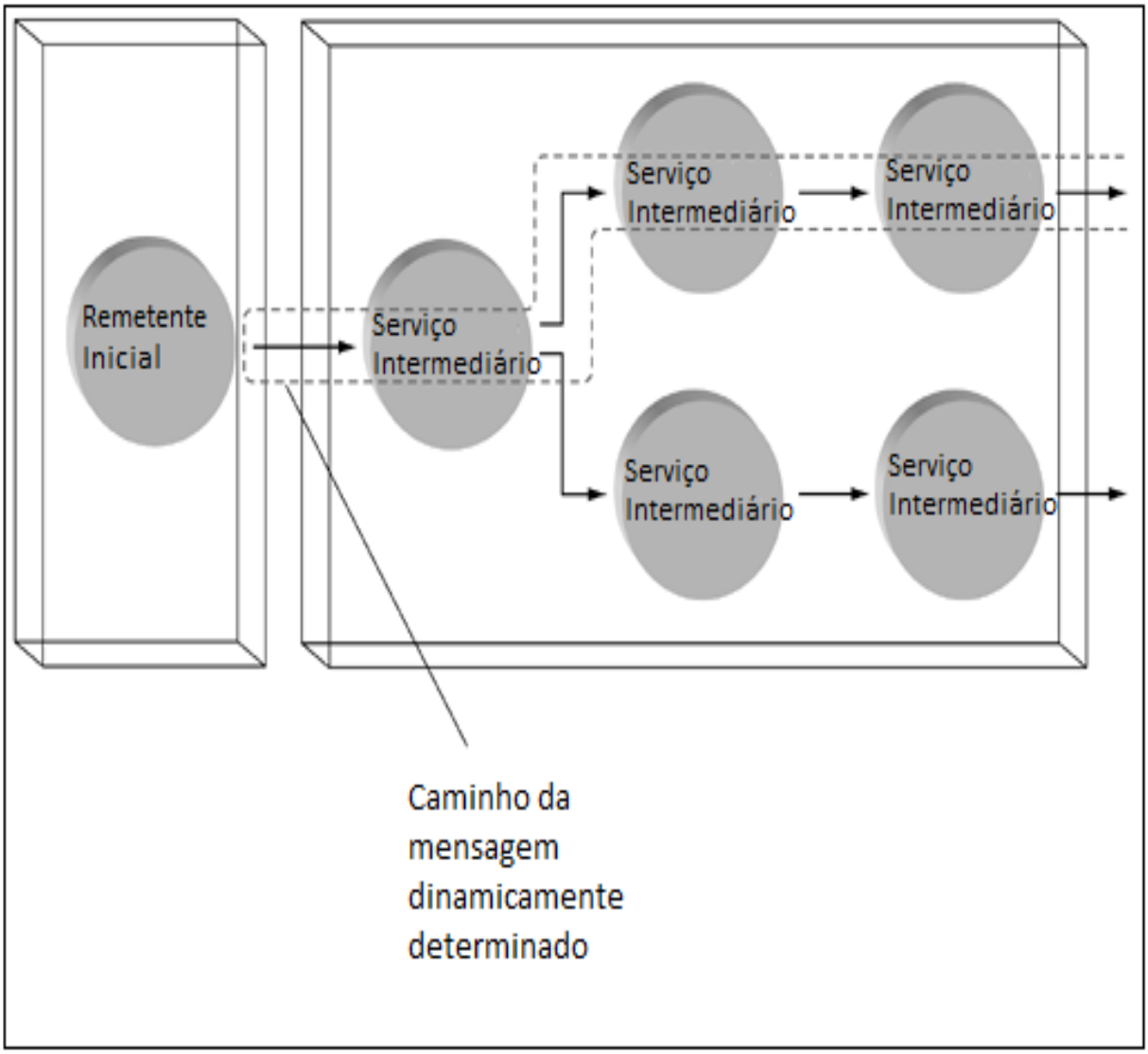
Figura 9 – Mensagem enviada através do Web Service



Fonte: Erl (2009)

Os Intermediários podem definir o caminho que uma mensagem irá percorrer, essa lógica de roteamento de mensagens serve para organizar os fluxos em tempo de execução do sistema, na Figura 10 é detalhado o fluxo de quando uma mensagem é enviada via dois caminhos de mensagens, onde o rumo é determinado pelo roteador intermediário (ERL, 2009).

Figura 10 – Roteador intermediário



Fonte: Erl (2009)

5. ESTUDO DE CASO

Este estudo de caso será utilizado para apresentar as metodologias e funcionalidades implementadas baseado nas propostas do projeto. Será evidenciado aqui o funcionamento do sistema em vários aspectos diferentes, apresentando os diagramas da linguagem de modelagem unificada e a interface do aplicativo.

Este projeto foi desenvolvido considerando o alto volume de carros dentro do centro da cidade de Americana, foi observado dificuldades quanto ao tempo gasto na procura de uma vaga de estacionamento, levando isso em conta foi proposto um aplicativo capaz de listar comércios de estacionamento mais próximos ao usuário, informando também os preços e as avaliações de outros que utilizaram o mesmo.

Os conhecimentos adquiridos durante a pesquisa teórica tornaram possível o desenvolvimento do sistema. Durante este capitulo, será demonstrado como foi feito a implementação desses conceitos examinado no conteúdo teórico.

5.1. LINGUAGEM DE MODELAGEM UNIFICADA

Os diagramas a serem apresentados a seguir tem como objetivo descrever o funcionamento do sistema com a finalidade de prover um melhor entendimento do aplicativo ao ponto de vista dum cliente, expondo também, os detalhes de como o usuário poderá interagir de formas diferentes aos eventos ocorridos.

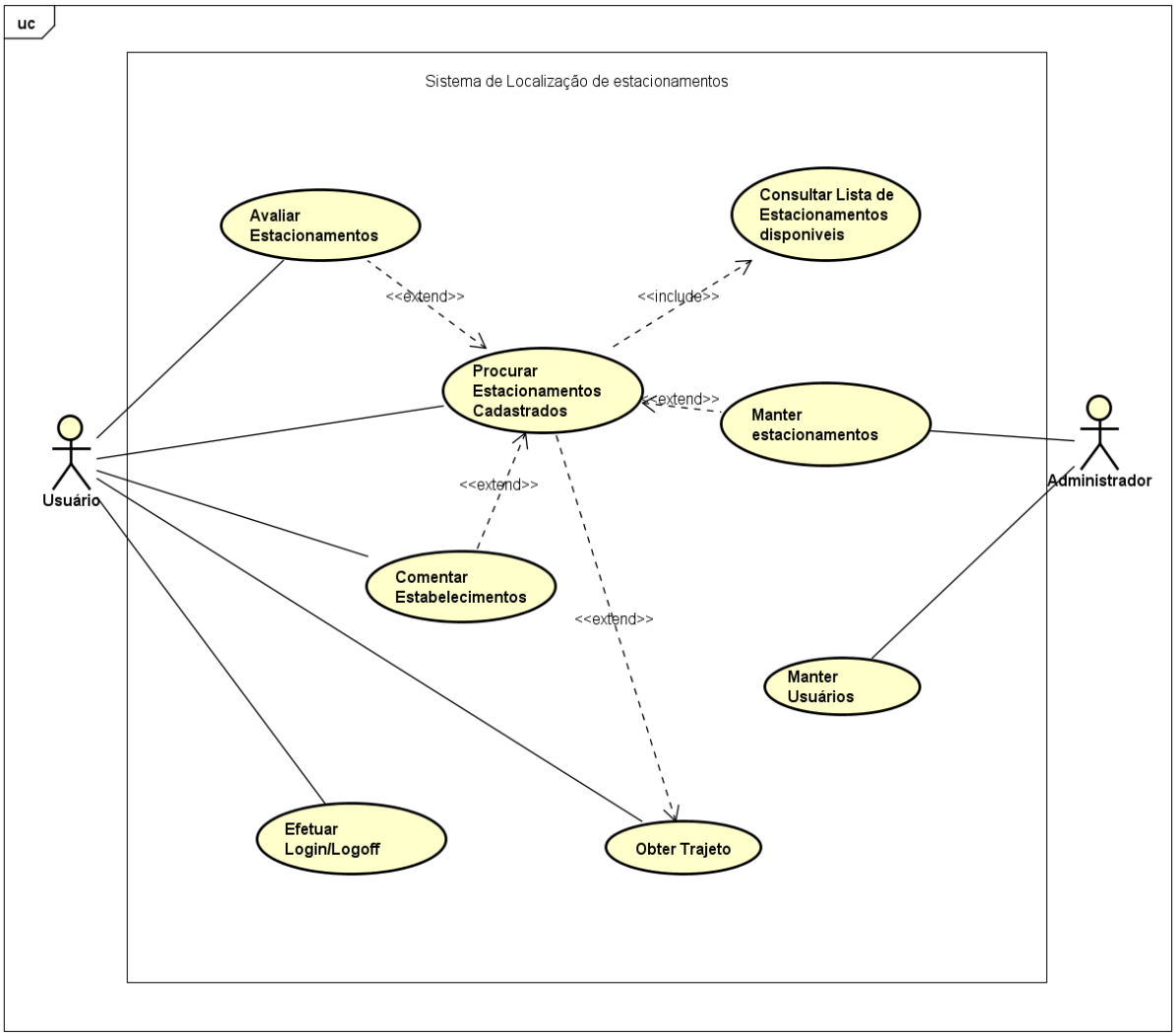
Serão representadas as partes essenciais do projeto e como cada ator se comunica com as diversas funcionalidades do mesmo. Como visto anteriormente nos conceitos abordados em Engenharia de Software, o UML se divide em vários segmentos, onde serão abordados aqui o Diagrama de classe, Diagrama de caso de uso, Diagrama de sequência e o Diagrama de atividades.

5.1.1. DIAGRAMA DE CASO DE USO

Observa-se na Figura 11 o sistema retratado em uma visão geral, este diagrama representa o papel do usuário e o administrador do sistema que terão funções

diferentes. O administrador poderá gerenciar os dados do usuário e também adicionar novos estacionamentos no banco de dados. O usuário exerce ações simples como procurar, comentar ou avaliar estacionamentos.

Figura 11 – Diagrama de Caso de Uso

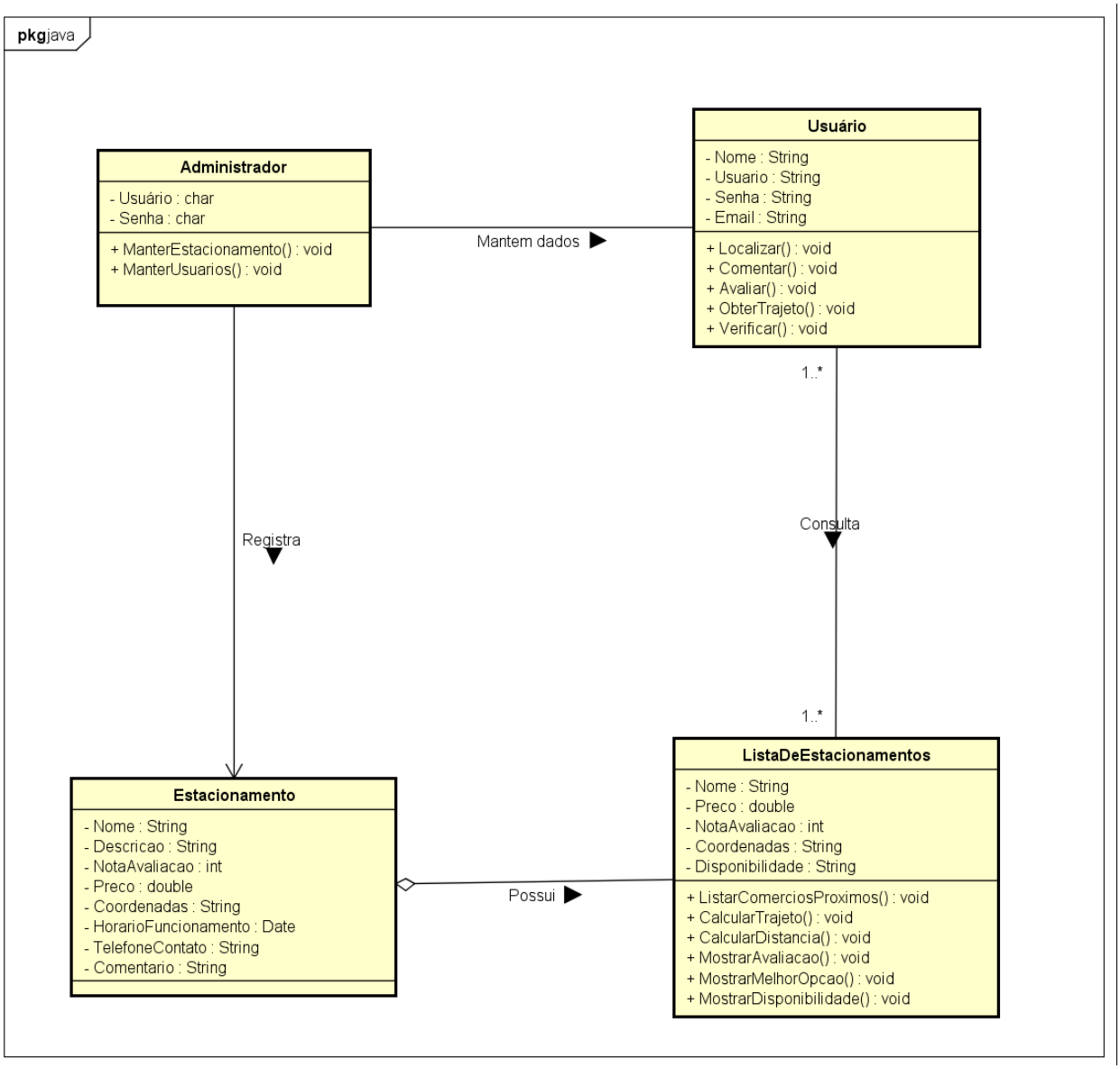


Fonte: Próprio autor

5.1.2. DIAGRAMA DE CLASSE

Este diagrama representa as classes usadas no desenvolvimento do sistema e como se interagem entre si, como pode se observar na Figura 12, a classe “Usuário” pode consultar a outra classe “Lista De Estacionamentos” que herda dados de “Estacionamentos”, estes dados são mantidos pelo Administrador.

Figura 12 – Diagrama de Classe

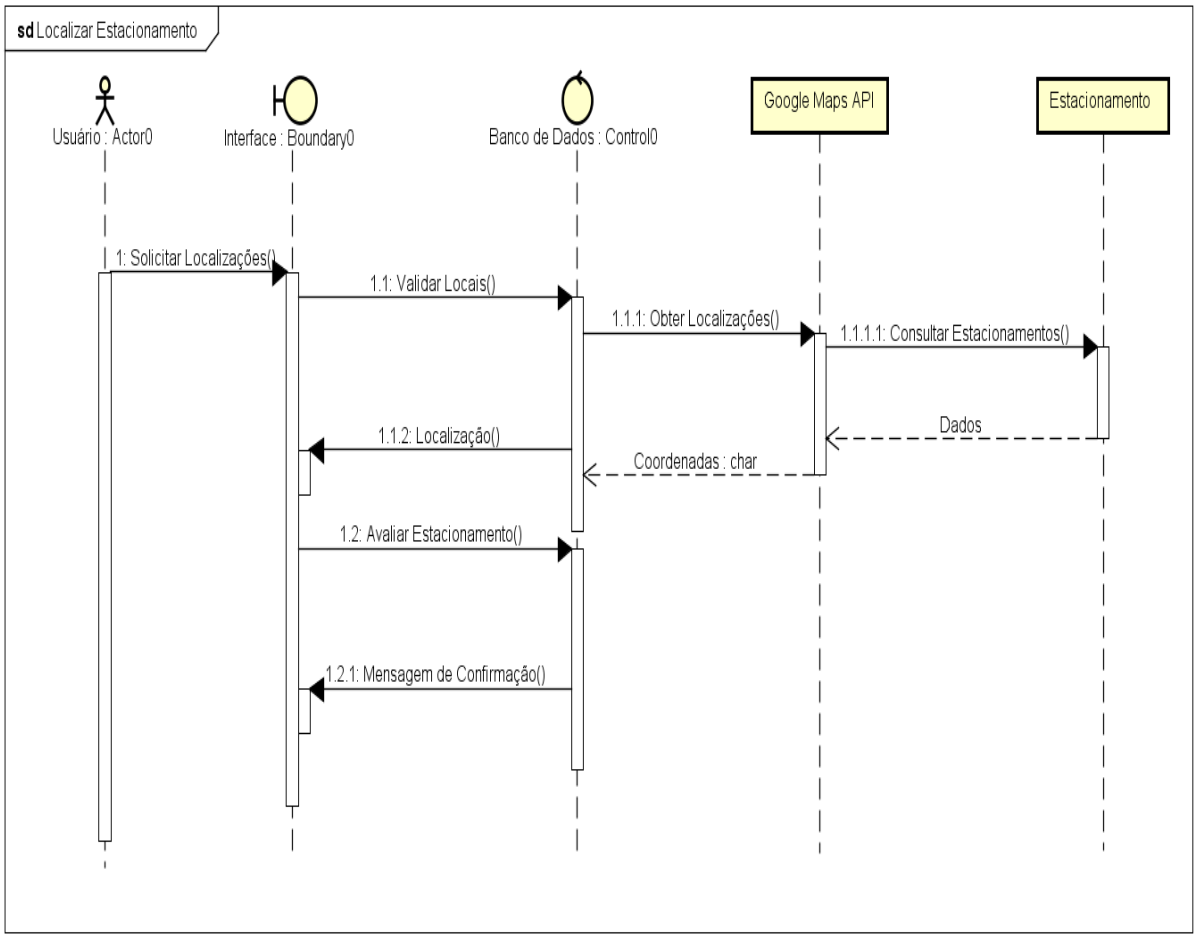


Fonte: Próprio autor

5.1.3. DIAGRAMA DE SEQUÊNCIA

No Diagrama de Sequência é relatado o caminho das ações de um ator, divididos nas atividades principais: Localizar Estacionamento, Manter usuário e Cadastrar Estacionamento. A figura 13 demonstra a passagem do Usuário ao optar por localizar um estacionamento, ele primeiramente irá solicitar o processo de localização, que por si irá acionar o banco de dados do *Google Maps*, retornando as localizações disponíveis, podendo, ao final, avaliar o estacionamento.

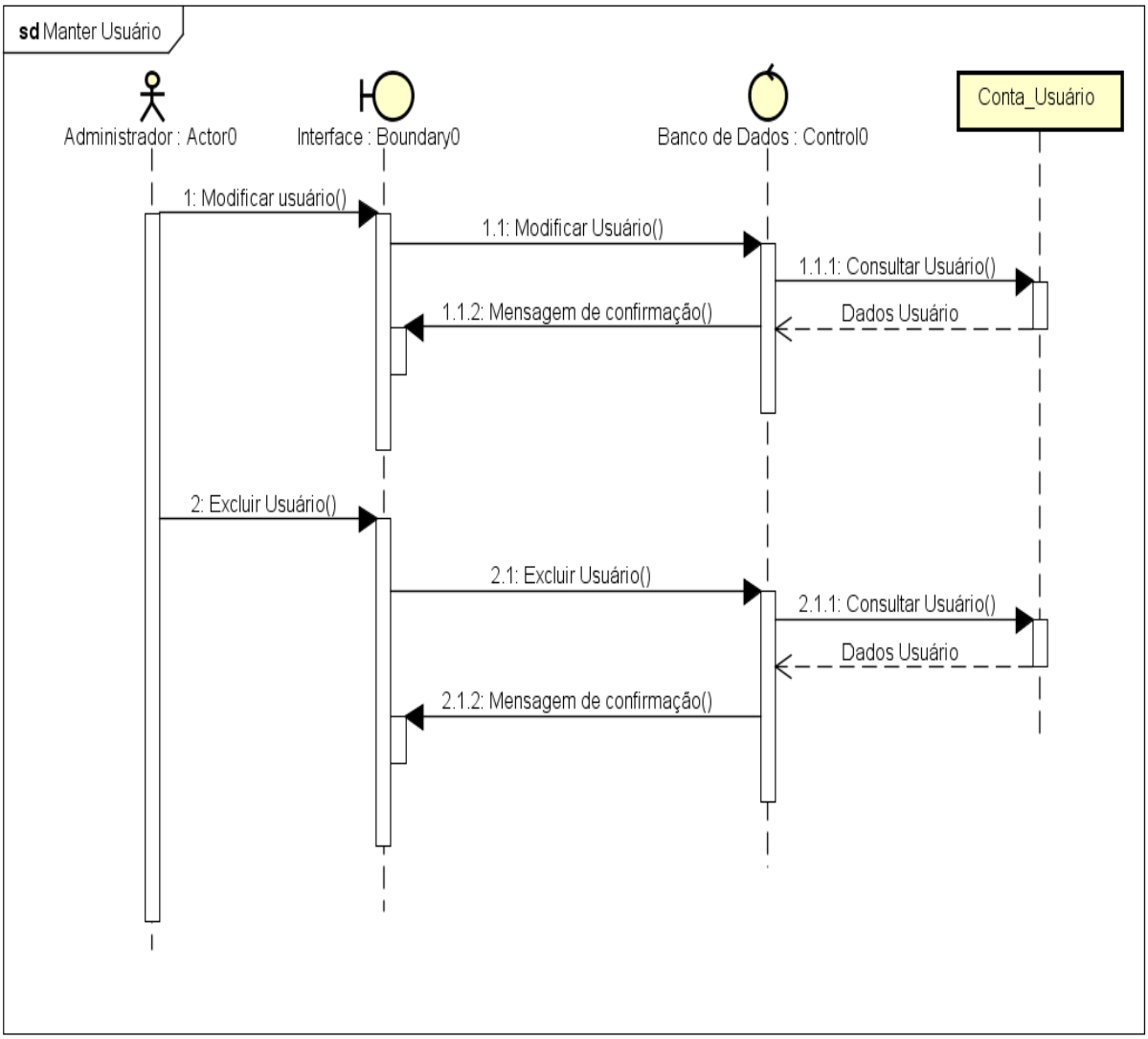
Figura 13 - Diagrama de Sequência: Localizar Estacionamento



Fonte: Próprio autor

O diagrama apresentado na Figura 14 indica a sequencia de ações exercido pelo Administrador ao manter os dados usuário, que terá a opção de modificar ou excluir, usando a função do sistema que irá efetuar a mudança requisitada no banco de dados, retornando uma mensagme de confirmação com os dados modificados.

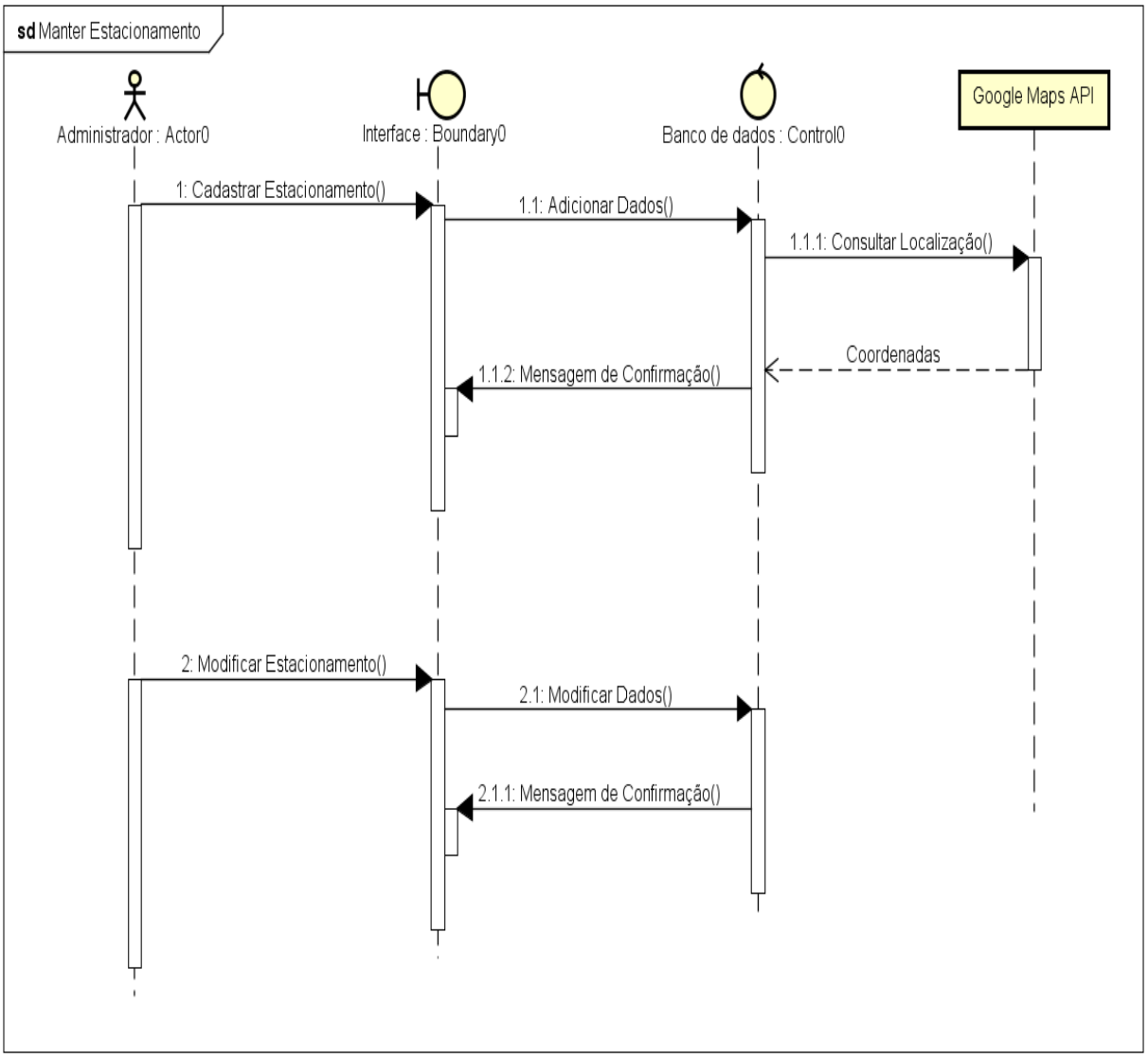
Figura 14 - Diagrama de Sequência: Manter Usuário



Fonte: Próprio autor

Como visto a seguir na Figura 15, a ilustração do diagrama de manter estacionamento é similar ao de manter usuário, porém este se aplica aos estacionamentos, onde o Administrador poderá cadastrar novos estabelecimentos, fazendo que o sistema consulte o banco de dados do *Google Maps* para adquirir as coordenadas a serem cadastradas ou modificar dados de um já existente.

Figura 15 - Diagrama de Sequência: Manter Estacionamento

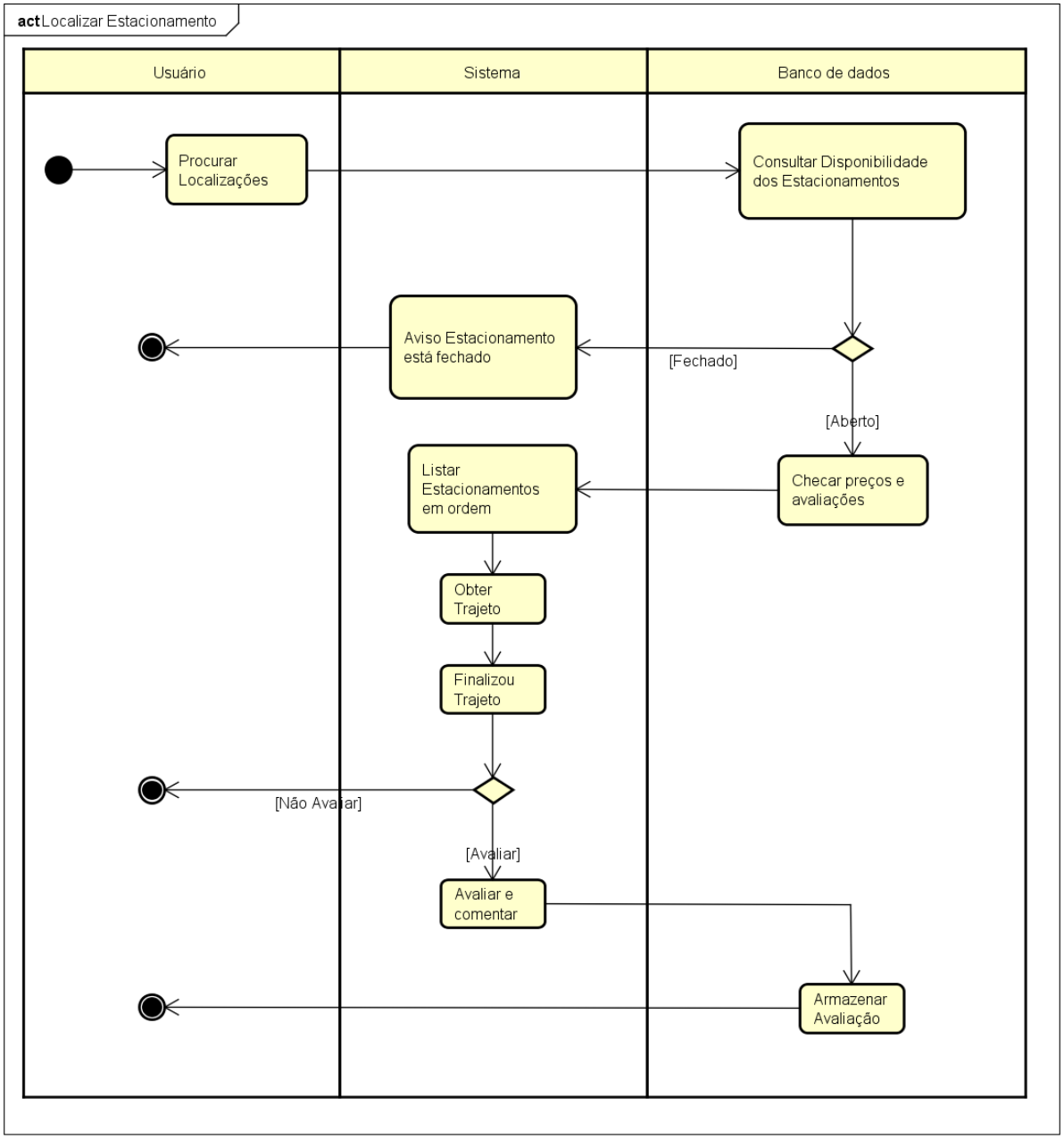


Fonte: Próprio autor

5.1.4. DIAGRAMA DE ATIVIDADES

O diagrama de atividades irá descrever os possiveis eventos que ocorrerão até o destino final das atividade básicas Localizar Estacionamento e Manter Usuário . A Figura 16 relata a atividade de localizar estacionamento e os ações que podem ocorrer ao usuário.

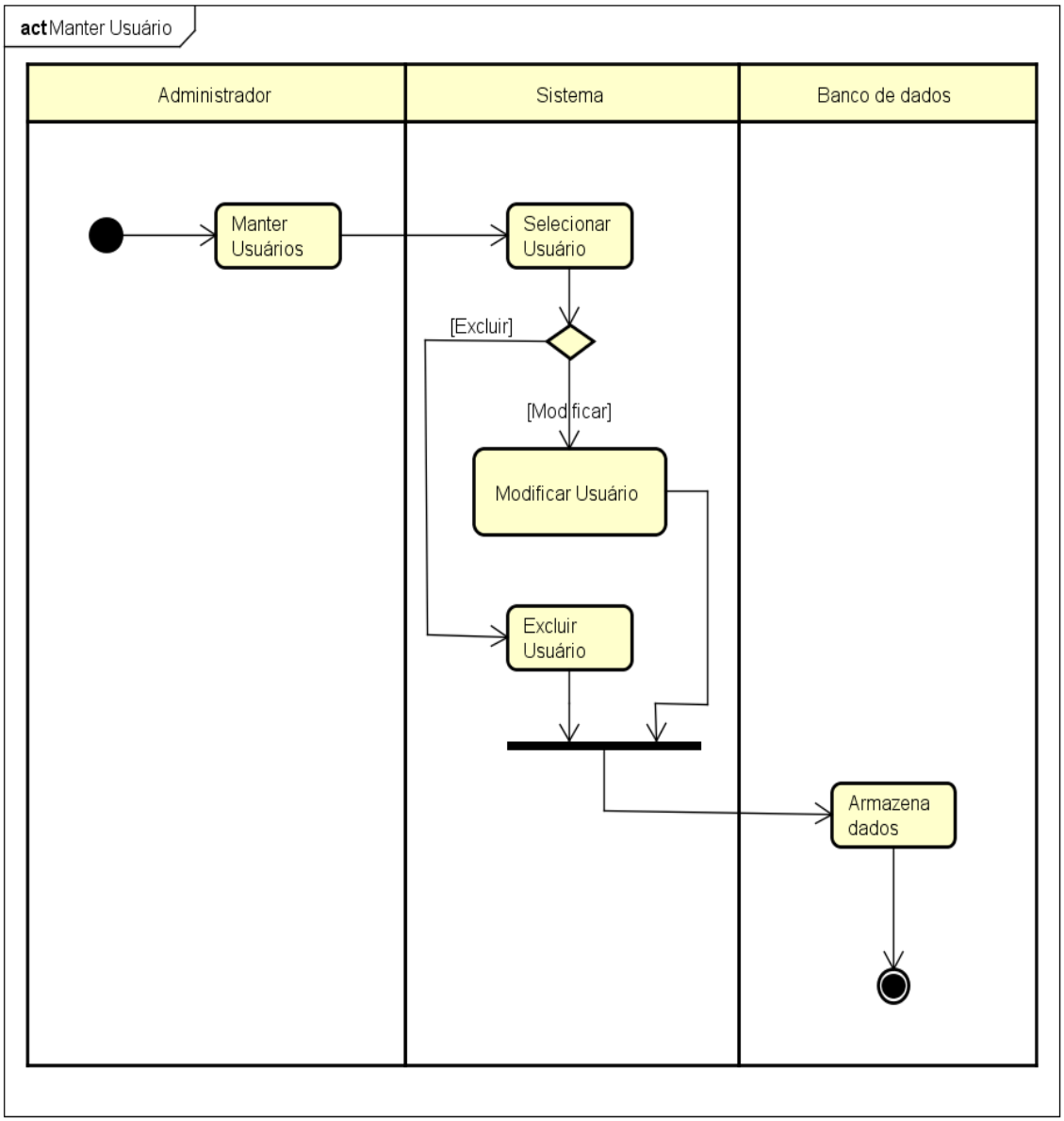
Figura 16 – Diagrama de Atividades: Localizar Estacionamento



Fonte: Próprio autor

Observa-se na Figura 17 que o Administrador ao manter os dados do usuário terá a opção de modificar ou excluir uma conta existente, a seguir será passado as informações para o banco de dados armazenar as mudanças solicitadas, por fim a atividade será finalizada.

Figura 17 - Diagrama de Atividades: Manter Usuário



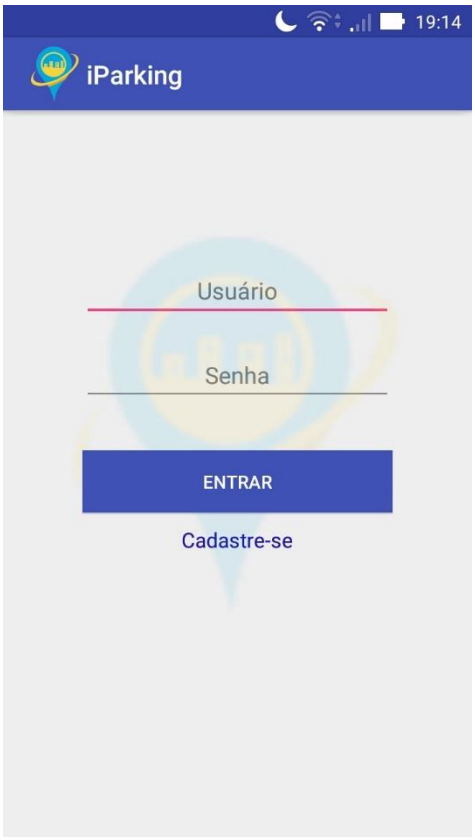
Fonte: Próprio autor

5.2.1. INTERFACE GRÁFICA DA APLICAÇÃO

Neste capitulo será apresentado as telas do aplicativo e explicado suas funções, essencialmente este sistema envolve dois tipos de usuários, o usuário comum e o administrador, onde cada um tem suas funcionalidades individuais.

Ao iniciar o sistema, tanto o usuário comum quanto o administrador se deparam com a tela de *login*, representada na Figura 18.

Figura 18 – Tela de Login de Acesso



Fonte: Próprio autor

Na tela de Login terá a opção de entrar com uma conta e caso o usuário não tenha cadastro ele poderá criar um, observa-se na Figura 19 a tela que contém os campos a serem inseridos com os dados do usuário.

Figura 19 – Tela de Cadastro de Usuário

Fonte: Próprio autor

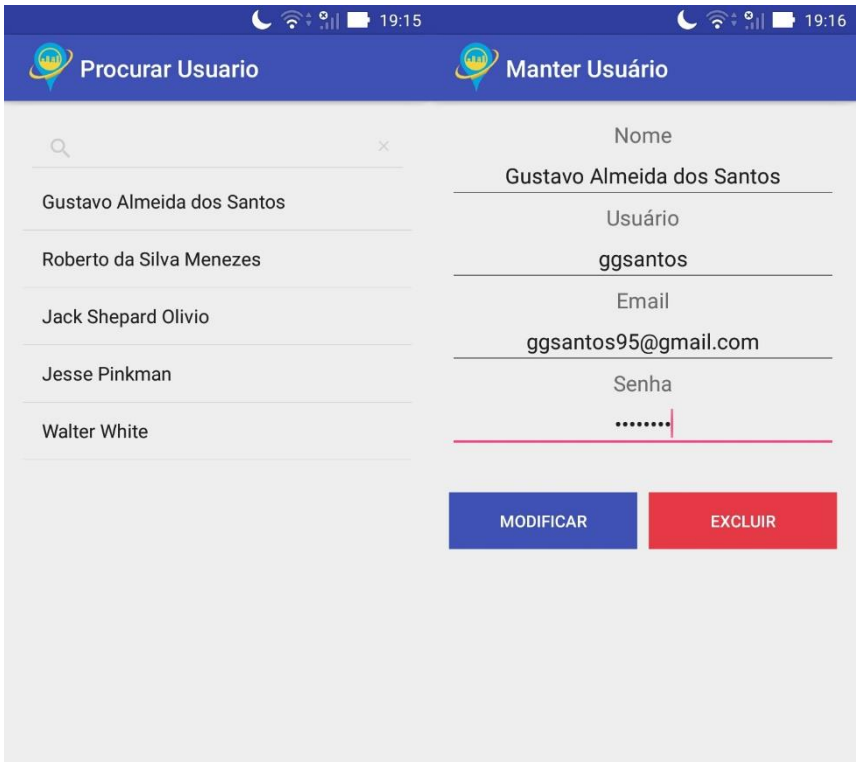
Ainda na tela de login, caso o login feito seja de um administrador então o sistema o reconhecerá e irá direto para a tela de menu dos administradores, a Figura 20 representa as opções do administrador.

Figura 20 – Tela Menu do Administrador

Fonte: Próprio autor

Ao escolher a primeira opção, será aberto uma tela para procurar um usuário existe para que possa modifica-lo, na Figura 21 é representado a lista com todos os usuários existentes, também sera possivel procurar alguem pelo nome que estará cadastrado no banco de dados. Ao selecionar o usuário escolhido, será aberto outra tela com os dados puchados do banco de dados, onde será possivel modifica-lo ou até mesmo exclui-lo.

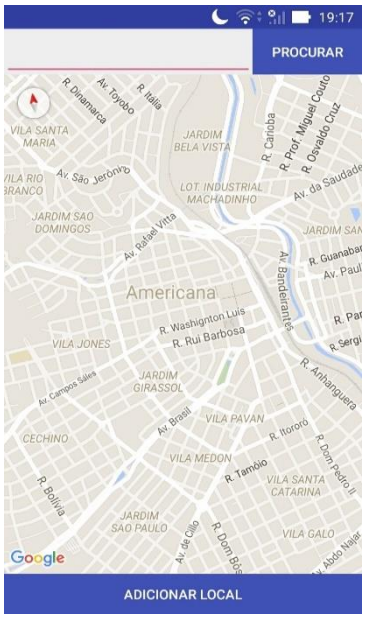
Figura 21 – Tela Manter Usuário



Fonte: Próprio autor

Uma tela muito importante é a de Adicionar Estacionamento onde se encontra na segunda opção do Administrador, ele poderá adicionar um novo local de estacionamento, como exibido na Figura 22, será aberto o mapa Google Maps onde será possível navegar sob a cidade e obter as coordenadas para uma posição geográfica onde se encontra o estacionamento.

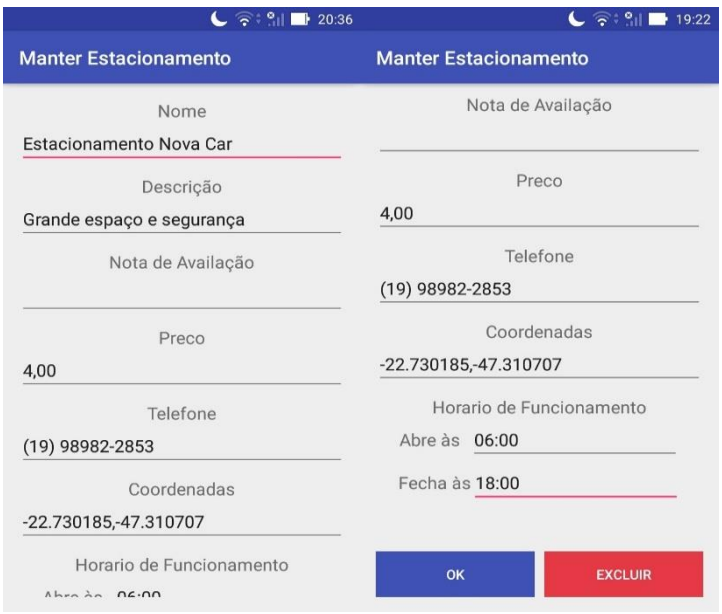
Figura 22 – Tela Adicionar Local



Fonte: Próprio autor

Após apontar para a localização a ser adicionada, será aberta a tela de Manter Estacionamento com as coordenadas já preenchidas, como observado na Figura 23, onde será adicionada outras informações do estabelecimento como o nome, preço, telefone, etc....

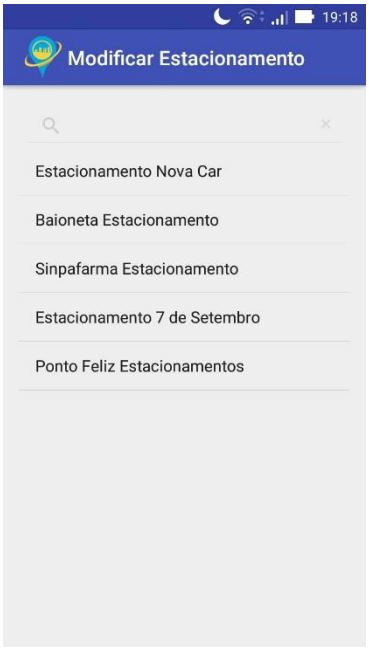
Figura 23 – Tela Manter Estacionamento



Fonte: Próprio autor

Já na terceira opção no menu do Administrador, é apresentado a tela Modificar Estacionamento que tem a função de listar todos os nomes dos estacionamentos já cadastrados, é possível escolher um deles, fazendo que redirecione para a mesma tela de Manter Estacionamento mostrado na Figura 23 com os dados puxados do estacionamento escolhido. A Figura 24 representa a lista de estacionamentos.

Figura 24 – Tela Modificar Estacionamento



Fonte: Próprio autor

O objetivo principal do *login* de Administrador é que ele possa gerenciar os estacionamentos sem que tenha acesso direto ao banco de dados como o desenvolvedor, podendo assim ser disponibilizados outros Administradores para que seja possível cadastrar estabelecimentos diferentes em diversas cidades.

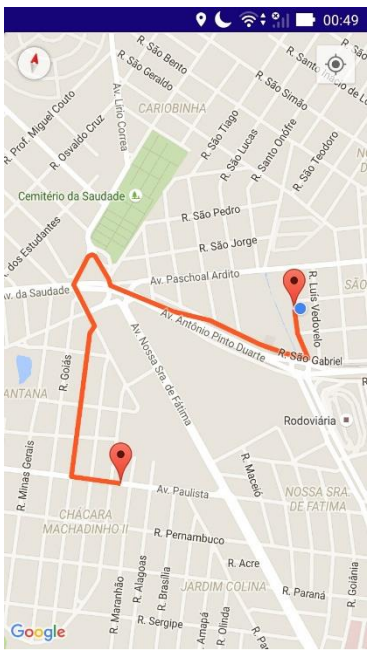
Quando o acesso é feito por um usuário normal na Tela de Login de Acesso, ele é levado diretamente para a Lista de Estacionamentos, aqui é listado todos os estacionamentos cadastrado pelo Administrador, observa-se na Figura 25 que cada opção apresenta o nome do estabelecimento, a distância entre o local atual até o local do estacionamento e sua nota.

Figura 25 – Tela Lista de Estacionamentos

Fonte: Próprio autor

Após escolher um estacionamento, será aberto o mapa do Google Maps representado na Figura 26, contento a rota do seu ponto atual até o estabelecimento preferido.

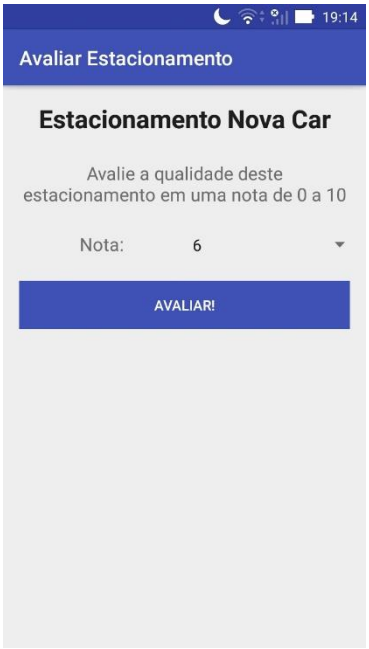
Figura 26 – Tela de Rotas



Fonte: Próprio autor

Ao chegar no destino estipulado será aberto a tela Avaliar Estacionamento representada na Figura 27, onde o usuário poderá julgar a qualidade do estacionamento que usou.

Figura 27 – Tela Avaliar Estacionamento



Fonte: Próprio autor

6. CONCLUSÃO

Diante dos estudos realizados, levando em conta as dificuldades observadas anteriormente, conclui-se que o sistema poderá ajudar seus usuários a localizarem um comercio de estacionamento ideal a sua escolha e podendo juntamente avalia-los. Através das pesquisas realizadas principalmente se tratando do sistema operacional *Android, Web Services, Google Maps* e os processos da engenharia de software, foi possível construir um protótipo de aplicativo móvel capaz de proporcionar informações ao usuário, permitindo-o uma fácil visualização geográfica de seu destino.

6.1. TRABALHOS FUTUROS

Na realização de trabalhos futuros, o projeto poderá se expandir para outras cidades, onde para cada uma, serão disponibilizadas contas de administradores, responsáveis pelo gerenciamento dos locais de estacionamentos, essas contas serão providenciadas pelo desenvolvedor.

Também se tem em vista implementar a funcionalidade de cada proprietário dum estacionamento inserir seu próprio comércio na lista de estacionamentos disponíveis, visando uma aplicação mais dinâmica e com mais opções para o usuário.

7. REFERÊNCIAS

BORDIN, Maycon Viana. **Introdução a Arquitetura Android**. Disponível em: <http://sites.setrem.com.br/stin/2012/anais/Maycon.pdf>. Acesso em: 05 Maio. 2016.

DARWIN, Ian F.. **Android Cookbook.** São Paulo: Novatec Editora Ltda, 2012.

LECHETA, Ricardo R.. **Google Android:** Aprenda a criar aplicações para dispositivos móveis com o Android SDK. 5. ed. São Paulo: Novatec Editora Ltda, 2015.

MONTEIRO, João Bosco. **Google Android:** Crie aplicações para celulares e tablets. Campo Grande, MS: Casa do Código, 2013.

PRESSMAN, Roger S. **Engenharia de *software:*** Uma abordagem profissional. Traduzido por Ariovaldo Griosi. 7ªed. Porto Alegre: AMGH, 2011.

ANDROID DEVELOPERS. 2016. Disponível em: <http://developer.android.com/about/dashboards/index.html>. Acesso em: 27 Março. 2016.

TURINI, Rodrigo. **EXPLORANDO APIS E BIBLIOTECAS JAVA**: JDBC, IO, Threads, Java FX e mais. Casa do Código, 2015.

Google I/O. ***Android Anatomy and Physiology***. Maio, 2008. Disponível em: <http://androidteam.googlecode.com/files/Anatomy-Physiology-of-an-Android.pdf>. Acesso em: 27 Março. 2016.

CHAHOUD, Juliana. ***Estratégias de monetização de aplicativos móveis*****.** Setembro, 2014. Disponível em: <http://pt.slideshare.net/julianachahoud/estratgias-de-modelizao-de-aplicativos-mveis-39217188>. Acesso em: 18 Maio. 2016.

ENGHOLM JUNIOR, Hélio. **Engenharia de *software* na prática**. São Paulo: Novatec Editora Ltda, 2010.

SOMMERVILLE, Ian. **ENGENHARIA DE SOFTWARE**. Traduzido por Selma Shin Shimizu Melnikoff, Reginaldo Arakaki, Edílson de Andrade Barbosa. 8ªed. São Paulo: Pearson Education do Brasil, 2007.

GUEDES, Gilleanes T.A. **UML 2:** Uma abordagem prática. 2. Ed. São Paulo: Novatec Editora Ltda, 2011.

ROUSE, Margaret. ***Framework***. Fevereiro, 2015. Disponível em: <http://whatis.techtarget.com/definition/framework>. Acesso em: 19 Maio. 2016.

CARVALHO, Suelen. ***Android Studio:*** vantagens e desvantagens com relação ao Eclipse. Dezembro, 2013. iMasters. Disponível em: <http://imasters.com.br/mobile/android/android-studio-vantagens-e-desvantagens-com-relacao-ao-eclipse>. Acesso em: 19 Maio. 2016.

KOSCIANSKI, André; SOARES, Michel dos Santos. **Qualidade de Software:** Aprenda as metodologias e técnicas mais modernas para o desenvolvimento de software. 2ªed. São Paulo: Novatec Editora Ltda, 2007.

DELAMARO, Márcio E. et al. **Introdução ao Teste de Software**. 2ªed. São Paulo: Novatec Editora Ltda, 2007.

MORO, Tharcis Dal; DORNELES, Carina F.; REBONATTO, Marcelo Trindade. Web services WS-* versus Web Services REST. Revista de Iniciação Cientifica, v. 11, n. 1, março, 2011.

DURÃES, Ramon. **Web Services para iniciantes**. Setembro, 2005. iMasters. Disponível em: <http://imasters.com.br/artigo/3561/web-services/web-services-para-iniciantes>. Acesso em: 22 Maio. 2016.

FIDEL, Brunno. **Web Services REST versus SOAP**. 15 Abr, 2015. Devmedia. Disponível em: <http://www.devmedia.com.br/web-services-rest-versus-soap/32451#>. Acesso em: 22 Maio. 2016.

ERL, Thomas. **Introdução às tecnologias Web Services:** SOA, SOAP, WSDL e UDDI – Parte 1. 05 Ago, 2009. Devmedia. Disponível em: <http://www.devmedia.com.br/introducao-as-tecnologias-web-services-soa-soap-wsdl-e-uddi-parte1/2873>. Acesso em: 22 Maio. 2016.

ROZLOG, Mike. **REST e SOAP**: Usar um dos dois ou ambos?. Traduzido por Marcelo Costa. 02 Out, 2013. InfoQ. Disponível em: <https://www.infoq.com/br/articles/rest-soap-when-to-use-each>. Acesso em: 22 Maio. 2016.

CARVALHO, Gustavo. **Como publicar no Google Play**. 1 Maio 2012. Tutoriandroid. Disponível em: <http://www.tutoriandroid.com/2012/05/como-publicar-no-google-play.html>. Acesso em: 26 Maio. 2016.