UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
DEPARTAMENTO ACADÊMICO DE INFORMÁTICA
ESPECIALIZAÇÃO EM DESENVOLVIMENTO PARA DISPOSITIVOS MÓVEIS

RODRIGO CASAGRANDE DE JESUS

APLICATIVO MÓVEL PARA LOCALIZAÇÃO DE ESTACIONAMENTOS EM CURITIBA

MONOGRAFIA DE ESPECIALIZAÇÃO

CURITIBA
2017

RODRIGO CASAGRANDE DE JESUS

APLICATIVO MÓVEL PARA LOCALIZAÇÃO DE ESTACIONAMENTOS EM CURITIBA

Monografia de Especialização apresentada ao Departamento Acadêmico de Informática, da Universidade Tecnológica Federal do Paraná como requisito parcial para obtenção do título de Especialista em Desenvolvimento para Dispositivos móveis.

Orientadora: Profa. Dra. Maria Claudia Figueiredo Pereira Emer

CURITIBA
2017

Ministério da Educação
UNIVERSIDADE TECNOLÓGICA FEDERAL DO PARANÁ
Câmpus Curitiba
Diretoria de Pesquisa e Pós-Graduação
Departamento Acadêmico de Informática
Coordenação do Curso de Especialização em Desenvolvimento para Dispositivos Móveis

TERMO DE APROVAÇÃO

"Aplicativo Móvel para Localização de Estacionamentos em Curitiba"

por

"Rodrigo Casagrande de Jesus"

Este Trabalho de Conclusão de Curso foi apresentado às 18:45 do dia 18 de dezembro de 2017 na sala B201 como requisito parcial à obtenção do grau de Especialista em Desenvolvimento para Dispositivos Móveis na Universidade Tecnológica Federal do Paraná - UTFPR - Campus Curitiba. O(a) aluno(a) foi arguido pela Banca de Avaliação abaixo assinados. Após deliberação, a Banca de Avaliação considerou o trabalho aprovado.

_____ Profa. Maria Claudia Figueiredo Pereira Emer (Presidente/Orientadora - UTFPR/Curitiba)	_____ Prof. Robson Ribeiro Linhares (Avaliador 1 – UTFPR/Curitiba)
_____ Prof. Marco Aurélio Wehrmeister (Avaliador 2 – UTFPR/Curitiba)	

"A Ata de Aprovação assinada encontra-se na Coordenação do Curso."

RESUMO

Jesus, Rodrigo C. de. Desenvolvimento de aplicação móvel para localização de estacionamentos em Curitiba. 2017. 63 f. Monografia (Especialização em Desenvolvimento para Dispositivos Móveis) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

O estacionamento em grandes centros urbanos, mais especificamente Curitiba, pode ser considerado um desafio, quando se leva em consideração a otimização do tempo, as formas de pagamento, a aceitação de bicicletas, o oferecimento de manobrista e a permissão do próprio dono do veículo de estacionar e levar a chave. Este trabalho se propõe a desenvolver uma aplicação na plataforma Android que auxilia as pessoas da capital paranaense na busca por estacionamento, oferecendo uma listagem no mapa de locais disponíveis nos arredores da localização do dispositivo móvel. Além disso, permite ao usuário fazer busca personalizada por meio da configuração de filtros disponíveis. O aplicativo foi comparado com outras soluções do mercado e avaliado por pessoas de diferentes idades e afinidades com tecnologia e o resultado preliminar mostrou que a proposta foi bem aceita, mas o sistema ainda precisa de melhorias para apresentar funcionalidades competitivas com as opções encontradas no mercado.

Palavras-chave: Android. Estacionamentos. Aplicativo móvel. Curitiba.

ABSTRACT

Jesus, Rodrigo C. de. Development of mobile application to locate parking lots in Curitiba. 2017. 63 f. Monografia (Especialização em Desenvolvimento para Dispositivos Móveis) – Departamento Acadêmico de Informática, Universidade Tecnológica Federal do Paraná. Curitiba, 2017.

The act of parking in big urban centers, more specific in Curitiba, might be considered a challenge, in the aspect of optimization of time, payments forms, bike acceptance, valet parking and the possibility of the vehicle owner to park and carry the keys. This paper proposes to develop an Android application that helps the people of the paranaense capital in the search for parking lots, offering on the devices a list of available places on the map of the surroundings. Besides that, it allows the user to make personalized searches through available filters configuration. The app was compared to other solutions available on the market and evaluated by people of different ages and affinities with technology, and the preliminary results showed that the proposal was well accepted, mas the system still needs improvements to produce competitive functionalities with the other options found on the market.

Key-words: Android, parking lots, mobile application, Curitiba.

LISTA DE FIGURAS

LISTA DE TABELAS

SUMÁRIO

1 INTRODUÇÃO

No Brasil, o principal meio de transporte utilizado é o rodoviário, fato que se originou na década de 50, quando o governo de Juscelino Kubitschek optou – pela facilidade de implantação de rodovias, pelo preço do petróleo na época e para dar um novo padrão de renda para as classes médias e altas da economia – por abrir o mercado brasileiro para a indústria automobilística introduzir no mercado nacional uma variedade de veículos automotores (FUNDAÇÃO DINARCO REIS, 2017).

Em grandes cidades, inclusive pela concentração de pessoas da classe trabalhadora em áreas mais afastadas das zonas mais centrais e nobres, que tendem a concentrar a oferta de postos de trabalho (FUNDAÇÃO DINARCO REIS, 2017), o principal meio de transporte utilizado é o público. No entanto, os meios de transporte automóveis, como os carros e motos, somaram 29,57% como principais meios de locomoção utilizados pelas pessoas em regiões metropolitanas brasileiras no ano de 2010 (IPEA, 2017).

Para distribuir equitativamente a quantidade de cidadãos que conseguem utilizar os estacionamentos de vias públicas centrais, surgiu o sistema de estacionamentos públicos rotativos nas grandes cidades, a chamada Zona Azul, que regulamenta o tempo de permanência e um preço estipulado para utilização, além de proporcionar maior rotatividade dinâmica do trânsito urbano (FILHO, 2017).

No entanto, a quantidade de vagas é limitada: em Curitiba, à época da construção deste trabalho, há 10.633 vagas do EstaR para veículos de passeio pagantes, 2.636 vagas para motos, 612 vagas de pisca-alerta e 248 de embarque e desembarque (SETRAN, 2017). Esta quantidade, se comparada diretamente ao número de veículos registrados na capital paranaense, por exemplo, mostra que as vagas ofertadas no centro da cidade estão aquém do necessário, uma vez que em junho de 2016, 976 mil automóveis estão registrados na cidade (GALINDO, 2017).

Para suprir a necessidade do depósito de veículos automotores nas grandes cidades, surgiram os estacionamentos comerciais, que são espaços onde as pessoas entregam a guarda o automóvel para o estabelecimento, mediante pagamento pelo tempo que o bem permanecer guardado (BELLEI, 2017).

No centro de Curitiba, haviam cerca de 433 estacionamentos privados no ano de 2011 e, como geralmente são bem localizados e próximos a locais de grande

fluxo de pessoas, foram alvos de grandes construtoras, que ofereceram grandes quantias para construção de novos empreendimentos, elevando o valor cobrado pelo serviço de estacionamento. (SCHONARTH, 2017).

Levando em consideração que os preços podem variar, uma ferramenta que possa indicar quais estabelecimentos oferecem o serviço de estacionamento pode se mostrar interessante. A partir daí é possível citar os seguintes aplicativos disponíveis nas lojas oficiais de aplicativos da Apple e do Google:

- LetsPark. Este aplicativo tem a proposta de encontrar um estacionamento conforme a preferência do usuário em relação a proximidade, valor cobrado, tempo previsto de permanência e tipo de veículo – moto, carro padrão e carro grande. Disponível tanto pra iOS quanto para Android (PLAY STORE, 2017);
- ParkMe Parking. A proposta desta solução visa, além de encontrar os serviços de estacionamento num mapa, permitir o pagamento via aplicativo. Disponível apenas em Android (PLAY STORE, 2017).

Além destes aplicativos, também existe uma série de aplicativos voltados para o pagamento e facilitação do uso de estacionamentos rotativos em várias cidades, como os da série ‘Estacionamento’, desenvolvidos pela empresa Mobilicidade Tools (PLAY STORE, 2017).

1.1 MOTIVAÇÃO

As principais motivações para o desenvolvimento deste trabalho são:

- A dificuldade de acesso a estacionamento no centro de Curitiba, tendo em vista que nem sempre é possível encontrar o melhor denominador comum entre o tempo, a proximidade, a possibilidade de levar a chave do carro, as formas de pagamento, o tipo de veículo que se usa, o preço e a chegada a determinado lugar que se deseja; ou seja, perguntas como: “O estacionamento aceita motos?”; “O estacionamento possibilita deixar bicicletas e/ou outros meios de transporte alternativos?”; “Qual a melhor opção próxima, considerando o tempo de permanência e proximidade com determinado lugar?” e

"Qual lugar tem manobrista e qual é possível estacionar e levar a chave?" são chaves para o desenvolvimento do trabalho;

- A necessidade de encontrar um local seguro e que atenda melhor às conveniências das pessoas se mostra pertinente, considerando que há pessoas que não andam com dinheiro e somente com cartão; também há aqueles que precisam permanecer por muitas horas; existem os que não gostam de deixar a chave do carro sob a responsabilidade de outros. As pessoas, agora em 2017, precisam adaptar suas necessidades para deixar seus veículos; considerando que a tecnologia possibilita o acesso às informações de forma mais fácil, não deveria haver uma forma de possibilitar aos indivíduos suprir essas deficiências?

1.2 OBJETIVOS

O objetivo geral deste trabalho é desenvolver um aplicativo móvel que apresente as opções de estacionamento disponíveis, mostrando em um mapa as opções de locais para deixar o veículo, as tarifas, a forma de tarifação, as formas de pagamento aceitas e os tipos de veículos aceitos.

Os objetivos específicos são os que seguem:

- A configuração de um servidor web que responda pela internet;
- A criação de um banco de dados central contendo dados dos estacionamentos;
- O desenvolvimento de um *Web Service* que seja responsável pela comunicação entre a aplicação e a base central de dados;
- Implementação de aplicação móvel que consuma os dados dispostos pelo *Web Service* e disponibilize as informações para os usuários;
- Avaliação do aplicativo desenvolvido, verificando requisitos como a usabilidade, design, funcionalidade e relevância para as pessoas.

1.3 ESCOPO

No contexto de fornecer uma ferramenta completa, é necessário coletar todas as informações, além de ser possível firmar parcerias com estabelecimentos para monetizar o aplicativo; no entanto, essa busca não foi iniciada até o momento, além de não haver uma previsão no cronograma para coleta de todas estas informações.

Dessa forma, o escopo deste trabalho é o desenvolvimento de um aplicativo, criado na plataforma Android, que permita aos usuários encontrarem a melhor opção de estacionamento no centro de Curitiba, Paraná, de acordo com suas necessidades. A manutenção e manipulação dos dados, a princípio, será manual e registrada por um administrador central.

1.4 METODOLOGIA

Os seguintes passos serão realizados, conforme a Figura 1, para o desenvolvimento do projeto:

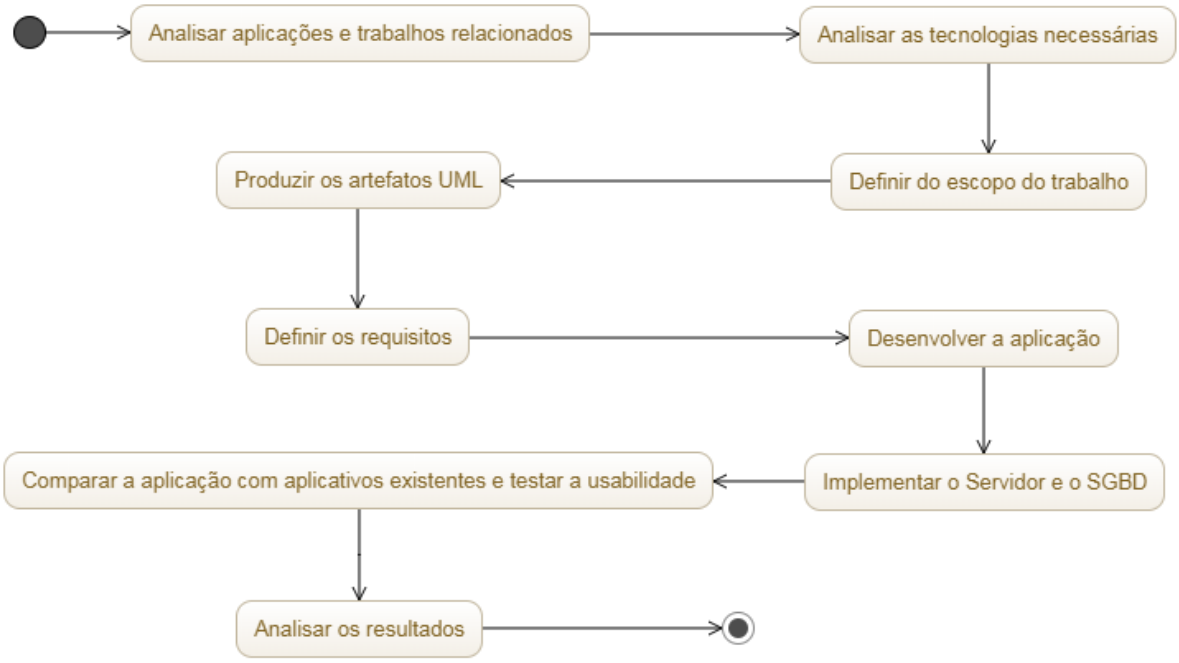

Figura 1- Diagrama de atividades que envolvem a metodologia (Fonte: autoria própria)

A análise de aplicativos existentes no mercado será realizada a partir da busca por resultados que envolvam estacionamentos nas lojas de aplicativos oficiais,

tanto para a plataforma Android quanto IOS, de modo que seja possível visualizar as seguintes informações:

- Quantidade de downloads realizados;
- Taxa de aceitação a partir das notas dadas pelos usuários;
- Leitura de comentários realizados sobre as plataformas;
- Utilização manual dos aplicativos, a fim de verificar o funcionamento, estabilidade, confiabilidade nos dados e conforto no manuseio.

A análise das tecnologias necessárias para o desenvolvimento do projeto inclui todas as ferramentas, APIs, banco de dados e ferramentas de integração para que tanto a aplicação móvel quanto o servidor de dados sejam implementados.

O escopo do trabalho está definido conforme a análise dos aplicativos existentes no mercado, bem como na janela de tempo prevista para o desenvolvimento.

Com o escopo do trabalho definido, a fase de análise e produção de artefatos UML será iniciada, cujo objetivo é definir e apresentar a forma que a aplicação móvel será desenvolvida, assim como os componentes.

Em seguida, os requisitos funcionais e não-funcionais serão definidos, de maneira a formatar as funcionalidades da aplicação.

A aplicação será desenvolvida, utilizando as tecnologias apresentadas e conforme as fases de análise.

O servidor de dados será desenvolvido para servir como fonte para que a aplicação consuma a listagem centralizada de locais de estacionamento.

Com as duas etapas de desenvolvimento prontas, a aplicação será comparada com as opções existentes no mercado e será executado um teste de usabilidade.

Por fim, o projeto passará para a fase de análise de resultados e comparação com aplicações existentes no mercado.

1.5 RESULTADOS ESPERADOS

O resultado esperado com este projeto é obter uma ferramenta, disponível na plataforma Android, contendo alguns dados base de estacionamento da região central de Curitiba, visando facilitar a vida das pessoas para encontrar estacionamentos no centro desta cidade.

Desse modo, espera-se com o desenvolvimento deste projeto: agilizar o acesso aos estabelecimentos; evitar a circulação desnecessária de veículos pelas ruas a procura de estacionamento; mitigar atrasos; possibilitar o consumo mais eficiente de combustível; mitigar a intensidade do trânsito na cidade e promover o uso de outros tipos de veículo como a bicicleta.

A aceitação do aplicativo é almejada, por parte dos usuários, considerando que o cadastro de estabelecimentos, neste momento inicial, fica restrito ao gerenciador da plataforma.

1.6 ORGANIZAÇÃO DO TRABALHO

Este trabalho está organizado com Introdução no Capítulo 1. No Capítulo 2, se encontra o referencial teórico utilizado como base para o desenvolvimento do projeto. No Capítulo 3 estão as informações pertinentes ao desenvolvimento prático. O Capítulo 4 mostra os resultados obtidos e a avaliação do aplicativo. O Capítulo 5 apresenta a conclusão do trabalho. A seguir são apresentadas as referências e, por fim, o Apêndice.

2 REFERENCIAL TEÓRICO

Este capítulo aborda as referências teóricas, isto é, aborda tanto as tecnologias quanto ferramentas de análise e engenharia de software que servirão de base para o desenvolvimento deste trabalho.

2.1 JAVA

Desenvolvida pela Sun Microsystems na década de 1990, o Java é uma das linguagens de programação mais utilizadas no mundo, segundo Deitel (2011). É uma linguagem compilada, onde é possível garantir integridade de código antes da execução de um programa. Interpretada através de uma máquina virtual, a linguagem se propôs a garantir portabilidade através de dispositivos, que foi um grande avanço para o mundo da época. (DEITEL, 2011).

O Java é orientado a objetos, cujo objetivo é implementar um conjunto de classes que serão responsáveis pela definição de padrões e comportamentos dos objetos que compõem e interagem entre si em um sistema de informação. Este modelo de análise é baseado na cognição, onde os objetos são representações de coisas reais, sejam físicas ou abstratas e, assim como na realidade, possuem características, limitações e formas de interação com o restante do ambiente. (DEITEL, 2011).

Dotado de várias bibliotecas para executar as mais diversas atividades, o Java pôde se popularizar, devido a algumas características, tais como (DEITEL, 2011):

- A facilidade de internacionalização devido à utilização do Unicode;
- Código aberto e contribuição pela comunidade;
- A portabilidade, que permitia a escrita de código apenas uma vez e o mesmo programa pode ser executado em qualquer plataforma pessoal, desde que esteja dotada da *Java Virtual Machine* (JVM);
- A distribuição de vários conjuntos de bibliotecas;
- Facilidade na criação de programas distribuídos, controle de múltiplas linhas de execução;
- Liberação automática de memória, através do Coletor de Lixo.

A partir dessas vantagens, a Google optou por utilizar sistemas operacionais de código aberto, adotando o Linux e o Java em seu sistema operacional para aplicações móveis: o Android. Desta forma, a principal linguagem de programação utilizada para desenvolver aplicações móveis para este sistema operacional é bastante conhecida de programadores habituados ao ambiente Web, facilitando a aquisição de conhecimento e o desenvolvimento de aplicações móveis (DEITEL, 2011).

2.2 ANDROID

Criado pela Android Inc. e adquirido pela Google em 2005, o Android foi desenvolvido baseado na plataforma Linux, possui código aberto, é dotado de máquina virtual Java embutida e é uma plataforma para tecnologia móvel que engloba pacotes de programas para celulares, sistema operacional, middleware, aplicativos e interface com usuário. O Android é apoiado pela *Open Handset Alliance* (OHA) – uma parceria de mais de quarenta empresas do segmento de telefonia móvel, abarcando operadoras de celular, fabricantes de aparelhos, empresas de semicondutores, de software e de comercialização (DEITEL, 2011).

O Android pode ser considerado o Sistema Operacional com maior utilização no mundo, pois, segundo a *International Data Corporation* (IDC, 2017), possui cerca de 85% (oitenta e cinco porcento) do mercado de Sistemas Operacionais de Smartphones no primeiro trimestre de 2017, mantendo quase essa mesma fatia de mercado durante todo o ano de 2016.

Por possuir o código aberto, o sistema evolui à medida que o mercado muda, bem como é flexível para incorporar novas tecnologias e para agregar as modificações propostas pela comunidade (SILVA, 2012).

Uma das questões recorrentes é a segurança, principalmente em um ambiente em que várias pessoas e empresas contribuem e desenvolvem para a plataforma, na qual algum indivíduo mal-intencionado pode desenvolver um código malicioso que poderia comprometer informações do usuário. Na plataforma, cada aplicação pode ser considerada *Sandbox*, isto é, roda em um contexto separado, não possuindo acesso a informações de outros aplicativos, sensores e mecanismos do dispositivo, a menos que seja dada expressa autorização pelo usuário:

> "*Como é executado em um kernel Linux, toda vez que um aplicativo for instalado em uma estação Android™, é criado um novo usuário Linux para aquele programa, com diretórios que serão usados pelo aplicativo, mas somente para aquele usuário Linux. Como os aplicativos ficam completamente isolados uns dos outros, qualquer tentativa de acessar informações de outro aplicativo precisa ser explicitamente autorizada pelo usuário, podendo ser negada a instalação do aplicativo, ou autorizada a instalação, mas controlando as permissões que este aplicativo poderá ter através de um mecanismo de permissão*" (SILVA; PEREIRA, 2012. p. 4).

O Android SDK é o kit de desenvolvimento para a tecnologia que, utilizando Java, dá ferramentas e interfaces para programação de aplicações da plataforma e o ambiente integrado de desenvolvimento (IDE) mais utilizado é o *Android Studio* (DEITEL, 2017).

2.3 ANDROID STUDIO

O Android Studio é a IDE oficial para desenvolvimento, é baseado no IntelliJ IDEA e oferece uma série de ferramentas de produtividade para desenvolvimento de aplicações, tais como (SILVA, 2012):

- Emulador de ambiente;
- Sistema de compilação baseado no Graddle e Maven;
- Compatibilidade com a ferramenta GitHub;
- Ferramentas de teste;
- Ferramentas de análise de código;
- Rico editor de Layout;
- Compatibilidade com o Google Cloud Platform.

Lançado em 2014, para substituir a ferramenta oficial até então – Eclipse Android Development Tools (ADT), foi criado especificamente para o desenvolvimento de aplicativos para o Sistema Operacional e está disponível para utilização nas plataformas Windows, Linux e Mac OS (SILVA, 2012).

A instalação e configuração da IDE é fácil, uma vez que em um mesmo pacote traz todo o conjunto de ferramentas necessárias que são instaladas no procedimento de instalação (SILVA, 2012).

2.4 GEOLOCALIZAÇÃO

A geolocalização é a localização de um dispositivo em relação a um referencial. A forma mais comum em aplicações móveis é o *Global Positioning System* (GPS) (MACHADO, 2015), que permite dar a localização precisa, baseado num sistema de 24 satélites que se encarregam de dar a posição em coordenadas geográficas de um dispositivo com receptor de sinais GPS (DILÃO, 2017).

Geralmente com um sensor integrado aos dispositivos móveis, o GPS necessita de permissão específica para utilização em uma aplicação Android (SILVA, 2012).

A definição e especificação da API de geolocalização está a cargo da World *Wide Web Consortium* (W3C) (W3C, 2017).

2.5 BANCO DE DADOS

O banco de dados é um conjunto de dados relacionados, que representam informações de um determinado domínio, de modo que é possível agrupar esses dados para representar da melhor forma possível (KORTH, 2006).

Amplamente utilizados no mercado, os bancos de dados entidade-relacionais foram criados com base na teoria de conjuntos e são compostos por tabelas e um sistema de gerenciamento de banco de dados (SGBD), que é responsável por garantir integridade dos dados armazenados, controle de acesso concorrente, a segurança e identidade de acesso, ferramentas de manipulação e execução de procedimentos, além, é claro, do rápido acesso aos dados (SCHIMIGUEL, 2017).

A linguagem utilizada para manipulação dos SGBDs relacionais é o *Structured Query Language* (SQL), cujo o objetivo é a realização de consultas para definição de estruturas e manipulação de conjuntos de informações (OLIVEIRA, 2017).

Em um ambiente de aplicações móveis, a limitação do hardware implica na necessidade de um SGBD com grandes informações estar em um servidor na nuvem, uma vez que o SQLite – disponível para rodar direto no dispositivo – é indicado para prover armazenamento local, promovendo economia, eficiência, independência e simplicidade (SQLITE, 2017). As consultas aos servidores são efetuadas através da internet utilizando WebServices, que encapsulam a consulta

direta à base de dados, expondo apenas uma camada de serviço simples que é acessível às aplicações (BICALHO, 2014).

2.7 POSTGRES

O Postgres é um SGBD objeto-relacional de código aberto, que é executado nos principais sistemas operacionais existentes e possui suporte completo a chaves estrangeiras, joins, views, triggers e procedimentos armazenados. (POSTGRESQL, 2017).

Os tipos de dados permitidos neste sistema de banco de dados são: INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL, TIMESTAMP, além de objetos binários, como figuras, áudios e vídeos. (POSTGRESQL, 2017).

2.8 REALM

Considerando que nem sempre a internet está disponível em um dispositivo móvel, seja por permissão de usuário, seja pelo ambiente em que se encontra, trabalhar com uma aplicação móvel que não possa trabalhar de forma desconectada à rede pode ser um problema, dessa forma, o aplicativo pode precisar persistir alguns dados localmente, permitindo a sua utilização mesmo sem conexão, podendo efetuar uma sincronização depois que a rede estiver disponível novamente (BICALHO, 2014).

O Realm, disponível para utilização em várias plataformas móveis, é uma ferramenta que permite o gerenciamento de banco de dados orientado a objetos (REALM, 2017).

Segundo a Realm (2017), esse sistema de gerenciamento traz algumas vantagens em relação a outras opções do mercado, como:

- Abrangência em várias plataformas;
- Funcionamento sem necessidade de utilização da internet;
- Sincronização com dados da nuvem.

2.9 WEB SERVICES

Os Web Services compõem uma tecnologia para comunicação entre aplicações ou plataformas diferentes, tornando disponível a integração entre sistemas e módulos de sistemas que estão tanto na nuvem, quando em PCs ou dispositivos móveis (SOA WEBSERVICES, 2017).

As requisições REST (*Representational State Transfer*), removem os detalhes de implementação como sintaxe de protocolo de comunicação, implementação detalhada de componentes; partindo do princípio que toda mensagem HTTP possui todas as informações necessárias para a comunicação entre dois sistemas, utilizando as operações disponíveis pelo próprio protocolo como o PUT, GET, POST e DELETE (BE CODE, 2017).

2.10 UML

A Linguagem de Modelagem Unificada (UML), é uma linguagem criada para definição, especificação, documentação e visualização de artefatos utilizados para construção e implementação de um software orientado a objeto (UFCG, 2017).

A utilização desta linguagem facilita a comunicação e o entendimento do sistema a ser implementado, uma vez que define itens, diagramas e relacionamentos que podem representar de forma visual todo o comportamento e estrutura do sistema (UML, 2017).

Para o desenvolvimento deste trabalho, os seguintes artefatos da UML serão utilizados:

- Diagrama de Caso de Uso: representa graficamente as funcionalidades do sistema do ponto de vista do usuário, cujo propósito é facilitar a comunicação entre o analista e o cliente (UFCG, 2017);
- Diagrama de Classes: representa os diversos tipos de objetos do sistema, bem como a intercomunicação entre eles. A perspectiva de classes de software, cuja representação mostra os detalhes de implementação, será utilizada neste trabalho. (UFCG, 2017).
- Diagrama de Atividades: apresenta um fluxo de atividades em um processo, demonstrando a interdependência entre as atividades; (UFCG, 2017).

- Diagrama de Componentes: mostra a estrutura do software, sob o ponto de vista de componentes, descrevendo-os juntamente com suas dependências e interfaces. (IBM, 2017).

2.11 RETROFIT

O Retrofit é uma API que permite utilizar um cliente HTTP, com segurança de tipo, para Android e Java. (SQUARE, 2017). Desenvolvida pela empresa Square, a biblioteca segue os padrões REST e transforma o cliente HTTP em uma interface Java, de modo a permitir uma simples implementação de comunicação entre uma aplicação e um servidor. (DEVMEDIA, 2017).

2.12 HIBERNATE

O Hibernate é uma interface de programação de aplicações desenvolvida em Java para o mapeamento objeto-relacional, funciona sobre a especificação do *Java Persistence API* (JPA), facilita a comunicação com o banco de dados e aumenta a produtividade de desenvolvimento. (CAELUM, 2017).

Segundo a Caelum, o Hibernate possui as seguintes vantagens:

> *"O Hibernate abstrai o seu código SQL, toda a camada JDBC e o SQL será gerado em tempo de execução. Mais que isso, ele vai gerar o SQL que serve para um determinado banco de dados, já que cada banco fala um 'dialeto' diferente dessa linguagem. Assim há também a possibilidade de trocar de banco de dados sem ter de alterar código Java, já que isso fica de responsabilidade da ferramenta".* (CAELUM, 2017).

2.13 SPRING BOOT

O Spring Boot é um *framework* cujo objetivo é aprimorar a produtividade no desenvolvimento de aplicações corporativas, uma vez que cria um ecossistema que permite rápida configuração de ambiente. É utilizado na etapa de implementação de aplicações que precisam de um servidor JEE, permitindo que um servidor Web entre em funcionamento em pouco tempo; possibilitando ao desenvolvedor focar na implementação da aplicação de fato e suas regras de negócio. (DEVMEDIA, 2017).

2.14 TOMCAT

O Tomcat é um servidor web Java desenvolvido pela Apache Software Foundantion, considerado mais leve e eficiente que outros servidores disponíveis para a plataforma (2011, TECHTUDO). É uma implementação em código aberto, sob a licença Apache versão 2, do Java Servlets, JavaServer Pages, Java Expression Language e Java WebSocket (2017, APACHE).

2.15 SYSTEM USABILITY SCALE

O *System Usability Scale* (SUS) é uma escala númerica de usabilidade, cujo objetivo é fornecer uma análise quantitativa através de testes com usuários reais. Permite a identificação de problemas e dificuldades que as pessoas podem enfrentar ao utilizar um sistema, através de um teste que é cientificamente apurado ao mesmo tempo em que não é muito longo para o usuário (SAURO, 2011).

O *System Usability Scale* permite melhor avaliação dos seguintes pontos de um sistema (SAURO, 2011):

- Efetividade;
- Eficiência;
- Satisfação.

É proposto um teste com 10 questões e, após sua aplicação com os usuários, a análise dos resultados é feita da seguinte forma (SAURO, 2011):

- Para questões de números ímpares, deve-se subtrair 1 da pontuação; por exemplo, na questão 1, se a nota dada foi 3, será contabilizado 2;
- Para questões de números pares, o valor a ser considerado consiste na subtração entre 5 e o valor obtido na resposta;
- Soma-se todos os novos valores calculados para cada usuário e multiplica-se por 2,5. O resultado é a pontuação da aplicação para um determinado usuário.

Com os valores obtidos, é possível analisar a pontuação do sistema em termos de usabilidade. Considerando que a média do SUS é 68 pontos, um valor inferior a este indica que o sistema possui problemas de usabilidade (SAURO, 2011).

2.16 APLICAÇÕES EXISTENTES NO MERCADO

Para levantamento dos requisitos, também é necessário o estudo das aplicações existentes no mercado. Na PlayStore, estão disponíveis duas aplicações: o ParkMe e o Let's Park.

O LetsPark é um aplicativo disponível para Android e IOS, desenvolvido pela *Let's Park Company*, que oferece a funcionalidade de encontrar estacionamentos abertos em várias cidades do Brasil, permitindo a apresentação do preço diretamente no marcador, conforme a seleção de tempo pretendido. A versão disponível na produção deste trabalho é a 2.0.0, atualizada em maio de 2015. Possui entre dez mil e cinquenta mil downloads e é classificada como 3,6 estrelas na escala de avaliação da Play Store. (PLAYSTORE, 2017).

As funcionalidades disponíveis pela aplicação são:

- Avaliação de locais pelos usuários;
- Imagem do estacionamento disponibilizada pelo Street View;
- Possibilidade ao usuário de envio de sugestões e correções para os desenvolvedores;
- Obtenção de rotas para o estabelecimento escolhido;
- Busca por endereço do estacionamento;
- Formulário de cadastro de locais, onde o usuário pode sugerir novos cadastros;
- Seleção de tipo de automóvel, permitindo a escolha entre moto, carro e caminhonete;
- Preferência de busca pelo melhor preço ou por andar menos até o local desejado;
- Horários e preços praticados pelos estacionamentos. (PLAYSTORE, 2017).

O aplicativo *ParkMe*, criado pela *INRIX, Inc*, é voltado para o público que deseja estacionar tanto em locais de Zona Azul, quanto estabelecimentos privados. A versão atual, 2.0.0, foi disponibilizada em 15 de novembro de 2017, está avaliada em 3.8 estrelas no ranking de classificação da Play Store (PLAYSTORE, 2017).

As funcionalidades disponíveis por esta aplicação são as seguintes:

- Avaliação de locais pelos usuários;

- Reserva de estacionamento;
- Foto do local;
- Permite ligação;
- Obtenção de direções;
- Permite informar o quão cheio está o local;
- Avaliação dos locais;
- Horários e preços praticados;
- Lista de serviços disponíveis;
- Cronômetro de contagem regressiva;
- Marcar local de estacionamento;
- Avaliação do aplicativo;
- Filtros por vários serviços;
- Permite filtrar estacionamentos de zona azul;
- Filtro por estacionamentos que trabalham com mensalidade;
- Exibir estacionamentos em forma de lista;
- Permite escolher endereço ou local para busca de estacionamentos nos arredores (PLAYSTORE, 2017).

3 DESENVOLVIMENTO

3.1 DESCRIÇÃO GERAL DO APLICATIVO MÓVEL

O aplicativo, cujo objetivo é guiar o usuário a um estabelecimento para estacionar o seu veículo conforme seus filtros, será desenvolvido sobre a plataforma Android, com a linguagem de programação Java e na IDE Android Studio.

A aplicação móvel mostrará, já na abertura, a lista de estacionamentos disponíveis nos arredores, baseando-se na localização atual do dispositivo móvel; a lista será apresentada sob a forma de *pins*, facilitando a visualização dos locais disponível.

Com a lista apresentada, o usuário pode clicar em um dos marcadores e será apresentada uma janela com o nome do estabelecimento. Ao clicar nesta janela, serão exibidos os detalhes do local, contendo o nome, endereço, serviços disponíveis, hora de abertura e fechamento, além das tarifas aplicadas.

O usuário poderá obter rotas e direções para o local escolhido, tanto da tela de detalhes quanto diretamente do mapa e, a aplicação padrão de navegação deverá ser aberta.

O usuário também poderá filtrar as localidades conforme suas necessidades, podendo escolher as seguintes opções:

- Tempo previsto de permanência;
- Aceita cartões;
- Possui bicicletário;
- Possui manobrista;
- Permite que o usuário estacione e leve a chave.

Além disso, será possível iniciar estacionamento, de maneira que o aplicativo emitirá um aviso ao usuário para alertá-lo sobre a permanência no local.

A lista de estacionamentos será mantida e atualizada pelo administrador de dados e será disponibilizada para a aplicação móvel através de um Web Service Restful, permitindo um cadastro centralizado dos dados.

3.2 REQUISITOS

Os requisitos funcionais do aplicativo são:

- Apresentar um mapa com estacionamentos em um raio definido em quilômetros;
- Permitir que o usuário filtre suas preferências de localização:
  - Raio de busca;
  - O tempo de permanência pretendido;
  - Se aceita cartões; se permite o estacionamento pelo próprio motorista;
  - Se possui manobrista;
  - Se possui bicicletário ou permite o estacionamento de bicicletas.
- Ao clicar no marcador de estacionamento no mapa (*pin*), o sistema deve apresentar o nome do estabelecimento;
- Ao clicar no nome do estabelecimento apresentado, o sistema deve apresentar os detalhes do estacionamento.
- Os detalhes do estacionamento devem incluir nome, endereço, serviços oferecidos; listagem de valores e a opção de obter rotas para o local;
- Ao selecionar a opção de rotas, o sistema deve abrir o aplicativo padrão de navegação do dispositivo, iniciando a navegação para o local.

Os requisitos não-funcionais são:

- Deve ser desenvolvido em linguagem de programação Java;
- Deve ser projetado para funcionar na plataforma Android;
- Deve consumir os dados de um Web Service;
- Não deve utilizar nenhum sistema de login ou solicitar informações do usuário.

3.3 DIAGRAMA DE CASO DE USO

O diagrama de caso de uso para o sistema possui as funcionalidades a seguir, cujos atores são o Usuário da aplicação móvel e o Administrador de dados do servidor:

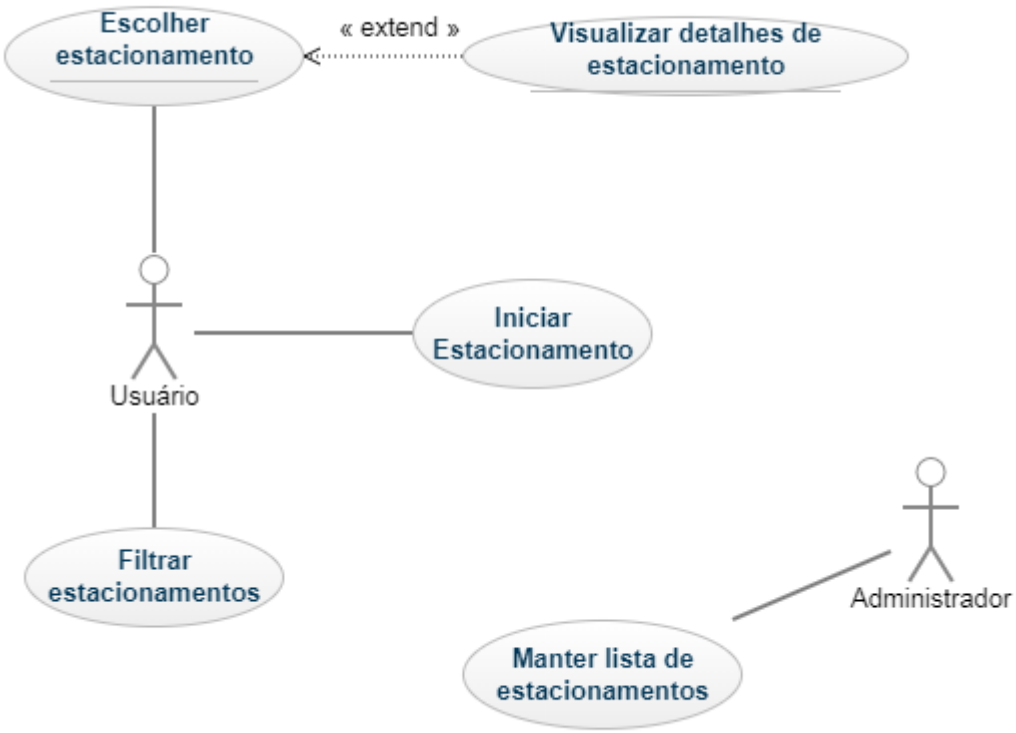

Figura 2 – Diagrama de Caso de Uso do sistema (Fonte: autoria própria)

3.3.1 Iniciar Estacionamento

Objetivo: O usuário inicia um cronômetro de contagem regressiva para controlar um tempo de estacionamento determinado.

Ator: Usuário da aplicação móvel

Prioridade: ☐ Essencial ■ Importante ☐ Desejável

Entradas e pré-condições: Sistema deve estar em execução.

Frequência de uso: ☐ Frequente ■ Eventual

Criticidade: ☐ Alta ■ Média ☐ Baixa

Fluxo principal:

I. O usuário abre o menu da aplicação e seleciona a opção ‘Estacionar’;
II. O sistema abre a tela Inicar Estacionamento contendo:
    a. Campo de escolha de tempo;
    b. Botão Iniciar estacionamento.
III. O usuário escolhe o período que deseja permanecer estacionado e seleciona a opção Iniciar Estacionamento;
IV. O sistema inicia uma contagem regressiva e, ao final da contagem, apresentará um alerta e uma notificação informando que o período terminou; [A1]
V. O caso de uso é encerrado.

Fluxos alternativos:

A1. Alterar contagem regressiva:

I. O usuário altera o campo de tempo para nova contagem regressiva e seleciona a opção ‘Estacionamento em andamento’;
II. O sistema cancela a contagem anterior;
III. Retorna ao fluxo principal no passo IV.

Saídas e pós-condições: O usuário é notificado que o seu tempo de estacionamento se encerrou.

3.3.2 Escolher Estacionamento

Objetivo: O usuário, de posse da listagem de estacionamentos, pode escolher um estacionamento e clica para visualizar informações gerais ou obter rotas.

Ator: Usuário da aplicação móvel

Prioridade: ■ Essencial ☐ Importante ☐ Desejável

Entradas e pré-condições: Sistema deve estar em execução na tela inicial.

Frequência de uso: ■ Frequente ☐ Eventual

Criticidade: ■ Alta ☐ Média ☐ Baixa

Fluxo principal:

I. O usuário abre a aplicação ou seleciona a opção ‘Início’;
II. O sistema abre o mapa centralizado na localização do dispositivo móvel [A1];

III. O sistema exibe os marcadores de estacionamento no mapa;
IV. O usuário clica em um marcador de estacionamento;
V. O sistema exibe janela com o nome do estacionamento.
VI. Caso o usuário clique na janela com o nome do estacionamento, o Caso de Uso Visualizar Detalhes do Estacionamento é iniciado;
VII. O caso de uso é encerrado.

Fluxos alternativos:

A1. Falha na comunicação com o Web Service:

I. O sistema encontra uma falha na comunicação com o Web Service;
II. O sistema exibe mensagem de erro de comunicação.
III. Retorna ao fluxo principal no passo VII.

Saídas e pós-condições: Nome do local é apresentado na janela de informações do marcador.

3.3.3 Visualizar Detalhes de Estacionamento

Objetivo: O usuário pode clicar na caixa de informações gerais do estacionamento e visualizar mais informações do local.

Ator: Usuário da aplicação móvel

Prioridade: ■ Essencial ☐ Importante ☐ Desejável

Entradas e pré-condições: Sistema deve estar em execução na tela inicial com a caixa de informações gerais aberta para um estacionamento. Marcadores de estacionamento disponíveis no mapa. Execução com sucesso do fluxo principal do Caso de Uso Escolher Estacionamento.

Frequência de uso: ■ Frequente ☐ Eventual

Criticidade: ■ Alta ☐ Média ☐ Baixa

Fluxo principal:

I. O sistema exibe nova tela, exibindo as seguintes informações:
    a. Nome do estabelecimento;
    b. Endereço;
    c. Serviços disponíveis;
    d. Tarifas aplicadas;

II. O sistema exibe opção de obter rotas para o local.
III. O caso de uso é encerrado.

Fluxos alternativos:

A1. Falha na comunicação com o Web Service:
I. O sistema encontra uma falha na comunicação com o Web Service na busca dos detalhes;
II. O sistema exibe mensagem de erro de comunicação;
III. O sistema retorna à tela inicial;
IV. Retorna ao fluxo principal no passo IV.

Saídas e pós-condições: Tela de detalhes do estacionamento é exibida. Sistema apresenta mensagem de erro apropriada.

Ao administrador de dados do servidor, cabe a ação de manter a lista de estacionamentos, sendo de responsabilidade deste ator a coleta, preenchimento e atualização de todas as informações referentes aos locais de estacionamento.

3.3.4 Filtrar Estacionamentos

Objetivo: o usuário pode escolher filtros personalizados para obter uma nova listagem de estacionamentos, conforme suas necessidades.

Ator: Usuário da aplicação móvel

Prioridade: ■ Essencial ☐ Importante ☐ Desejável

Entradas e pré-condições: Sistema deve estar em execução na tela inicial com a caixa de informações gerais aberta para um estacionamento. Marcadores de estacionamento disponíveis no mapa.

Frequência de uso: ☐ Frequente ■ Eventual

Criticidade: ■ Alta ☐ Média ☐ Baixa

Fluxo principal:

I. O usuário acessa o menu de opções do aplicativo móvel;
II. O usuário seleciona a opção Filtros do menu;
III. O sistema apresenta os seguintes filtros para o usuário [A1]:
    a. Raio de busca;
    b. Tempo previsto de permanência;
    c. Aceita cartão;

d. Possui bicicletário;
e. Possui manobrista;
f. Permite estacionar e levar a chave.

IV. O usuário marca as opções de filtro que necessita;
V. O usuário seleciona a opção Confirmar [A2];
VI. O sistema retorna à tela inicial apresentando a lista de estacionamentos filtrada;
VII. O caso de uso é encerrado.

Fluxos alternativos:

A1. Falha na comunicação com o Web Service:
I. O usuário seleciona o botão voltar;
II. O sistema retorna à tela inicial, sem nenhum alteração;
III. Retorna ao fluxo principal no passo VII.

A2. Falha na comunicação com o Web Service:
I. O sistema encontra uma falha na comunicação com o Web Service na busca dos detalhes;
II. O sistema exibe mensagem de erro de comunicação;
III. O sistema retorna à tela inicial;
IV. Retorna ao fluxo principal no passo VII.

Saídas e pós-condições: Lista de estacionamentos filtradas é apresentada no mapa.

Já ao administrador de dados do servidor, cabe a ação de manter a lista de estacionamentos, sendo de responsabilidade deste ator a coleta, preenchimento e atualização de todas as informações referentes aos locais de estacionamento.

3.3.5 Manter Lista de Estacionamentos

Descrição do caso de uso: O administrador mantem a listagem e os detalhes dos estacionamentos, efetuando inclusões, exclusões e alterações.

Ator: Administrador de dados

Prioridade: ■ Essencial ☐ Importante ☐ Desejável

Entradas e pré-condições: A interface de administração da aplicação deve estar aberta na tela inicial.

Frequência de uso: ☐ Frequente ■ Eventual

Criticidade: ■ Alta ☐ Média ☐ Baixa

Fluxo principal:

I. O Administrador seleciona a opção de manter estacionamentos;
II. O sistema exibe a lista de estacionamentos [A2] [A3] [A4];
III. O caso de uso é encerrado.

Fluxos alternativos:

A1. Falha na comunicação com o Web Service:

I. O sistema encontra uma falha na comunicação com o Web Service na busca dos detalhes;
II. O sistema exibe mensagem de erro de comunicação;
III. Retorna ao fluxo principal no passo III.

A2. Cadastrar novo estacionamento:

I. O usuário seleciona a opção de cadastro de novo estacionamento;
II. O sistema exibe os seguintes campos para preenchimento:
    a. Nome do estabelecimento;
    b. Latitude e Longitude;
    c. Serviços disponíveis;
    d. Tarifas aplicadas;
III. O usuário preenche todas as informações e seleciona a opção Confirmar [A1];
IV. Retorna ao fluxo principal no passo II.

A3. Alterar Estacionamento

I. O usuário seleciona a opção de edição de estacionamento presente na lista;
II. O sistema exibe os seguintes campos preenchidos com as informações do estacionamento:
    a. Nome do estabelecimento;

b. Latitude e Longitude;
c. Serviços disponíveis;
d. Tarifas aplicadas;

III. O usuário altera quaisquer campos que desejar e seleciona a opção Confirmar [A1];
IV. Retorna ao fluxo principal no passo II.

A4. Remover Estacionamento:

I. O usuário seleciona a opção de exclusão de estacionamento presente na lista;
II. O sistema remove o estacionamento da lista;
III. Retorna ao fluxo principal no passo II.

Saídas e pós-condições: Mensagem de informações do estacionamento incluídas com sucesso.

3.4 DIAGRAMAS DE CLASSES

Utilizando a perspectiva de classes de software, com objetivo de facilitar a modelagem e desenvolvimento do sistema, a figura a seguir representa as classes do aplicativo:

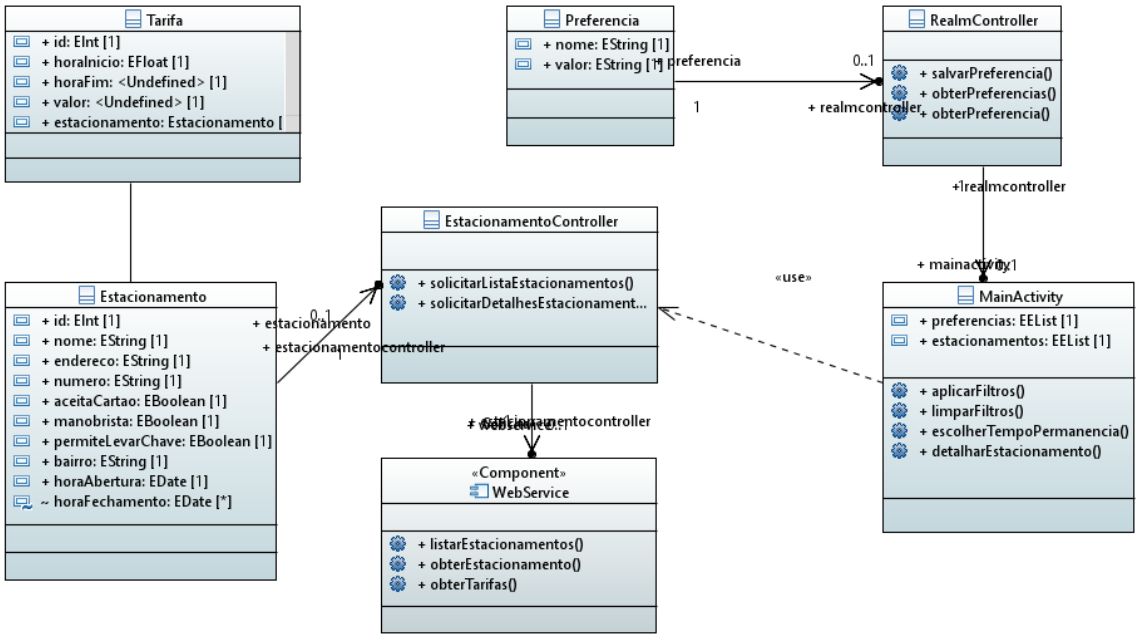

Figura 3 - Diagrama de Classe da aplicação móvel (Fonte: autoria própria)

As classes Estacionamento e Tarifa representam objetos que serão obtidos do Web Service com as informações pertinentes aos locais de estacionamento e suas tarifas.

A MainActivity é a classe principal da aplicação móvel, ela controla todos os componentes de tela.

EstacionamentoController é a classe responsável pela comunicação com o Web Service e armazenamento das listas de resultados.

Preferência é a classe responsável por guardar as preferências de filtros que serão persistidas na base de dados local Realm e, RealmController é responsável por executar as consultas na base de dados Realm.

O componente WebService é a extensão de classes para busca das informações no servidor.

A figura a seguir ilustra as classes que deverão estar presentes no servidor, de maneira a obter todas as informações do banco de dados e fornecê-las via Web Service:

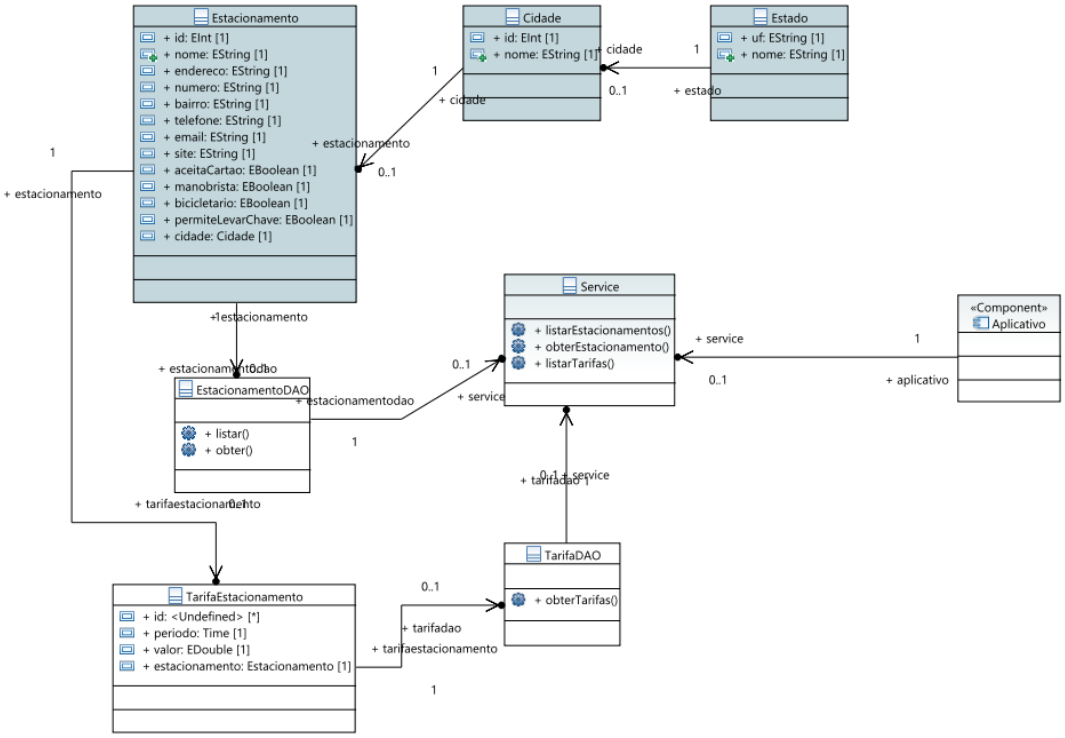

Figura 4 - Diagrama de Classes do Servidor (Fonte: autoria própria)

A principal classe do servidor é a Service, ela é responsável por expor o serviço web para que a aplicação móvel consuma. As classes DAO são responsáveis pelo acesso e manipulação de dados no SGBD Postgres. E as classes Estacionamento, TarifaEstacionamento, Cidade e Estado são representações de entidades do banco de dados.

3.5 DIAGRAMA DE ENTIDADE-RELACIONAMENTO

A ilustração a seguir representa o diagrama de entidade-relacionamento das tabelas de banco de dados no servidor:

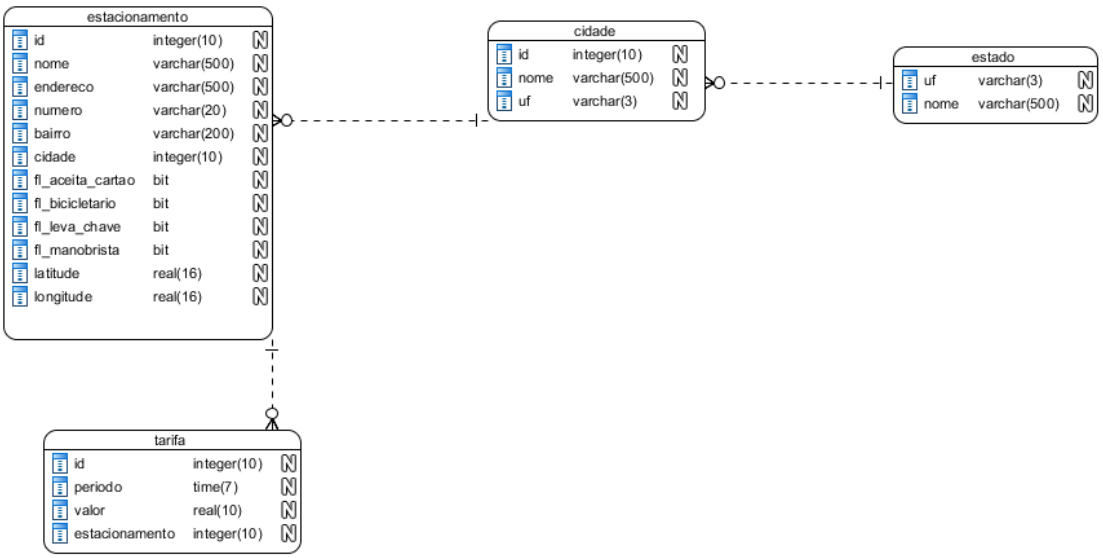

Figura 5 - Diagrama de Entidade-Relacionamento (Fonte: autoria própria)

A tabela Estacionamento possui os dados relativos ao local, bem como as informações dos serviços oferecidos. A tabela Tarifa apresenta as tarifas aplicadas por cada estacionamento; a tabela Cidade possui apenas os nomes dos municípios e a tabela Estado possui os nomes e siglas das Unidades da Federação brasileira.

3.6 DIAGRAMA DE COMPONENTES

Os componentes do sistema estão divididos entre aplicação móvel e servidor. Enquanto a aplicação móvel é responsável pela interface e interações com o usuário, o servidor se subdivide em dois componentes: Web Service, como servidor de dados para o aplicativo e o Banco de dados Postgres, que armazena as informações de estacionamentos e tarifas. A seguir, o diagrama apresenta os componentes do sistema:

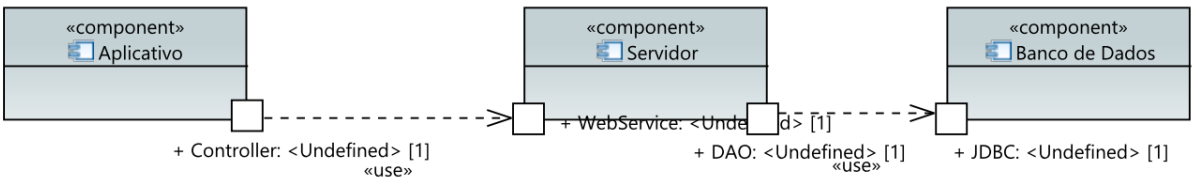

Figura 6 - Diagrama de Componentes do sistema (Fonte: autoria própria)

3.7 DIAGRAMA DE ATIVIDADES

As atividades da aplicação móvel estão descritas no diagrama a seguir:

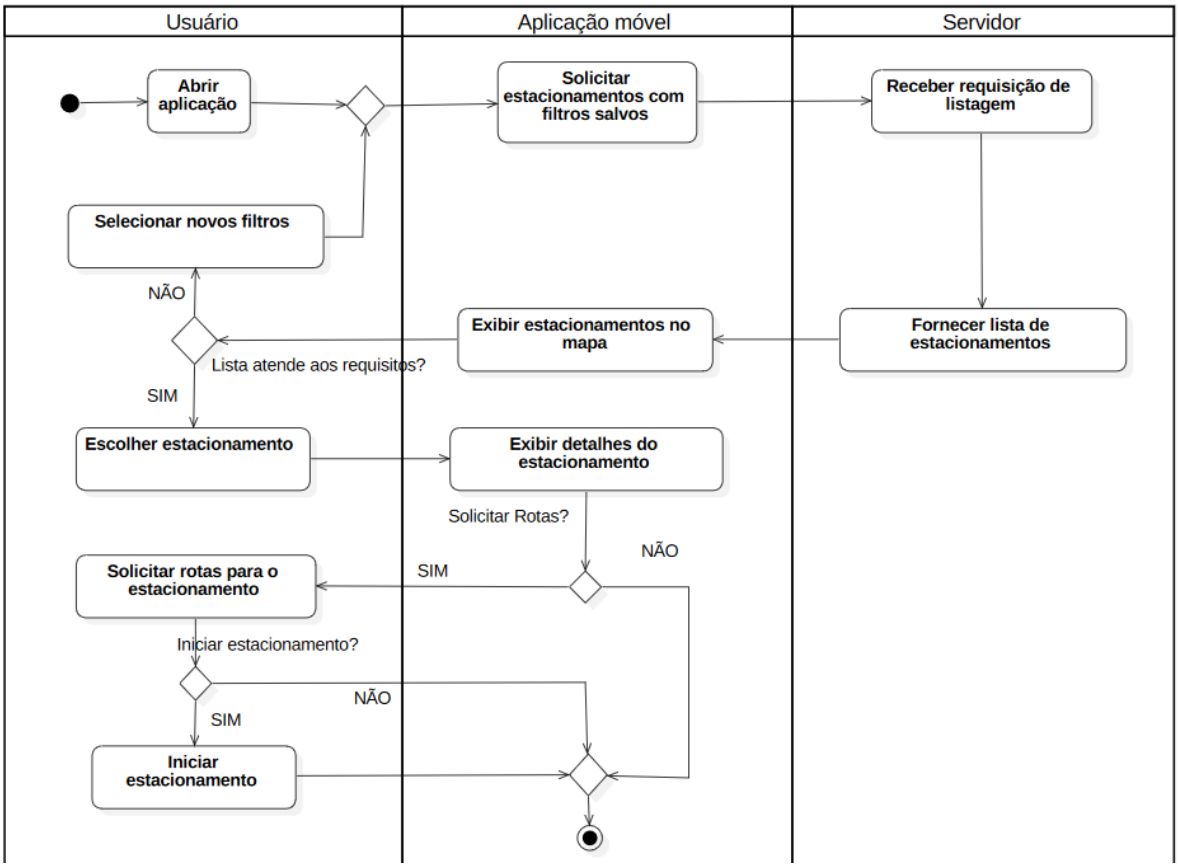

Figura 7 - Diagrama de atividades (Fonte: autoria própria)

As atividades se iniciam ao abrir a aplicação móvel, que inicia a atividade de solicitação de lista de estacionamentos para o servidor, baseando-se nos filtros salvos; servidor recebe a requisição da aplicação móvel e responde com uma listagem filtrada de estacionamentos; a aplicação móvel então exibe os estacionamentos no mapa. Caso a lista não atenda às necessidades do usuário, é possível ao usuário a seleção de novos filtros e o fluxo retorna à atividade de solicitação de estabelecimentos baseada nos filtros salvos, podendo repetir o ciclo quantas vezes forem necessárias.

Se a listagem for suficiente ao usuário, a atividade de escolha de estacionamento é iniciada e o sistema exibe, na sequência, os detalhes do local escolhido.

Caso o usuário queira solicitar rotas para o local, é iniciada a atividade de solicitação de rotas, que abrirá a aplicação padrão de navegação; caso contrário o fluxo é encerrado.

A partir da atividade de solicitação de rotas, se o usuário desejar é possível: iniciar estacionamento, que inicia uma contagem regressiva e, ao final, o fluxo é encerrado; ou encerrar o fluxo sem nenhuma outra ação.

3.8 PROTÓTIPOS DE TELA

Para o desenvolvimento da aplicação, protótipos de tela foram desenvolvidos para facilitar a visualização e entendimento visual das principais funcionalidades e formas de interação com o usuário.

A Figura 8 apresenta a tela inicial da aplicação, cuja função é apresentar o mapa e a marcação de estacionamentos próximos em um determinado raio de pesquisa.

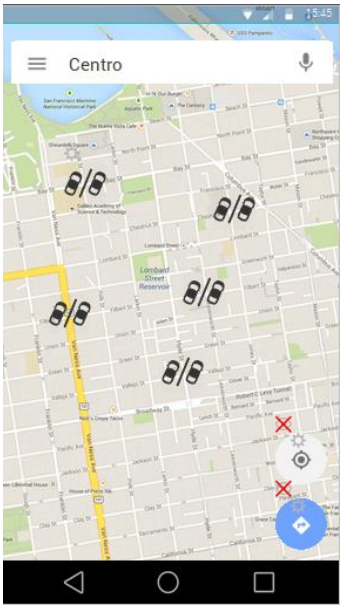

Figura 8 - Protótipo da tela inicial do aplicativo (Fonte: autoria própria)

A figura a seguir representa a tela logo após o usuário clicar em um estabelecimento, apresentando as informações gerais do local.

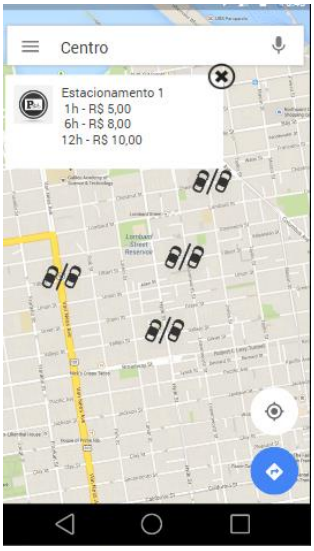

Figura 9 - Pin de estacionamento (Fonte: autoria própria)

A ilustração a seguir apresenta o menu de opções para o usuário, de maneira que disponibiliza navegabilidade para as principais telas do sistema: a inicial e a tela de configurações.

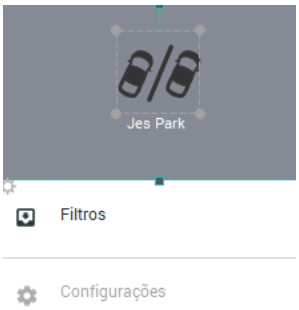

Figura 10 - Menu do sistema (Fonte: autoria própria)

A Figura 11 a seguir representa as configurações do sistema, cujo objetivo é escolher os filtros que melhor se aplicam para o usuário e sua necessidade de encontrar um local que melhor o atende.

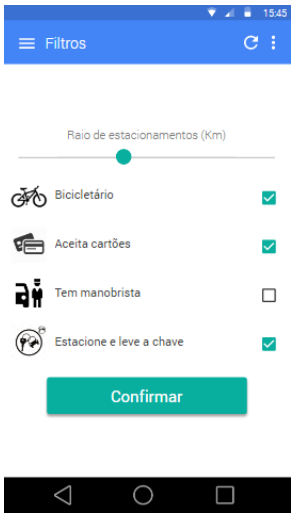

Figura 11 - Tela de filtros/configurações (Fonte: autoria própria)

4 RESULTADOS OBTIDOS

4.1 IMPLEMENTAÇÃO

A implementação do projeto foi dividida, conforme a explanado na Seção 3.6, em dois módulos: a aplicação móvel, que apresenta toda a interface com o usuário; e o servidor, que possui o banco de dados central de estacionamentos e o disponibiliza via Web Service RESTFUL.

4.1.1 Implementação da aplicação móvel

A implementação da aplicação móvel para o sistema operacional móvel Android utilizou a IDE Android Studio e a linguagem de programação Java para o desenvolvimento das ações e comportamentos. Optou-se por utilizar fragmentos (*Fragments*) gerenciados por uma Atividade (*Activity*) principal, de modo a facilitar a transição entre as telas.

A atividade principal (*ActivityMain*) tem por função gerenciar as transições de tela, a comunicação entre os fragmentos, assim como manter a instância do controlador principal (*EstacionamentoController*). A transição de tela ocorre por meio do método *addFragment*, apresentado na figura a seguir:

```
private void addFragment(Fragment fragment) {
    fragment.setArguments(new Bundle());
    FragmentTransaction fragmentTransaction = getFragmentManager().beginTransaction();
    fragmentTransaction.add(R.id.content_main, fragment);
    fragmentTransaction.addToBackStack(null);
    fragmentTransaction.commit();
}
```

Figura 12 - Método addFragment (Fonte: autoria própria)

A classe controladora de estacionamentos (EstacionamentoController), tem as seguintes funções: armazenar a lista de estacionamentos e os filtros de busca que devem ser aplicados; realizar as chamadas de Web Service, usando o Retrofit para o servidor de dados, assim como o processamento do resultado.

A seguir, será apresentado o trecho de código responsável pela requisição de lista filtrada de estacionamentos e pela atualização do fragmento de mapa:

```
public void solicitarListaEstacionamentos() {
    if(restService == null) {
        return;
    }
    restService.listarEstacionamentosFiltrados(getFiltrosParam()).enqueue(new Callback<ArrayList<Estacionamento>>() {
        @Override
        public void onResponse(Call<ArrayList<Estacionamento>> call, Response<ArrayList<Estacionamento>> response) {
            if (response.isSuccessful()){
                setEstacionamentos(response.body());
                mapFragment.createMap();
            } else {
                Log.e( tag: "Mensagem erro", response.message());
            }
        }

        @Override
        public void onFailure(Call<ArrayList<Estacionamento>> call, Throwable t) {
            t.printStackTrace();
            setEstacionamentos(new ArrayList<Estacionamento>());
            mapFragment.createMap();
        }
    });
}
```

Figura 13 - Método solicitar lista de estacionamentos (Fonte: autoria própria)

A representação de cada uma das quatro telas da aplicação ficou a cargo dos fragmentos gerenciados pela atividade principal, dessa forma, os seguintes fragmentos foram implementados:

- *MapFragment*: este fragmento é responsável pela exibição do mapa; dos marcadores de estacionamentos e pela exibição das informações gerais do estacionamento; representa a tela inicial da aplicação, sendo exibida toda vez que o aplicativo é iniciado;
- *DetalhesFragment*: este componente é responsável pela tela de detalhes do estacionamento, exibindo todos os detalhes previstos na fase de análise do projeto. Também é responsável por iniciar as atividades de navegação, para que o usuário possa obter rotas para o local, bem como iniciar estacionamento;
- *FiltrosFragment*: apresenta a tela de filtros para o usuário, seu funcionamento representa a escolha e aplicação dos filtros de estacionamento, personalizando a lista de marcadores que aparecerão na tela inicial. Também é responsável pelas chamadas para o controlador do ReamDB (*RealmController*), a fim de persistir as preferências de filtros escolhidas pelo usuário;
- *EstacionarFragment*: apresenta o componente de tempo *TimePicker* para que o usuário escolha o tempo previsto que permanecerá

estacionado e inicia uma *PendingIntent*, que será executada quando o contador chegar ao zero.

A classe RealmController implementa a conexão e manipulação dos dados na base de dados Realm, executando as consultas e comandos necessários para persistir a listagem de preferências de filtros.

4.1.2 Implementação do servidor

A implementação do servidor se deu através da criação de um banco de dados Postgres, contendo as tabelas necessárias para o funcionamento da aplicação móvel. Em seguida, criou-se um Web Service, utilizando a linguagem de programação Java e o servidor Tomcat.

O banco de dados possui quatro tabelas: estado, cidade, estacionamento e tarifa. A imagem a seguir ilustra o script de criação das tabelas do servidor:

```
CREATE TABLE tb_estado (
    uf varchar(2) PRIMARY KEY,
    nome varchar(100) NOT NULL
);

CREATE TABLE tb_cidade (
    id SERIAL PRIMARY KEY,
    nome varchar(200) NOT NULL,
    uf varchar(2) REFERENCES tb_estado(uf)
);

CREATE TABLE tb_estacionamento (
    id SERIAL PRIMARY KEY,
    nome varchar(500) NOT NULL,
    latitude double precision NOT NULL,
    longitude double precision NOT NULL,
    hr_abertura time NOT NULL,
    hr_fechamento time NOT NULL,
    fl_aceita_cartao boolean NOT NULL,
    fl_bicicletario boolean NOT NULL,
    fl_leve_chave boolean NOT NULL,
    fl_manobrista boolean NOT NULL,
    id_cidade REFERENCES tb_cidade(id)
);

CREATE TABLE tb_tarifa (
    id SERIAL PRIMARY KEY,
    id_estacionamento bigint REFERENCES tb_estacionamento(id),
    hr_inicio float NOT NULL,
    hr_fim  float NULL,
    valor float NOT NULL
);
```

Figura 14 - Script de criação do banco de dados (Fonte: autoria própria)

O Web Service foi implementado utilizando o Eclipse como IDE de desenvolvimento e utilizou a biblioteca Spring para disponibilização do servidor Tomcat e JPA para acesso e manipulação dos dados.

A classe responsável pela exposição dos dados, bem como a camada de banco de dados é a *ServiceController*; possui uma instância de *EstacionamentoDAO*, classe responsável pelas consultas na base de dados.

A imagem a seguir representa um excerto da classe *ServiceController*:

```
public ServiceController() {
    dao = new EstacionamentoDAO();
}

@RequestMapping("/estacionamentos")
public List<EstacionamentoResponse> getEstacionamentos() {
    LinkedHashSet<Estacionamento> estacionamentos = dao.getAll();
    return getResponseList(estacionamentos);
}

@RequestMapping("/estacionamentos/filtered")
public List<EstacionamentoResponse> getEstacionamentos(
        @RequestParam(value = "cartao") Boolean aceitaCartao,
        @RequestParam(value = "manobrista") Boolean manobrista,
        @RequestParam(value = "bicicletario") Boolean bicicletario,
        @RequestParam(value = "chave") Boolean leveChave,
        @RequestParam(value = "permanencia") int permanencia
        ) {

    Calendar now = Calendar.getInstance();
    Date horaInicio = now.getTime();
    Calendar fim = Calendar.getInstance();
    fim.set(Calendar.HOUR_OF_DAY, now.get(Calendar.HOUR) + permanencia);
    Date horaFim = fim.getTime();

    LinkedHashSet<Estacionamento> estacionamentos =
            dao.getAllFiltered(aceitaCartao, bicicletario, leveChave, manobrista, horaInicio, horaFim);
    return getResponseList(estacionamentos);
}
```

Figura 15 - Trecho de código da classe ServiceController (Fonte: autoria própria)

Os dois métodos *getEstacionamentos* expõem uma lista de estacionamentos. O método cuja assinatura não recebe parâmetros é responsável por retornar toda a lista de estacionamentos cadastradas na base de dados; já o de assinatura com parâmetros é responsável pela obtenção de listagem filtrada conforme os parâmetros recebidos.

A implementação da consulta que busca os dados do servidor, os parâmetros booleanos *aceitaCartao*, *manobrista*, *bicicletario* e *leveChave,* que representam filtros dos serviços disponibilizados pelos estacionamentos, só são considerados

caso possuam o valor verdadeiro; caso contrário, o sistema entende que a consulta é indiferente ao valor em questão

4.2 A APLICAÇÃO MÓVEL

Com o nome de *JesParking*, a aplicação móvel desenvolvida como resultado deste trabalho, conforme a proposta, dispôs das funcionalidades propostas apresentando a tela inicial com o mapa e os *pins* indicando os locais de estacionamento. A figura a seguir apresenta a tela inicial do aplicativo:

Figura 16 - Tela inicial do aplicativo (Fonte: autoria própria)

É possível, também, visualizar na imagem anterior a janela de informações gerais do estabelecimento.

A figura a seguir mostra o menu da aplicação móvel:

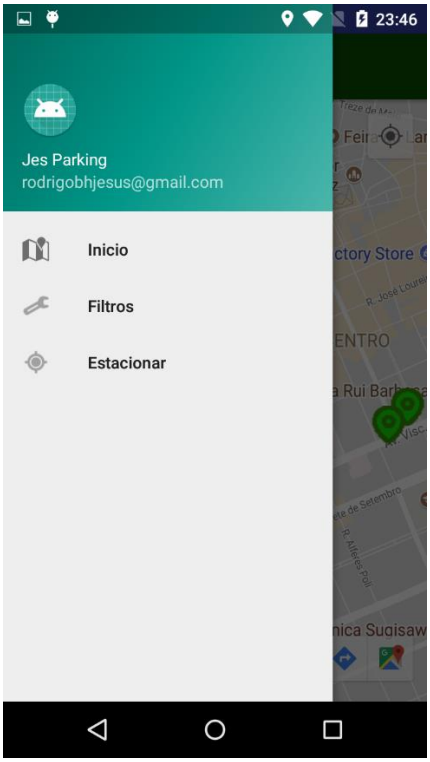

Figura 17 - Menu do aplicativo (Fonte: autoria própria)

A figura a seguir apresenta as opções de filtros disponíveis para o usuário:

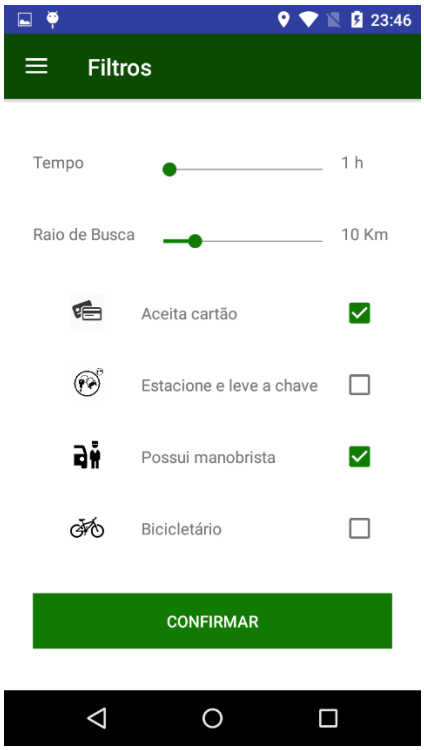

Figura 18 - Tela de filtros da aplicação (Fonte: autoria própria)

A figura a seguir ilustra os detalhes de um estacionamento:

Figura 19 - Tela de detalhes do estacionamento (Fonte: autoria própria)

A figura a seguir mostra a solicitação para abrir a aplicação padrão de navegação do dispositivo móvel, quando o botão Rotas é pressionado:

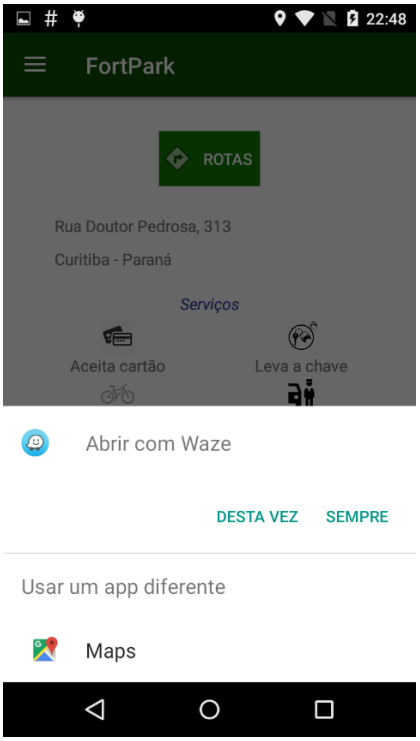

Figura 20 - Ação de obter rotas na tela de detalhes (Fonte: autoria própria)

A tela ‘Estacionar’, cuja função é iniciar uma contagem regressiva para avisar o usuário quando o tempo pré-determinado expirar, exibindo notificações push ao

usuário, alerta vibratório e sonoro, e mensagem *Toast,* a funcionalidade emite um aviso quando o tempo está próximo de finalizar (90% do tempo determinado inicialmente) e outro alerta quando o tempo expira, como é possível visualizar na figura a seguir:

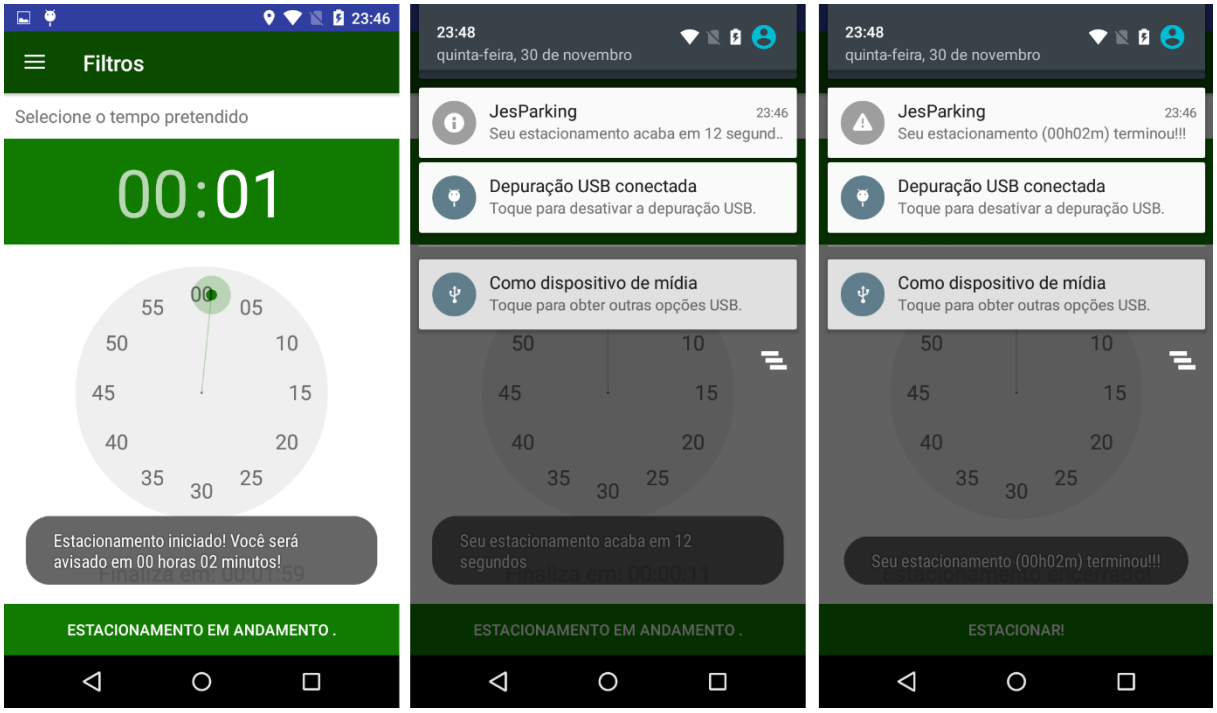

Figura 21 - Tela Estacionar e seu funcionamento (Fonte: autoria própria)

4.3 AVALIAÇÃO DO APLICATIVO

A avaliação da aplicação móvel ocorreu por meio da comparação entre o resultado obtido e as aplicações encontradas na loja de aplicativos da Google (Play Store).

A busca por aplicativos relevantes ocorreu por número de downloads e comentários positivos e, como citado na Introdução deste trabalho, foram encontrados dois aplicativos relevantes: LetsPark e ParkMe.

4.3.1 Comparação com outros aplicativos

Traçando um comparativo entre a aplicação desenvolvida e a aplicação *LetsPark*, é possível verificar que a possibilidade de visualização dos serviços

oferecidos pelo estabelecimento agrega novos valores para o usuário final, uma vez que a *LetsPark* versão 2.0.0 não oferece essas informações.

Este trabalho não possui as seguintes funcionalidades oferecidas pelo *LetsPark*:

- Avaliação dos locais de estacionamento por parte dos usuários;
- Foto do local com o Google Street View na tela de detalhes do estacionamento;
- Possibilidade de escolher o tipo de veículo automotor que o usuário deseja estacionar, como motos, carros de passeio e caminhonetes;
- Possibilidade de contribuição dos usuários, através formulário de cadastro de novos estacionamentos e possibilidade de alertar que o local não existe mais.

É possível, por meio da comparação com o resultado deste trabalho, notar que o ParkMe oferece as funcionalidades que não foram desenvolvidas por este projeto:

- Localize meu carro: marca no mapa o local de estacionamento do carro;
- Reserva: o sistema permite a realização de reserva de vaga em estacionamentos.

Além destas funcionalidades, também é possível verificar que o aplicativo permite filtros personalizados que podem ser aplicados para a busca, mas não há nenhum indicativo nos detalhes do local.

Para ilustrar as diferenças entre cada uma das aplicações, a seguir, é apresentado um quadro comparativo entre suas as funcionalidades:

Tabela 1 - Quadro comparativo entre as funcionalidades dos aplicativos

Este projeto	LetsPark	ParkMe
• **Filtros por aceitação de cartão, por possuir manobrista, por permitir estacionar e levar a chave e por possuir**	• Avaliação de locais; • Imagem do Street View; • Sugestões e correções; • Obtenção de rotas;	• Reserva de estacionamento; • Foto do local; • Opções de ligação e obtenção de direções; • Permite informar o

bicicletário. • Permite o início de cronômetro de estacionamento; • Independe de contribuição da comunidade para funcionar; • Não exige login; • **Permite filtro de raio de busca;** • **Permite salvar as preferências de pesquisa do usuário;** • Obtenção de rotas; • Cronômetro de contagem regressiva; • Exibe horários e preços praticados.	• Pesquisa por endereço; • Formulário de cadastro de local; • Seleção de tipo de automóvel; • Preferência por melhor preço ou andar menos; • Horários e preços praticados.	qução cheio está o local; • Avaliação dos locais; • Horários e preços praticados; • Lista de serviços disponíveis; • Cronômetro de contagem regressiva; • Marcar local de estacionamento; • Avaliação do aplicativo; • Filtros por vários serviços; • Permite filtrar estacionamentos de zona azul; • Filtro por estacionamentos que trabalham com mensalidade; • Exibir estacionamentos em forma de lista; • Permite escolher endereço ou local para busca de estacionamentos nos arredores.

O quadro comparativo anterior ilustra todas as funcionalidades dos aplicativos apresentados e, em negrito, estão dispostas as funcionalidades disponíveis neste projeto que não são contempladas pelos outros.

Para destacar as funcionalidades desenvolvidas por este trabalho, pode-se enumerar:

- Raio de busca: personaliza a distância que a aplicação deve apresentar os estacionamentos em relação à localização do usuário;
- Busca por bicicletário, por manobrista e por permitir levar a chave: permite a aplicação de filtros para que locais de estacionamento diferenciados sejam apresentados;
- Não exigência de login: a aplicação desenvolvida não exige login e não coleta nenhuma informação do usuário.
- Permissões: as únicas permissões que o aplicativo necessita são: Internet para comunicação com o banco de dados central e a localização para traçar as rotas até o destino;
- Salvar preferências: as preferências, ou seja, os filtros aplicados em uma busca anterior são persistidos, de maneira que facilite nova busca do usuário.

Também pode-se observar que as funções de cronômetro e obtenção de rotas são funcionalidades desenvolvidas neste projeto que foram observadas nos dois aplicativos móveis analisados.

4.3.1 Teste de usabilidade

Os testes de usabilidade com os usuários levaram em consideração o *System Usability Scale*, de maneira que um questionário foi criado com dez questões sobre o sistema, além de três questões adicionais: idade do entrevistado, afinidade com tecnologia e sugestões para a aplicação.

Considerando que *r* indica o valor da resposta e q_n indica o número da questão, a seguinte fórmula foi aplicada nas respostas obtidas no questionário (SAURO, 2011) para o cálculo do resultado de *System Usability Scale* por cada usuário entrevistado:

$$Resultado = \left[(r_{q1} - 1) + (5 - r_{q2}) + (r_{q3} - 1) + (5 - r_{q4}) + (r_{q5} - 1) + (5 - r_{q6}) + (r_{q7} - 1) + (5 - r_{q8}) + (r_{q9} - 1) + (5 - r_{q10})\right] \times 2{,}5$$

Quatro pessoas foram entrevistadas e, aplicando a fórmula apresentada anteriormente, a seguir são apresentados os resultados dos questionários:

Tabela 2 - Resultados dos testes de usabilidade

Perfil	Resultado	Sugestões
26 anos, alta afinidade com tecnologia	95	• Permitir verificar em qual estacionamento estou; • Junto com o a notificação push, de acordo com a opção do usuário, gerar widget que apresente qual estacionamento estou e a hora faltante; • Verificar onPause/onStop quando mapa está em modo off-line e é pressionado HOME (botão redondo) ou GERENCIADOR DE TAREFAS (botão quadrado).
50 anos, média afinidade com tecnologia.	77,5	Nenhuma sugestão foi dada pelo entrevistado.
43 anos, baixa afinidade com tecnologia.	92,5	• Poder lançar um endereço destino para localizar estacionamentos próximos.
19 anos, alta afinidade com tecnologia.	87,5	• O marker que mostra a localização poderia ser mais bem apresentado, aumentando a experiência de usuário; • Melhor tratamento de exceções para evitar fechamentos inesperados da aplicação

A partir destes valores, a pontuação calculada do sistema na escala de usabilidade de sistema foi 88,13 pontos, que permite a avaliação positiva neste aspecto da aplicação.

Considerando as sugestões dadas pelos entrevistados, bem como suas idades e afinidade com tecnologia, também é possível verificar o quão usável é o aplicativo móvel pelos usuários, assim como é possível perceber as questões mais pertinentes para melhorar a experiência de usuário, como a apresentação dos marcadores de estacionamento no mapa.

5 CONCLUSÃO

Conforme o projeto seguiu o curso de desenvolvimento, foi possível observar que a aplicabilidade e a necessidade de utilização se mostram pertinentes, uma vez que os testes de usabilidade apontaram que as respostas apresentadas, disponíveis no Apêndice, para a questão 'Eu acho que gostaria de usar esse aplicativo com frequência' foram 'Concordo totalmente' ou 'Concordo parcialmente' por parte das pessoas entrevistadas.

Os pontos positivos deste projeto consistem em:

- Não depender da comunidade ou de *Crowdsourcing* para produção da base de dados inicial da aplicação, permitindo que mesmo que possua poucos usuários, ainda seja possível utilizá-lo;
- Providenciar uma base de dados mais atualizada em comparação a aplicativos que estão sem atualizações na loja de aplicativos do Google;
- Exibir visualmente os serviços oferecidos pelo estabelecimento;
- Não exigir ou solicitar o login de usuário; e,
- Fornecer os filtros: locais que permitem o estacionamento de bicicletas, por manobristas e por oferecer a comodidade de levar a chave, funcionalidades não oferecidas pelos aplicativos analisados no mercado.

No entanto, o projeto não pode ser considerado totalmente inovador, já que existem ferramentas no mercado que oferecem funcionalidades de participação dos usuários, de reserva de estacionamento, imagem do Google Street View, bem como possibilidade de filtros por mais serviços não contemplados por este projeto e pesquisa pelo nome do local de estacionamento desejado.

Também é importante salientar a existência de imprecisão nos resultados, uma vez que o número de pessoas entrevistadas para avaliação da aplicação móvel é pequeno e, ainda que esse conjunto seja composto por diferentes idades e afinidades com tecnologia, um número maior de testes de usabilidade poderia trazer maior carga de confiança nos resultados obtidos de usabilidade.

As sugestões dos usuários foram muito importantes, pois identificaram alguns pontos negativos deste projeto e também forneceram parâmetros para a melhoria da aplicação, inclusive em pontos que não são oferecidos por outros aplicativos existentes no mercado.

A proposta de inclusão da opção de pesquisa por local desejado e resposta em forma de lista de estacionamentos nos arredores do local se mostrou interessante para a evolução do aplicativo, de modo a aliar a busca por estacionamentos à pesquisa por destino pelos usuários.

Do ponto de vista técnico de desenvolvimento, a facilidade de obtenção de respostas para os desafios de implementação, bem como bibliotecas e interfaces de programação de aplicações (APIs), permitiu a rapidez de produção das funcionalidades do aplicativo; sendo possível citar como principais facilitadores para o desenvolvimento:

- O Spring Framework para a minimização de impacto no tempo de configuração de servidor;
- O Realm, para persistência de objetos Java;
- O Hibernate, para gerenciamento de conexão e mapeamento de objetos relacionais do banco de dados;
- O Retrofit, para fácil comunicação e implementação de requisições HTTP para o Web Service disponibilizado.

É possível ainda concluir que os testes e o cuidado com o tratamento de erros são muito importantes, uma vez que o software apresentou algumas falhas que causaram o encerramento da aplicação durante os testes de usabilidade com os usuários.

A definição de trabalhos futuros para este projeto se baseou principalmente na avaliação dos usuários, no não atendimento de interface de usuário que disponibilize o caso de uso Manter Lista de Estacionamentos e na identificação de funcionalidades existentes nos aplicativos analisados na Seção 4.3.1, de maneira que a implementação dos seguintes requisitos foi considerada para evolução da aplicação móvel:

- Pesquisa por destino, podendo ser um endereço, coordenadas ou um local desejado, e a exibição de lista de estacionamentos disponíveis nas imediações do local pesquisado;
- A troca dos marcadores de estabelecimentos, uma vez que foi apontado como um facilitador da experiência de usuário;

- Melhoria na apresentação da janela de informações gerais do estabelecimento, para indicar que é possível obter mais informações sobre o local escolhido;
- O módulo de administração da aplicação, para que o usuário administrador de dados possa realizar manter as informações cadastradas sem execuções de queries direto no banco de dados;
- Ao estacionar, identificar pela tela de detalhes qual é estacionamento, assim facilitando a localização e o gerenciamento de valores de tarifas aplicadas;
- Permitir criação de lista de locais favoritos;
- Criação de *Widget* para visualização de detalhes sobre o estacionamento iniciado;
- Exibição de fotos e imagens do Google Street View nos detalhes do estacionamento;
- Criar ferramenta colaborativa no aplicativo para a informação problemas, sugestões de locais, avaliação dos locais e quaisquer contribuições desejadas pelos usuários;

Por fim, o resultado trabalho pode ser considerado como efetivo, uma vez que os requisitos da aplicação móvel foram identificados, a partir da avaliação de outros aplicativos existentes no mercado e da observação de necessidades de ferramentas para o nicho de mercado específico de Curitiba, e implementados. Além disso, o resultado obtido com o teste de usabilidade, mesmo que tenha sido um teste preliminar, foi positivo.

REFERÊNCIAS

DEITEL, Paul J.; DEITEL, Harvey M. **Java for Developers (Deitel Developer).** 2ª Edição. Nova Jersey: Prentice Hall, 2011.

INTERNATIONAL DATA CORPORATION. **Smartphone OS Market Share, 2017 Q1**. Disponível em: <https://www.idc.com/promo/smartphone-market-share/os>. Acesso em 10 de outubro de 2017.

SILVA, Michel L.; PEREIRA, Lucio C. O. **Android para Desenvolvedores.** 2ª Edição. São Paulo: Brasport, 2012.

SCHIMIGUEL, Juliano. **Gerenciamento de Banco de Dados: Análise Comparativa de SGBD'S**. Disponível em: <http://www.devmedia.com.br/gerenciamento-de-banco-de-dados-analise-comparativa-de-sgbds/30788>. Acesso em 15 de outubro de 2017.

REALM. **Realm Mobile Platform**. Disponível em: <https://realm.io/docs/get-started/overview/>. Acesso em 15 de outubro de 2017.

Be Code. **O que é API? REST e RESTful? Conheça as definições e diferenças!** Disponível em: <https://becode.com.br/o-que-e-api-rest-e-restful/>. Acesso em 15 de outubro de 2017.

W3C. **Geolocation API Specification**. Disponível em: <https://dev.w3.org/geo/api/spec-source.html>. Acesso em 04 de outubro de 2017.

W3C. **Geolocation API Specification 2nd Edition**. Disponível em: <https://www.w3.org/TR/geolocation-API/>. Acesso em 08 de outubro de 2017.

Unified Modeling Language. **What is UML**. Disponível em: <http://www.uml.org/what-is-uml.htm>. Acesso em 02 de outubro de 2017.

Unified Modeling Language. **What is UML**. Disponível em: <http://www.uml.org/what-is-uml.htm>. Acesso em 02 de outubro de 2017.

Universidade Federal de Campina Grande. **UML**. Disponível em: <http://www.dsc.ufcg.edu.br/~sampaio/cursos/2007.1/Graduacao/SI-II/Uml/uml.htm>. Acesso em 15 de outubro de 2017.

FUNDAÇÃO DINARCO REIS. **A questão dos transportes nas grandes cidades**. Disponível em: <https://www.pcb.org.br/fdr/index.php?option=com_content&view=article&id=66:a-questao-dos-transportes-nas-grandes-cidades-&catid=12:dossie-ip>. Acesso em 20 de outubro de 2017.

IPEA. **SIPS Mobilidade Urbana**. Disponível em <http://www.ipea.gov.br/portal/images/stories/PDFs/SIPS/110504_sips_mobilidadeurbana.pdf>. Acesso em 20 de outubro de 2017.

FILHO. Antônio C. E. S. **Cobrança em zona azul: uma possível inconstitucionalidade**. Disponível em: <https://jus.com.br/artigos/24191/cobranca-em-zona-azul-uma-possivel-inconstitucionalidade>. Acesso em 19 de outubro de 2017.

SETRAN. **Total de vagas do EstaR com e sem cobrança**. Disponível em: <http://www.setran.curitiba.pr.gov.br/estar/quadro-de-vagas>. Acesso em 19 de outubro de 2017.

GALINDO. Rogerio W. **Número de carros em Curitiba diminui pela primeira vez em uma década**. Disponível em: <http://www.gazetadopovo.com.br/vida-e-cidadania/numero-de-carros-em-curitiba-diminui-pela-primeira-vez-em-uma-decada-7js0eet0fyvsuowh74i244e1k>. Acesso em 19 de outubro de 2017.

PLAY STORE. **ParkMe Parking**. Disponível em: <https://play.google.com/store/apps/details?id=com.parkme.consumer>. Acesso em 20 de outubro de 2017.

PLAY STORE. **LetsPark**. Disponível em: < https://play.google.com/store/apps/details?id=br.com.letspark.app>. Acesso em 20 de outubro de 2017.
APPSTORE. **LetsPark**. Disponível em <https://itunes.apple.com/br/app/letspark/id695606122?mt=8>. Acesso em 20 de outubro de 2017.

BELLEI. Maria. **A nova ótica da responsabilidade civil dos estacionamentos**. Disponível em: <http://www.direitonet.com.br/artigos/exibir/497/A-nova-otica-da-responsabilidade-civil-dos-estacionamentos>. Acesso em 21 de outubro de 2017.

SCHONARTH. João P. **Em cidade onde falta terreno, quem tem estacionamento é rei**. Disponível em: <http://www.gazetadopovo.com.br/economia/em-cidade-onde-falta-terreno-quem-tem-estacionamento-e-rei-5hurzu70fwz946giu82azwvny>. Acesso em 21 de outubro de 2017.

PLAY STORE. **Mobilicidade**. Disponível em: <https://play.google.com/store/apps/developer?id=Mobilicidade>. Acesso em 21 de outubro de 2017.

DILÃO, Rui. **Sistema de Posicionamento Global**. Disponível em: <http://www.cienciaviva.com/latlong/anterior/gps.asp>. Acesso em: 22 de outubro de 2017.

MACHADO, Everto F. S. **Desenvolvimento de Sistemas de Geolocalização e Rastreamento para a plataforma Android**. Disponível em: <http://repositorio.roca.utfpr.edu.br/jspui/bitstream/1/6914/1/FB_DESIDM_I_2014_01.pdf>. Acesso em: 22 de outubro de 2017.

OLIVEIRA, Celso H. P. **SQL e Programação de Banco de Dados**. Disponível em: <http://www.devmedia.com.br/sql-e-programacao-de-banco-de-dados/3139>. Acesso em: 22 de outubro de 2017.

SQLITE. **Appropriate Uses for SQLite**. Disponível em: <https://sqlite.org/whentouse.html>. Acesso em: 22 de outubro de 2017.

KORTH. Henry F.; SILBERSCHATZ, Abraham. **Sistemas de Bancos de Dados**. 5ª Edição. Rio de Janeiro: ELSEVIER, 2006.

BICALHO, Conrado C. **Bancos de Dados em Dispositivos Móveis**. Disponível em <http://www.decom.ufop.br/guilherme/BCC441/1-2014/seminario_bancos-de-dados-em-dispositivos-moveis.pdf>. Acesso em 22 de outubro de 2017.

SOA WEBSERVICES. **Como funciona os WebServices**. Disponível em: <http://www.soawebservices.com.br/como-funciona.aspx>. Acesso em 22 de outubro de 2017.

IBM. **Diagrama de Componentes**. Disponível em: < https://www.ibm.com/support/knowledgecenter/pt-br/SS4JE2_7.5.5/com.ibm.xtools.modeler.doc/topics/ccompd.html>. Acesso em 19 de novembro de 2017.

SQUARE. **Retrofit**. Disponível em: < http://square.github.io/retrofit/>. Acesso em 26 de novembro de 2017.

DEVMEDIA. **Android Retrofit: Primeiros passos com a Retrofit API**. Disponível em: < https://www.devmedia.com.br/android-retrofit-primeiros-passos-com-a-retrofit-api/31857>. Acesso em 26 de novembro de 2017.

POSTGRESQL. **Sobre o PostgreSQL**. Disponível em: < http://www.postgresql.org.br/pages/sobre-o-postgresql.html>. Acesso em 26 de novembro de 2017.

CAELUM. **Uma introdução prática ao JPA com Hibernate**. Disponível em: < https://www.caelum.com.br/apostila-java-web/uma-introducao-pratica-ao-jpa-com-hibernate/#14-2-java-persistence-api-e-frameworks-orm>. Acesso em 28 de novembro de 2017.

DEVMEDIA. **Spring Boot: simplificando o Spring**. Disponível em: < https://www.devmedia.com.br/spring-boot-simplificando-o-spring/31979>. Acesso em 28 de novembro de 2017.

TECHTUDO. **Apache Tomcat**. Disponível em: < http://www.techtudo.com.br/tudo-sobre/apache-tomcat.html >. Acesso em 28 de novembro de 2017.

APACHE. **Apache Tomcat**. Disponível em: < http://tomcat.apache.org/index.html >. Acesso em 28 de novembro de 2017.

SAURO, Jeff. **Measuring usability with the system usability scale (sus)**. Disponível em: < https://measuringu.com/sus/ >. Acesso em 30 de novembro de 2017.

APÊNDICE

APÊNDICE A – Resultados do Teste de Usabilidade

TESTE DE USABILIDADE 1

Qual sua idade? 43 anos

Qual sua afinidade com tecnologia? Baixa

Sugestões: O marker que mostra a localização poderia ser mais bem apresentado, aumentando a experiência de usuário. Melhor tratamento de exceções para evitar fechamentos inesperados da aplicação

Pergunta	Nota (De 1 a 5) 1 – Discordo totalmente 5 – Concordo totalmente	Valor Calculado
1. Eu acho que gostaria de usar esse aplicativo com frequência.	5	4
2. Eu acho o aplicativo desnecessariamente complexo.	1	4
3. Eu achei o aplicativo fácil de usar.	3	2
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o aplicativo.	1	4
5. Eu acho que as várias funções do aplicativo estão muito bem integradas.	5	4
6. Eu acho que o aplicativo apresenta muita inconsistência.	2	3
7. Eu imagino que as pessoas aprenderão como usar esse aplicativo rapidamente.	5	4
8. Eu achei o aplicativo atrapalhado de usar.	1	4
9. Eu me senti confiante ao usar o aplicativo.	5	4
10. Eu precisei aprender várias coisas novas antes de conseguir usar o aplicativo.	1	4

Resultado obtido: (4 + 4 + 2 + 4 + 4 + 3 + 4 + 4 + 4 + 4) x 2,5 = 92,5

TESTE DE USABILIDADE 2

Qual sua idade? 19 anos

Qual sua afinidade com tecnologia? Alta

Sugestões: O marker que mostra a localização poderia ser mais bem apresentado, aumentando a experiência de usuário. Melhor tratamento de exceções para evitar fechamentos inesperados da aplicação

Pergunta	Nota (De 1 a 5) 1 – Discordo totalmente 5 – Concordo totalmente	Valor Calculado
1. Eu acho que gostaria de usar esse aplicativo com frequência.	5	4
2. Eu acho o aplicativo desnecessariamente complexo.	1	4
3. Eu achei o aplicativo fácil de usar.	4	3
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o aplicativo.	2	3
5. Eu acho que as várias funções do aplicativo estão muito bem integradas.	4	3
6. Eu acho que o aplicativo apresenta muita inconsistência.	1	4
7. Eu imagino que as pessoas aprenderão como usar esse aplicativo rapidamente.	4	3
8. Eu achei o aplicativo atrapalhado de usar.	2	3
9. Eu me senti confiante ao usar o aplicativo.	5	4
10. Eu precisei aprender várias coisas novas antes de conseguir usar o aplicativo.	1	4

Resultado obtido: (4 + 4 + 3 + 3 + 3 + 4 + 3 + 3 + 4 + 4) x 2,5 = 87,5

TESTE DE USABILIDADE 3

Qual sua idade? 50 anos

Qual sua afinidade com tecnologia? Média

Sugestões: Sem sugestões

Pergunta	Nota (De 1 a 5) 1 – Discordo totalmente 5 – Concordo totalmente	Valor Calculado
1. Eu acho que gostaria de usar esse aplicativo com frequência.	4	3
2. Eu acho o aplicativo desnecessariamente complexo.	2	3
3. Eu achei o aplicativo fácil de usar.	3	2
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o aplicativo.	2	3
5. Eu acho que as várias funções do aplicativo estão muito bem integradas.	4	3
6. Eu acho que o aplicativo apresenta muita inconsistência.	2	3
7. Eu imagino que as pessoas aprenderão como usar esse aplicativo rapidamente.	4	3
8. Eu achei o aplicativo atrapalhado de usar.	1	4
9. Eu me senti confiante ao usar o aplicativo.	4	3
10. Eu precisei aprender várias coisas novas antes de conseguir usar o aplicativo.	1	4

Resultado obtido: (3 + 3 + 2 + 3 + 3 + 3 + 3 + 4 + 3 + 4) x 2,5 = 77,5

TESTE DE USABILIDADE 4

Qual sua idade? 26 anos

Qual sua afinidade com tecnologia? Alta

Sugestões: Permitir verificar em qual estacionamento estou.; Junto com o a notificação push, de acordo com a opção do usuário, gerar widget que apresente qual estacionamento estou e a hora faltante; Verificar onPause/onStop quando mapa está em modo off-line e é pressionado HOME(botão redondo) ou GERENCIADOR DE TAREFAS (botão quadrado).

Pergunta	Nota (De 1 a 5) 1 – Discordo totalmente 5 – Concordo totalmente	Valor Calculado
1. Eu acho que gostaria de usar esse aplicativo com frequência.	5	4
2. Eu acho o aplicativo desnecessariamente complexo.	1	4
3. Eu achei o aplicativo fácil de usar.	5	4
4. Eu acho que precisaria de ajuda de uma pessoa com conhecimentos técnicos para usar o aplicativo.	1	4
5. Eu acho que as várias funções do aplicativo estão muito bem integradas.	4	3
6. Eu acho que o aplicativo apresenta muita inconsistência.	2	3
7. Eu imagino que as pessoas aprenderão como usar esse aplicativo rapidamente.	5	4
8. Eu achei o aplicativo atrapalhado de usar.	1	4
9. Eu me senti confiante ao usar o aplicativo.	5	4
10. Eu precisei aprender várias coisas novas antes de conseguir usar o aplicativo.	1	4

Resultado obtido: (4 + 4 + 4 + 4 + 3 + 3 + 4 + 4 + 4 + 4) x 2,5 = 95