Faculdades Integradas de Taquara - Faccat
Av. Oscar Martins Rangel, 4.500.
Taquara, RS, CEP 95600-000

Curso de Sistemas para Internet

VAGASAPP

APLICAÇÃO MOBILE PARA RESERVAR VAGAS EM ESTACIONAMENTOS

Paulo Dreyer de Oliveira
Faculdades Integradas de Taquara – Faccat – Taquara – RS – Brasil
dreyerpaulo89@gmail.com

Flávia Pereira de Carvalho
Professora Orientadora
Faculdades Integradas de Taquara – Faccat – Taquara – RS – Brasil
fpereira@faccat.br

Resumo

Este presente artigo disserta sobre a elaboração do aplicativo VagasApp. Com o alto número de veículos circulando pelas vias, a procura de vagas para estacionar torna-se, muitas vezes, uma tarefa demorada. Pensando nisso, o aplicativo tem como objetivo auxiliar na busca e reserva de vagas em estacionamentos. Desenvolvido em React Native, uma das bibliotecas mais utilizadas para aplicações mobile. O software permite que estacionamentos possam cadastrar suas vagas, efetuando o gerenciamento das mesmas e aos motoristas, ajuda a encontrar a melhor reserva para deixar o seu automóvel.

Palavras-chave: Vagas. Aplicativo Móvel. Estacionamento. Motorista. Reserva.

MOBILE APPLICATION TO RESERVE PARKING SPACES

Abstract

This article has been published about VagasApp application development. With the high number of vehicles on the roads, finding parking spaces is often a time-consuming task. Thinking about that, the application aims to help search and reserve parking spaces. Developed in React Native, one of the most widely used libraries for mobile applications. The software allows parking lots to register their vacancies by making or managing them and drivers, helping to find a better reserve for leaving your car.

Keywords*: Space Available. Mobile App. Parking lot. Driver. Reserve.*

1 INTRODUÇÃO

Um dos principais problemas enfrentados atualmente nas cidades do mundo inteiro, principalmente nos grandes centros urbanos, é a alta concentração de veículos circulando por suas vias. Quanto mais próximo do centro, maior o fluxo de automóveis e consequentemente maior o transtorno gerado por eles. Um dos motivos desse transtorno ocorre em suma, pela necessidade dos motoristas de acharem um lugar para estacionar, que seja perto do seu destino. Por vezes, ao não achar a vaga no lugar desejado, o condutor acaba rodando lentamente pelas ruas, até encontrar um novo local de estacionamento.

Uma grande parcela destes estacionamentos está nas vias públicas, que com a sua ocupação diminui o espaço destinado a movimentação de veículos. Cidades que não possuem sinalizações corretas de suas vagas, podem sofrer problemas em seu trânsito, com motoristas que deixam seus veículos parados em lugares não indicados. Desta forma uma das soluções são os estacionamentos privados que além de serem mais cômodos são ainda mais seguros, já que diminui o risco de acontecer algum dano ao automóvel, como furto, arranhões, amassados, entre outros.

A relação entre a quantidade de estacionamentos nas cidades e o aumento nos deslocamentos feitos de carro é um tópico de discussão característico no planejamento urbano. Vagas em excesso e a um custo relativamente acessível tendem a estimular o uso do automóvel individual, piorar os congestionamentos e aumentar a dependência do carro como meio de transporte. Ainda assim, em todo o mundo, muitas cidades estabelecem, em seus planos diretores, a construção de um número mínimo de vagas de estacionamentos para empreendimentos, tanto residenciais quanto comerciais. De modo geral, a oferta de estacionamentos é vista não como a causa para taxas de motorização cada vez mais altas, mas apenas como uma consequência (PACHECO, 2016).

Com o aumento no número de automóveis circulando pelas ruas das cidades, cresce a demanda por locais de estacionamentos, sejam eles públicos ou privados, gratuitos ou pagos. Motoristas buscam por vagas, onde possam deixar seus veículos perto do seu local de destino, em um lugar seguro, sem um gasto elevado. Através deste cenário surge a demando por meios que possam facilitar a oferta e procura por vagas de estacionamentos.

2 REFERENCIAL TEÓRICO

2.1 **Visão geral sobre estacionamentos**

Os urbanistas são unânimes em dizer que quanto mais espaços se criam para os carros, mais carros aparecem para ocupá-los. Essa constatação é facilmente percebida em cidades brasileiras como Rio de Janeiro, São Paulo e Brasília, onde as taxas de motorização são altas e o tempo médio gasto para ir da casa ao trabalho é 31% maior que em Xangai, Nova York Tóquio e Paris, segundo dados levantados pelo Ipea. E os estacionamentos, em particular, ocupam espaços valiosos das cidades e suas vias, afetando de forma negativa o planejamento urbano (LAMAS, 2014).

Uma pesquisa da EY Consultoria, realizada em 2014, para quantificar as vagas em 15 distritos do centro expandido da capital paulista, mostra que existe lugar para apenas 384 mil carros dos 509 mil que vão para a região diariamente. Ou seja, 125 mil motoristas não conseguem vagas, passando mais tempo no trânsito procurando por elas (LAMAS, 2014).

Os estacionamentos deixaram de ser cômodos para se tornar mais um catalisador dos problemas da mobilidade urbana em um cenário que se repete na maioria das capitais brasileiras: excesso de veículos nas ruas, congestionamentos, índices perigosos de poluição do ar e horas perdidas no trânsito atrás de uma vaga (LAMAS, 2014).

2.1.1 Estacionamento rotativo

Conhecido como sistema de estacionamento rotativo ou zona azul, é um modo de estacionamento onde as vagas são rotativas, ou seja, um veículo pode ficar estacionado em um local, mediante a pagamento de uma taxa, dando o direito de ficar na vaga por um tempo determinado, após esse período o espaço é liberado, possibilitando a outro motorista estacionar na vaga.

Muita gente não gosta do estacionamento rotativo, mas a existência dessas vagas garante que os carros saiam do local e deem espaço aos demais, para que a via pública seja utilizada por todos. Essa é a forma de oferecer uma maior troca de veículos nas vagas, já que nem todos os estabelecimentos comerciais contam com estacionamentos e ter que parar muito longe do local onde você deseja visitar pode se tornar um problema (NASCIMENTO, 2018).

2.1.2 Estacionamento público

Os estacionamentos públicos, são vagas disponibilizadas em locais públicos, de forma gratuita, sem discriminação de utilização. Normalmente não há tempo limite para a ocupação da vaga.

2.1.3 Estacionamento privativo

Estacionamentos privativos, são vagas disponibilizadas em uma área privada, onde somente tem acesso veículos autorizados pelo proprietário do local.

Deverá dar a garantia aos veículos lá estacionados da segurança necessária conforme já definiu o Superior Tribunal de Justiça (STJ) na sua sumula de No 130 que traz no seu bojo que "a empresa responde, perante o cliente, pela reparação de dano, ou furto de veículo ocorridos em seu estacionamento", aplicando-se esta norma a todos os estacionamentos sejam gratuitos ou pagos na forma do que preceitua o Código de Defesa do Consumidor (MELO, 2015).

2.1.4 Disponibilidade de vagas especiais (deficientes, idosos e gestantes)

De acordo com a Lei Brasileira de Inclusão da Pessoa com Deficiência, 2% das vagas de estacionamento da cidade são reservadas para deficientes físicos, e outros 5% para idosos, conforme prevê o Estatuto do Idoso. Ainda não existe uma lei federal específica para vagas de gestantes, mas alguns lugares, como é o caso do estado do Paraná, possui uma lei que determina 2% do total de vagas devem ser destinadas às grávidas ou pessoas com crianças de colo (SHIMOSAKAI, 2017).

Para controlar o uso das vagas de estacionamento reservadas, foram criados cartões de estacionamento, tanto para pessoas com deficiência como para idosos. O cartão é pessoal e não para o veículo, por isso a pessoa que possui o cartão pode utilizá-lo em qualquer veículo onde estiver sendo transportado (SHIMOSAKAI, 2017).

As vagas devem existir em todas as áreas de estacionamento aberto ao público, de uso público ou privado de uso coletivo e em vias públicas. devem estar localizadas próximas aos acessos de circulação de pedestres, e devem estar devidamente sinalizadas (MARTINS, 2018).

3 METODOLOGIA

Para o desenvolvimento deste projeto foram empregados os conceitos e rotinas indicados pela Metodologia Ágil.

Metodologias Ágeis ou métodos ágeis, baseiam-se em uma abordagem incremental para a especificação, desenvolvimento e entrega do software. Eles são mais adequados ao desenvolvimento de aplicativo nos quais os requisitos de sistema mudam rapidamente durante o processo de desenvolvimento. Têm como objetivo reduzir a burocracia do processo (SOMMERVILLE, 2011).

Em 13 novembro de 2001, 17 profissionais se reuniu para discutir como seria possível melhorar o desempenho dos projetos de desenvolvimento de software. A partir desse encontro nasceu o Manifesto Ágil, com os seguintes princípios: (OLIVEIRA, 2015).

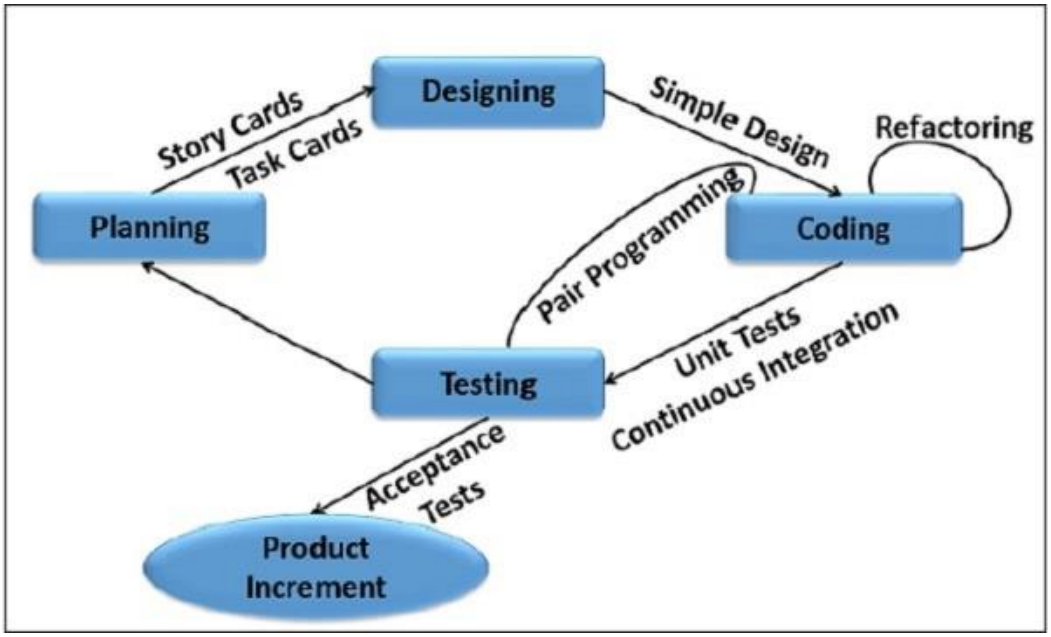
- Indivíduos e interação entre eles mais que processos e ferramentas;
- Software em funcionamento mais que documentação abrangente;
- Colaboração com o cliente mais que negociação de contratos;
- Responder a mudanças mais que seguir um plano.

Entre as várias metodologias ágeis, tais como Kanban, Scrum, Dynamic System Development Model (DSDM), Extreme Programming (XP) e Feature Driven Development (FDD), optou-se pela utilização do método XP.

Em XP, os requisitos são expressos como cenários, que são implementados diretamente como uma série de tarefas (SOMMERVILLE, 2009).

Segundo Cruz (2015), o XP, é uma metodologia ágil direcionada a times de tamanho pequeno e médio que desenvolvem softwares frente a requisitos vagos, desconhecidos ou em mudança constante. Ajuda a criar sistemas de melhor qualidade, produzindo softwares em menos tempo e de forma mais econômica. A Figura 1, mostra um exemplo de como funciona o XP.

Figura 1 – Exemplificação do Extreme Programming.



Fonte: www.ques10.com.

3.1 Análise

A análise do projeto, que resultou na construção do aplicativo VagasApp, foi dividida em etapas.

Na primeira, foi levantado o problema, que consiste na necessidade de auxiliar motoristas que precisam estacionar seus veículos, a encontrar uma vaga de estacionamento. Aos donos destes estabelecimentos, permitir gerenciar as vagas ofertadas.

Na segunda etapa, foram enumerados os requisitos funcionais e não funcionais, necessários para a resolução da aplicação. Com esses requisitos foram elaborados os casos de usos. Ainda nesta fase, foram realizados o diagrama de Caso de Uso e os diagramas de Atividades, referentes aos Requisitos Funcionais (RF).

A terceira fase, foi destinada ao desenvolvimento do software. Para cada RF levantado na etapa anterior, obtiveram suas funcionalidades e particularidades desenvolvidas.

Na última etapa do projeto, a fase de testes, coube testar cada requisito com o desenvolvimento finalizado, caso houvesse a necessidade, voltava-se para a etapa de desenvolvimento, posteriormente retornando para a fase de teste.

3.2 Modelagem

A modelagem de um software implica em criar modelos de software. Um modelo de software captura uma visão de um sistema físico, é uma abstração do sistema com um certo propósito, como descrever aspectos estruturais ou comportamentais do software (GUEDES, 2011).

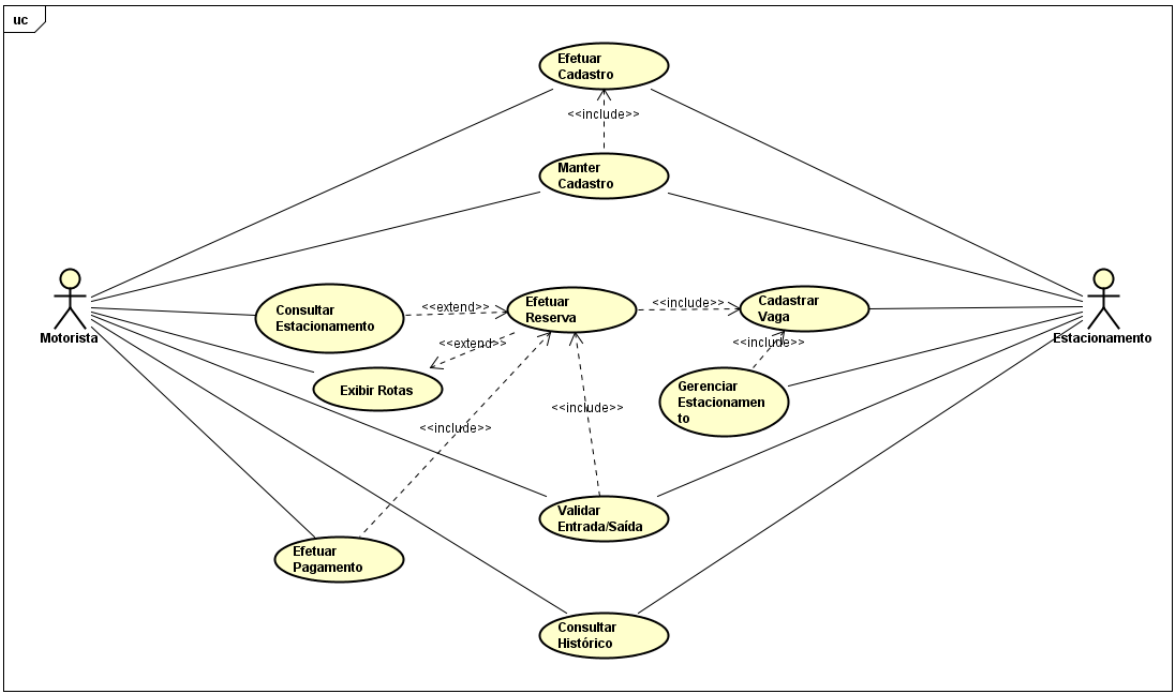
No projeto foi adotada a Unified Modeling Language, conhecida pela sigla UML. Conforme Guedes (2011), disse em seu livro[1], sobre a UML - "É uma linguagem visual utilizada para modelar softwares baseados no paradigma de orientação a objetos. É uma linguagem de modelagem de propósito geral que pode ser aplicada a todos os domínios da aplicação."

Para a modelagem UML dos diagramas foi utilizada a ferramenta Astah Community. Na sequência, será demonstrado o UC da aplicação e Diagramas de Aplicação. De acordo com Guedes (2011), o diagrama de casos de uso é o diagrama mais geral e informal de UML, utilizado normalmente nas fases de levantamento e análise de requisitos do sistema. Apresenta uma linguagem simples e de fácil compreensão para que os usuários possam ter uma ideia geral de como o sistema irá se comportar.

Conforme o autor citado no parágrafo anterior, o diagrama de atividade preocupa-se em descrever os passos a serem percorridos para a conclusão de uma atividade específica, podendo esta ser representada por método com certo grau de complexidade, um algoritmo, ou mesmo por um processo completo. O diagrama de atividade concentra-se na representação do fluxo de controle de uma atividade. Figura 2, Diagrama de Caso de Uso, cada UC representa um requisito funcional do aplicativo projetado.

[1] Livro UML 2 Uma Abordagem Prática.

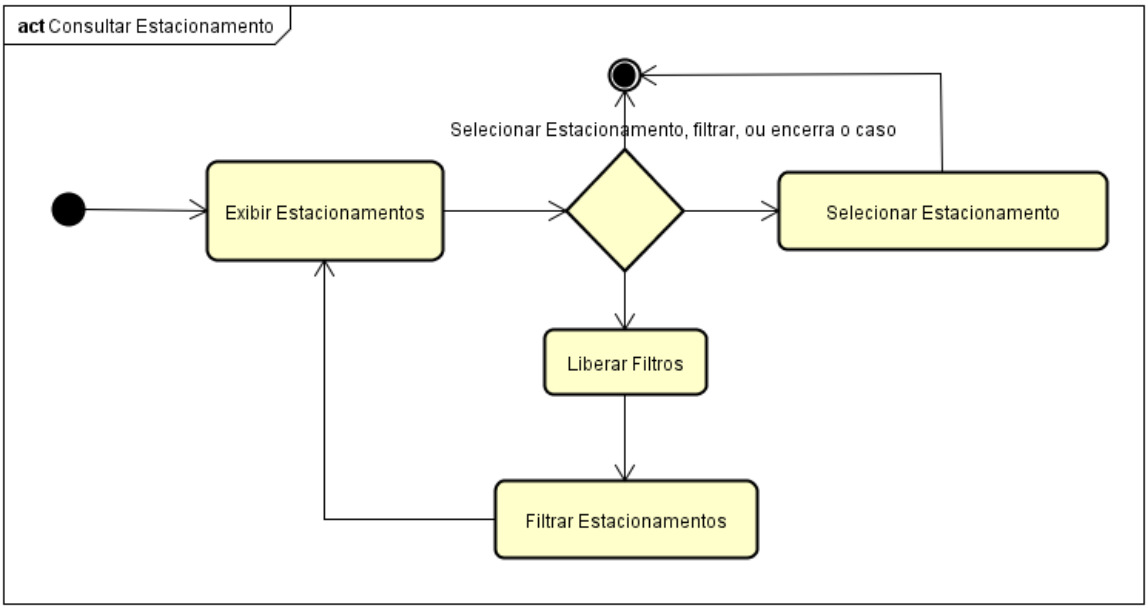
Figura 2 – Diagrama Caso de Uso.



Fonte: Autor.

Figura 3, exibe o diagrama de atividade do caso de uso Consultar Estacionamento. Nele é apresentada a sequência do funcionamento da consulta de estacionamento. Na página inicial do perfil de motorista, apresenta alguns estacionamentos próximos ao local atual do motorista. O usuário pode selecionar alguns filtros, para que somente apareça na pesquisa estacionamentos que atendam a pesquisa.

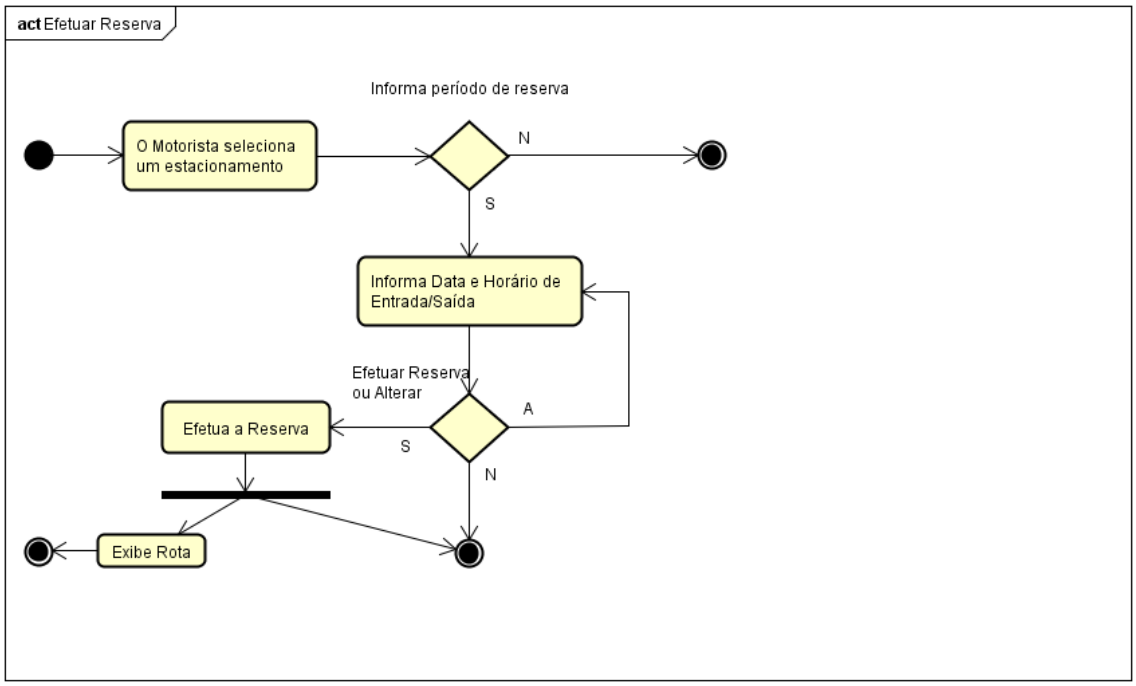
Figura 3 – Diagrama de atividade - Consultar estacionamento.



Fonte: Autor.

Figura 4, diagrama de atividade do requisito Funcional Efetuar Reserva. Ao selecionar estacionamento, o motorista escolhe o dia e horário de reserva e confirma a mesma. Ao confirma a reserva, possibilita ao motorista consulta a rota para o estacionamento.

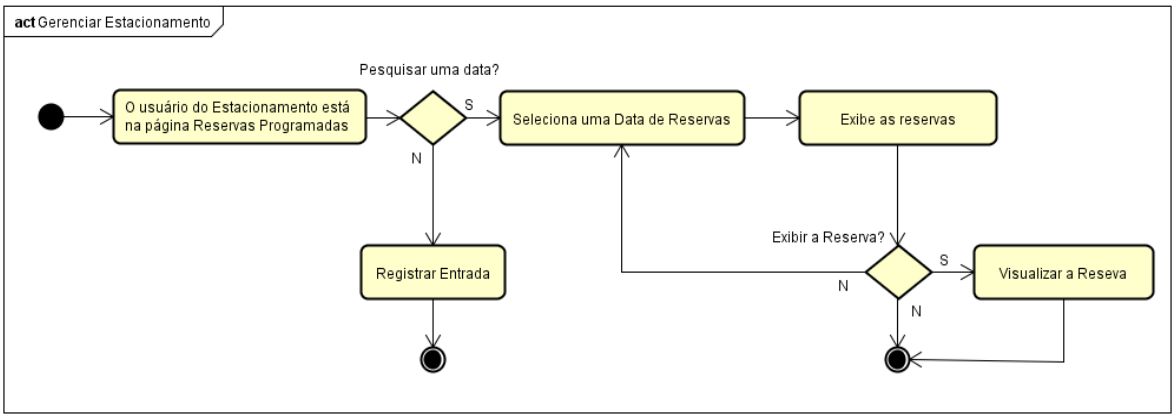
Figura 4 – Diagrama de Atividade – Efetuar Reserva.



Fonte: Autor.

Figura 5, diagrama de atividade do UC, Gerenciar Estacionamento. Na página inicial do aplicativo no perfil de estacionamento, são exibidas as reservas agendadas para o dia atual. O usuário tem a opção de pesquisar outra data, para conferência das reservas, verificar detalhes da mesma ao selecioná-la ou escolher a opção de registrar entrada, que irá para outra página para registro e liberação da entrada de um veículo.

Figura 5 – Diagrama de Atividade – Gerenciar Estacionamento.



Fonte: Autor.

3.3 Desenvolvimento

3.3.1 Arquitetura MVVM

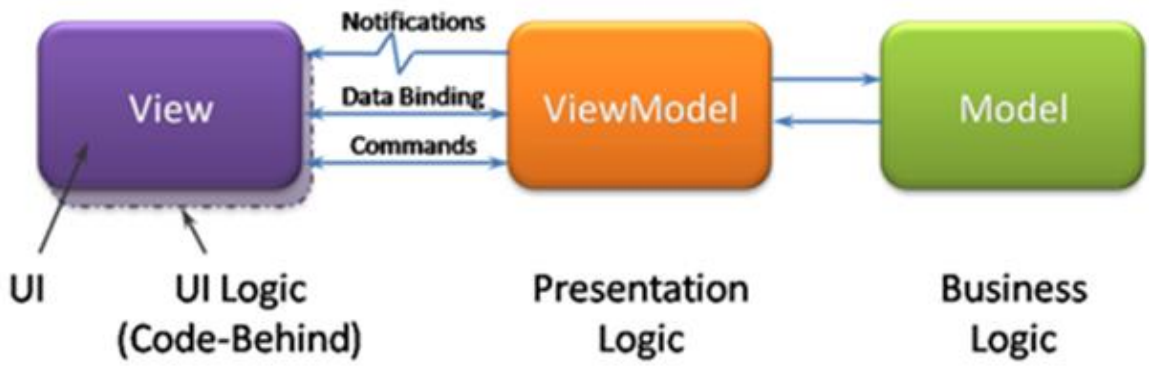
Model-View-ViewModel ou como é conhecida MVVM. Foi criada para resolver o problema do Model View Controller (MVC). Seu objetivo é inserir uma camada chamada de ViewModel, que possui todo o código necessário para que a camada de apresentação, no caso a view, funcione, diminuindo o acoplamento entre as camadas (BASTOS, 2017).

No Model irá conter Data + State + Business Logic[2], de forma não técnica. Pode-se usar como exemplo lógica comercial, acesso a dados e regra de negócios, que não está ligada a View ou Controller e com isto se torna muito reutilizável (NUNES, 2017).

A View liga-se a variáveis e ações expostas pelo ViewModel de forma flexível (NUNES, 2017).

O ViewModel é responsável por expor métodos, comandos e propriedades que mantém o estado da View, assim como manipular a Model com resultados de ações da View e preparar dados Observable necessários para a exibição. Ele também fornece ganchos para que a View passe eventos para o Model. No entanto o ViewModel não está vinculado à View. Existe uma relação de muitos-para-um entre a View e ViewModel, o que significa que uma ViewModel pode mapear muitas Views (NUNES, 2017).

Figura 6 – Exemplo do MVVM.



Fonte: php-faq.com.

[2] Data + State + Business Logic: Representa os dados em seus diferentes estados, com a aplicação da regra de negócio.

3.3.2 React Native

O React Native (RN) é uma estrutura JavaScript (JS) para escrever aplicativos móveis híbridos e nativos para iOS[3] e Android[4]. Ele é baseado no React, a biblioteca JS do Facebook para criar interfaces de usuário, mas em vez de segmentar o navegador, ele segmenta plataformas móveis. Pode-se escrever aplicativos móveis que parecem verdadeiramente "nativos". A maior parte do código pode ser compartilhada entre as plataformas e, portanto, o React Native facilita o desenvolvimento simultâneo para Android e iOS (EISENMAN, 2017).

Aplicações em RN, são escritos com uma mistura de Marcação JS e XML[5]. Utiliza as API[6]s de renderização padrão de sua plataforma hospedeira. Permite acessar recursos da plataforma, como a câmera ou a localização do telefone (EISENMAN, 2017).

React Native apresenta desempenho igual aos aplicativos nativos por conta da sua arquitetura, pois dentro de seu core, que contém um interpretador de Javascript, ele consegue manipular nativamente os componentes do sistema Android/iOS, bem como implementar a lógica da aplicação (ALVES, 2018).

3.3.3 Node JS

Node.js é um runtime de execução JavaScript de código aberto e de multi plataforma. Executa o mecanismo JavaScript V8, o núcleo do Google Chrome, permitindo um excelente desempenho (NODEJS, 2019).

O Node é executado em um único processo, sem criar um novo thread para cada solicitação. Processa diretamente a entrada e saída (E/S), assíncronas em sua biblioteca padrão que impedem o código JavaScript de bloquear (NODEJS, 2019).

Quando o Node.js precisa executar uma operação de (E/S), como leitura da rede, acesso a um Banco de Dados ou sistema de arquivos, em vez de bloquear, retomará as operações quando obtiver a resposta. Isso permite que o Node manipule milhares de conexões simultâneas com um único servidor (NODEJS, 2019).

---

[3] iOS – Sistema Operacional desenvolvido e mantido pela empresa Apple, utilizado em smartphones produzidos pela própria empresa.
[4] Android – Sistema Operacional desenvolvido e mantido pela Google, utilizado em algumas parcas de smartphones.
[5] XML – Linguagem de marcação estendida.
[6] API – Application Programming Interface, permite que sistemas terceiros possam trocar informações entre si.

3.3.4 Firebase

Firebase é uma plataforma de desenvolvimento mobile (e web) adquirida pela Google em 2014. Com foco em ser um back-end completo e de fácil usabilidade, essa ferramenta disponibiliza diversos serviços diferentes que auxiliam no desenvolvimento e gerenciamento de aplicativos (GASPERIN, 2017).

Para utilizar o Firebase, um console web foi criado para facilitar a implementação. Neste, o desenvolvedor adiciona um projeto e inclui os serviços que desejar, cada um com uma explicação de como proceder. Nem todos os serviços são grátis, porém é possível criar um plano conforme as necessidades do desenvolvedor, caso ele precise de algo a mais do que já é oferecido gratuitamente (GASPERIN, 2017).

Esta ferramenta disponibiliza vários tipos de serviços como:

Realtime Database: Banco de Dados hospedado na nuvem. Os dados são armazenados como JSON[7] e sincronizados em tempo real com todos os clientes conectados (FIREBASE 1, 2019).

Authentication: Fornece serviços de back-end fáceis de usar e bibliotecas de Interface prontas para autenticar usuários. Oferece suporte à autenticação por meio de senhas, números de telefone e provedores de identidade como Google, Facebook e Twitter, entre outros (FIREBASE 2, 2019).

Google Analytics para Firebase: Uma solução de análise ilimitada e gratuita que está no núcleo do Firebase. Os recursos do Firebase são integrados ao Analytics, que fornece geração ilimitada de relatórios para até 500 eventos distintos. Os resultados desses relatórios, podem ser levados em conta, para a otimização de aplicações (FIREBASE 3, 2019).

3.3.5 JavaScript

Criado por Brendan Eich, em 1995, na Netscape Communications Corporation, originalmente esta linguagem se chamava LiveScript e Mocha, sendo alterado posteriormente para JavaScript. Tinha o propósito de oferecer aos desenvolvedores formas de tornar determinados processos de páginas web mais dinâmicos, tornando seu uso mais agradável (SILVA, 2015).

[7] JSON – JavaScript Object Notation, é uma forma de armazenar dados em forma de texto, normalmente representado com uma chave (rótulo) e um valor.

Ao invés de rodar remotamente em servidores na internet, o JavaScript tem como característica rodar programas localmente, do lado do cliente. Assim sendo, o JavaScript fornece às páginas web a possibilidade de programação, transformação e processamento de dados enviados e recebidos, interagindo com a marcação e exibição de conteúdo da linguagem HTML e com a estilização desse conteúdo proporcionada pelo CSS nessas páginas (SILVA, 2015).

3.3.6 Visual Studio Code

Como ambiente de codificação foi utilizado o Visual Studio Code (VSCode), um editor de código-fonte, está disponível para Windows, macOS e Linux. Tem suporte embutido para JavaScript, TypeScript e Node.js, além de extensões para várias linguagens, como: C++, C#, Java, Python, Go, PHP, entre outras linguagens (MICROSOFT, 2019).

No desenvolvimento do projeto foi utilizada a versão 1.32.3 para Windows 64bits, utilizando a biblioteca do React Native, que faz uso da linguagem JavaScript. O VSCode possibilita ainda a integração com o GitHub.

4 RESULTADO

O projeto Aplicação Mobile para Reservar Vagas em Estacionamento, teve como finalidade a criação do aplicativo chamado VagasApp. Essa aplicação foi desenvolvida com o propósito de auxiliar motoristas, a encontrarem vagas para estacionar. Para estacionamentos, têm o propósito de ajudar a gerir suas vagas, controlando entrada e saída de veículos, e reservas agendadas.

4.1 Tela de Login

Na tela inicial do app, o usuário tem a opção de se cadastrar com um perfil de motorista ou estacionamento. Após efetuar o cadastro como motorista ou estacionamento, o usuário efetua o login na aplicação utilizando o e-mail e senhas cadastrados. Inicialmente era prevista a possibilidade de conectar utilizando uma conta do Google ou Facebook, mas com a necessidade de cadastrar informações adicionais, como: endereço e Cadastro Nacional de Pessoa Jurídica (CNPJ), para um perfil de estacionamento, ou placa e Cadastro de Pessoa Física (CPF), para o perfil de motorista, foi descartada a possibilidade de efetuar o acesso, pelas redes sociais. Caso o usuário venha a esquecer a senha, ele pode optar pelo Esqueci minha senha, habilitando a redefinição de senha do usuário. Na figura 7 abaixo, é exibida a tela inicial do Aplicativo VagasApp.

Figura 7 – Tela e Login do App VagasApp.

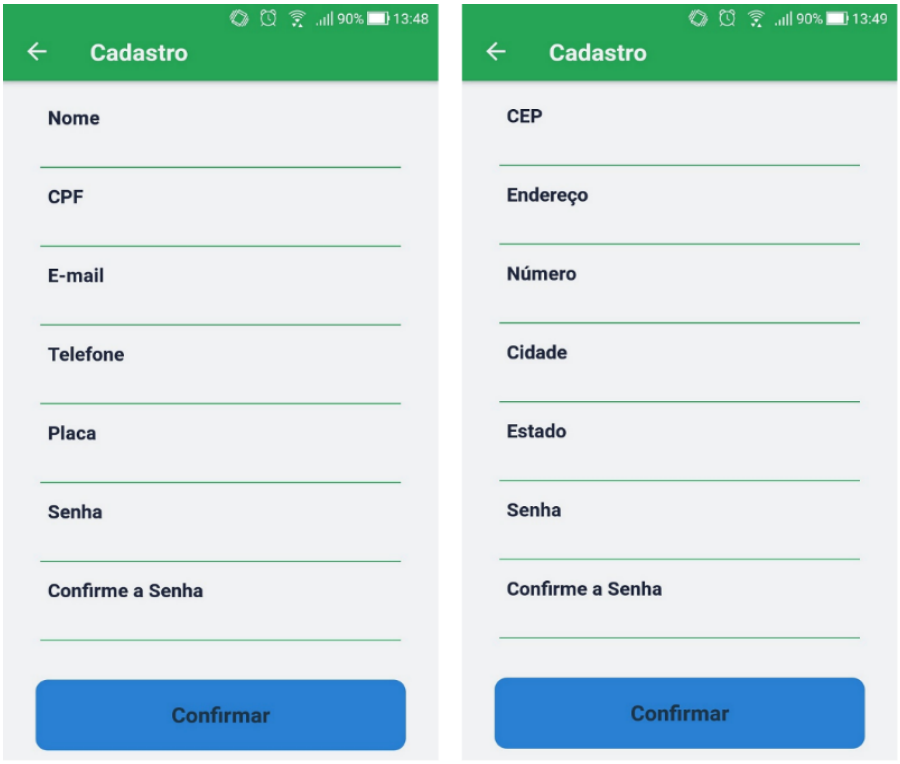
Fonte: Autor.

4.2 Tela de Cadastro

A tela de cadastro foi dividida em duas. Uma para cadastro do perfil de motorista, através da opção: *Procurando uma vaga? Cadastre-se*, ou *Cadastre seu estacionamento. Crie sua conta aqui!*, para o perfil do estacionamento. Ao efetuar o cadastro é obrigatório informar todos os campos para confirmar a criação da conta. Na opção do estacionamento todos os campos são obrigatórios, exceto o número de telefone comercial e número de telefone, mas ao menos um deles deve ser informado. Ao informar um Código de Endereçamento Postal (CEP), no registro do estacionamento, é utilizada a API *viacep.com.br*, que retorna um JSON, com o logradouro do CEP. Ao registrar este tipo de usuário, é feita a chamada da API *maps.googleapis.com/maps/api/geocode/*, que com o endereço indicado no cadastro, é retornada a latitude e longitude.

Ao completar o cadastro é redirecionado o usuário para a tela inicial do app. Na figura 8, traz o layout das telas de cadastro do motorista a esquerda e do cadastro do estacionamento a direita.

Figura 8 Telas de Cadastro de Motorista e de Estacionamento.



Fonte: Autor.

4.3 Perfil Motorista

4.3.1 Tela Inicial

Após o usuário conectar na aplicação com o acesso de motorista, é direcionado para a página inicial. A tela carrega o mapa, que inicialmente exibe para o usuário todos os estacionamentos cadastrados em um raio de 5 km, do seu local atual.

Renderizar o mapa na tela, é possível através do componente *react-native-maps*. Para fazer uso do mapa, é necessária a permissão do usuário para acessar e ativar a localização do dispositivo móvel, que é solicitada na primeira utilização da funcionalidade.

Ao demarcar a localização dos estacionamentos no mapa, pelo componente mencionado no parágrafo anterior, é aplicada a API do Google *maps.googleapis.com/maps/api/directions*, que pesquisa a localização atual do usuário, junto a localização dos estacionamentos, através da latitude e longitude cadastradas, retornando um JSON, com a distância entre os pontos pesquisados. Abaixo na figura 9, a representação da tela inicial do perfil motorista.

Figura 9 - Tela Inicial – Motorista.

Fonte: Autor.

4.3.2 Tela Filtro

Nesta tela possibilita o motorista, escolher alguns filtros, afim de personalizar a procura de vagas. O usuário pode filtrar por: distância, avaliação, preço, vagas especiais (vagas para deficientes ou idosos), estado e cidade. Na seleção dos estados, é utilizada a API *api.londrinaweb.com.br/PUC/Estados*, e na seleção das cidades, a API *api.londrinaweb.com.br/PUC/Cidades*, retornando as cidades do estado solicitado. Ao ter seus filtros selecionados e aplicados, o usuário pode voltar a tela do mapa, que será atualizada. A figura 10 a seguir, mostra a tela de filtro parametrizada.

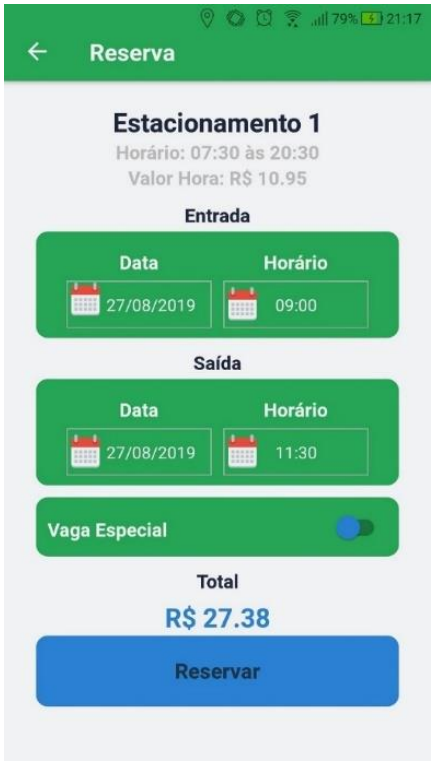
Figura 10 – Tela Filtro.

Fonte: Autor.

4.3.3 Tela Efetuar Reserva

Após o motorista selecionar um estacionamento, ele é direcionado para a tela de reserva de vaga. Nessa tela o usuário pode escolher o dia e horário, que deseja efetuar a reserva. Ao decidir uma data e horário, o aplicativo atualiza o valor a ser cobrado pelo período estipulado. Com a marcação da reserva, é disponibilizada a visualização da rota, do seu local atual até o estacionamento, através da integração com o aplicativo do Google Maps. A figura 11, a seguir mostra uma data e horário escolhidos, com o valor a pagar por este período.

Figura 11 – Tela Efetuar Reserva.



Fonte: Autor.

4.3.4 Tela Consultar Histórico

Nesta tela o motorista consulta os agendamentos confirmados, na sua conta. Quando selecionada uma data, exibe o valor a ser gasto neste dia com as reservas, e o valor total de todas as reservas já efetuadas. Abaixo dos valores são apresentadas todas as reservas programadas para o dia. A figura 12, apresenta a tela do histórico, com a reservas de um dia específico.

Figura 12 – Tela Consultar Histórico.

Fonte: Autor.

4.3.5 Tela Manter Reserva

Na tela Mante Reserva o motorista, tem a autonomia de alterar as reservas, que ainda estejam abertas. Podendo alterar datas e horários, assim como cancelar as mesmas. Através dessa tela o usuário tem a possibilidade de visualizar a rota até o destino da sua vaga, ou validar a entrada ao chegar no estacionamento. Após o motorista validar a saída, habilita uma opção para avaliar o estacionamento, com uma nota entre 0 e 5. Figura 13, exemplo da página Manter Reserva.

Figura 13 – Tela Manter Reserva.

Fonte: Autor.

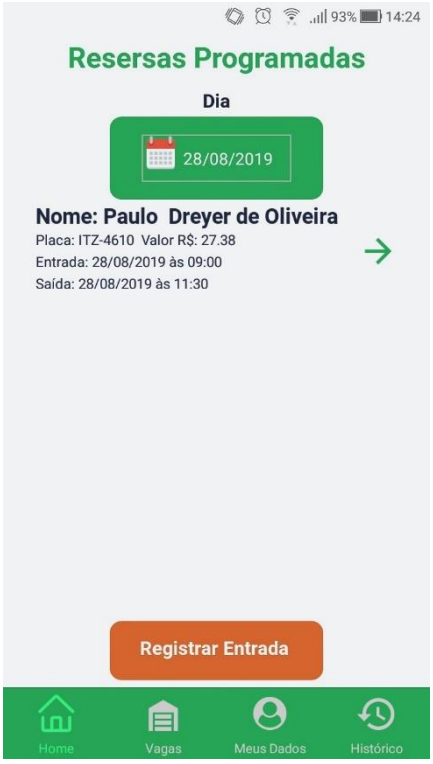
4.4 Perfil Estacionamento

4.4.1 Tela Inicial

Após o usuário conectar na aplicação com o perfil de estacionamento, é direcionado para a página inicial do app. Na página exibe todas as reservas programadas para a data atual. O usuário pode pesquisar uma outra data para verificar agendamentos futuros. Abaixo na tela tem um botão Registrar Entrada, que permite ao estacionamento registrar a entrada de um veículo com a leitura de Qr Code[8]. Abaixo na figura 14, a representação da tela inicial do perfil estacionamento.

[8] Qr Code é um código bidimensional, gerado normalmente a partir de um endereço web ou texto. Esta informação pode lida através de um dispositivo compatível com a leitura.

Figura 14 – Tela inicial do estacionamento.

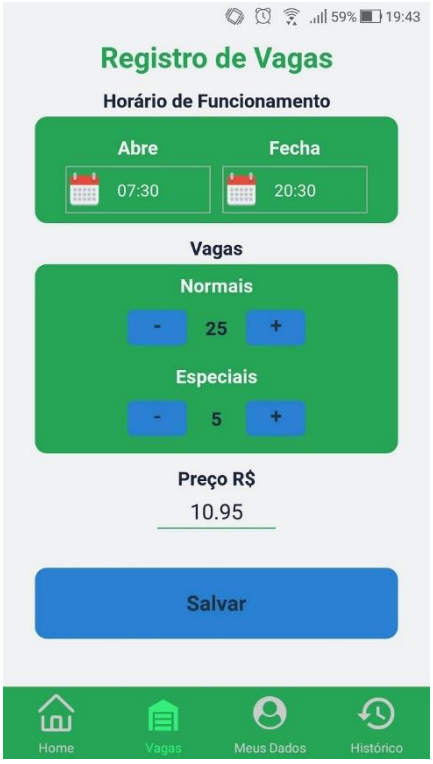


Fonte: Autor.

4.4.2 Tela Manter Vagas

O usuário com o perfil de estacionamento, ao conectar a primeira vez precisa informar o horário de funcionamento, o número de vagas disponíveis e o valor e ser pago, somente após o cadastro destes dados que o seu estacionamento ficará disponível para que motoristas, possam visualizá-los. A qualquer momento o usuário tem a possibilidade de atualizar seus dados. O aplicativo proporciona que o mantenedor do perfil, possa diferenciar as vagas normais de vagas especiais (idosos, portadores de deficiências ou gestantes). A figura 15, apresenta a tela de manter vagas.

Figura 15 – Tela Manter Vagas.



Fonte: Autor.

4.4.3 Tela Visualizar Reserva

Ao selecionar e visualizar uma reserva na tela inicial da aplicação, que exibe as reservas agendadas para o dia. O usuário, poderá conferir os dados da solicitação. Ao estabelecimento somente é permitido cancelar a reserva. Na figura 16, exibe um exemplo da tela de visualização da reserva.

Figura 16 – Tela Visualizar Reserva.

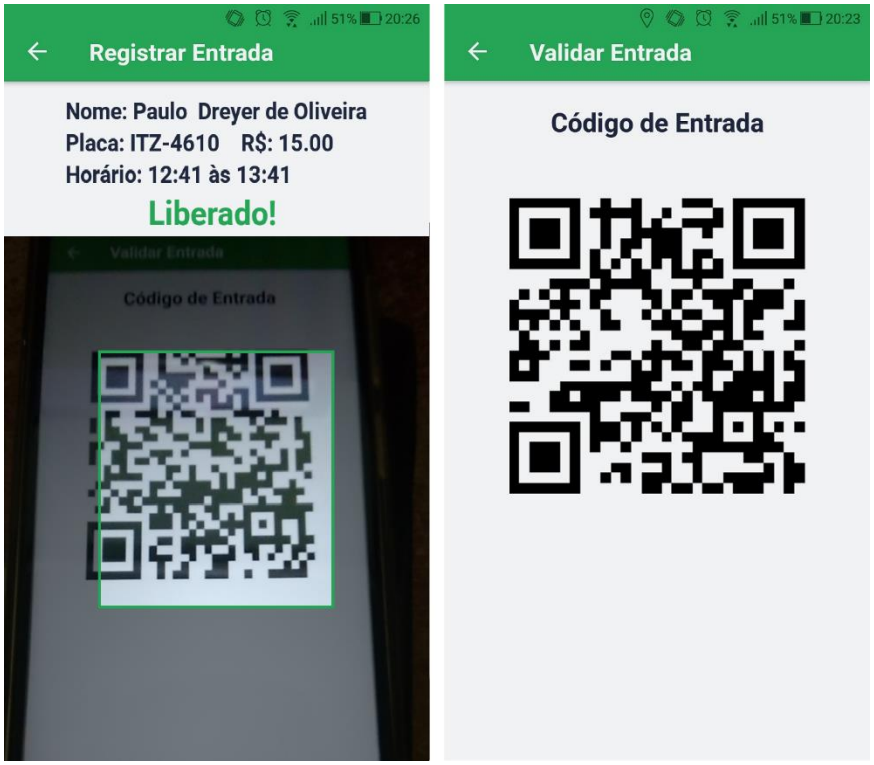
Fonte: Autor.

4.4.4 Tela Registrar Entrada

Quando clicar em registrar entrada na tela inicial, o usuário, é direcionado para a página Registrar Entrada. Ao acessá-la pela primeira vez, é solicitada a permissão para acessar a câmera, somente com essa permissão será possível autorizar a entrada pelo app. Para liberar o acesso o usuário fará a leitura do Qr Code, gerado pela tela de validar entrada do motorista, que gera o código, com o identificador da reserva.

Após a leitura exibe na tela do estacionamento os dados de acesso e a informação de liberado, mas caso o acesso seja negado, exibe na tela o status de rejeitado. A figura 17, exibe a tela para registrar a entrada e a tela com o Qr Code, gerado pelo motorista.

Figura 17 – Tela Registrar Entrada e Validar Entrada.



Fonte: Autor.

5 TESTES

Os testes foram planejados e executados em quatro etapas distintas, utilizando o método da caixa preta, que permite derivar séries de condições de entrada que utilizarão completamente todos os requisitos funcionais de um programa. Este teste tenta encontrar erros de funções incorretas ou faltando, erros de interface, erros em estruturas de dados ou acesso a base de dados, erros de comportamento ou desempenho e erros inicialização ou termino (PRESSMAN, 2011). Foram criadas cinco personas[9], duas com perfil de motorista e três com perfil de estacionamento. Para um dos atores motoristas, foram utilizados dados reais do autor, o segundo motorista, teve informações fictícias. Nos usuários de estacionamento, não foram utilizados dados verdadeiros. As razões sociais, CNPJs, telefones, nenhum deles corresponde a um estacionamento de verdade ou a algum outro tipo de estabelecimento. Assim como suas localizações no mapa, elas foram escolhidas meramente ao acaso para fins demonstrativos, não correspondendo a um estacionamento ali existente.

[9] Personas são personagens hipotéticos criados com o objetivo de reproduzir os usuários de um determinado produto.

Na primeira etapa de teste, a cada finalização do desenvolvimento de um requisito funcional, ele é submetido a testes unitários, verificando as funcionalidades elaboradas, caso necessário volta para desenvolvimento, para realização de eventuais correções. Com as correções aplicadas, volta para o ciclo de teste.

A segunda etapa, é o passo seguinte dos testes, onde requisitos funcionais, que se comunicam são testados em conjunto, certificando se suas relações estão corretas. Conforme a necessidade de ajustes, eles voltam para o desenvolvimento, e após este passo, é realizado todo o teste novamente, desde a primeira etapa, para o requisito ajustado, e a segunda para todos requisitos envolvidos.

A terceira fase, consiste na realização de teste da aplicação como um todo, um teste para cada perfil. Como na etapa anterior, caso haja alguma correção a ser feita, volta para a fase de desenvolvimento, e posteriormente repete-se os testes iniciando na primeira etapa.

Com a conclusão das etapas de testes que eram destinadas ao desenvolvimento do aplicativo, realizou-se o teste de usabilidade com usuários reais, testando ambos os perfis da aplicação. Os retornos obtidos dos usuários, serão aplicados em eventuais atualizações do aplicativo VagasApp.

6 CONCLUSAO

Este artigo dissertou sobre a elaboração do aplicativo VagasApp. Com a separação do software por perfil de motorista e de estacionamento, tem o objetivo de ajudar os usuários motoristas a pesquisar e encontrar vagas nos estacionamentos, e aos estacionamentos, permitir gerenciar essas vagas, controlando a entrada e saída de veículos do seu estabelecimento, como também a quantidade disponível para locação. Tornando-se potencialmente um app, que possa ajudar seus usuários, lhes poupando tempo e dinheiro, ao estacionar seus automóveis. Aos donos dos estabelecimentos proporciona uma melhor administração do seu negócio, ao consegui gerir suas vagas alocadas e disponíveis com o auxílio da aplicação.

Empregado o padrão MVVM, a construção da aplicação foi fundamentada na biblioteca React Native, que utiliza a linguagem JavaScript. Sendo ela uma das mais populares ferramentas multiplataforma para o desenvolvimento mobile. Com auxílio de componentes do React Native e de API’s, principalmente as do Google, foi possível a integração com mapas, localizações e distâncias entre pontos. Ao ser integrado com o Banco de Dados Realtime

Database do Firebase, garante que as informações do sistema, sempre estejam atualizadas, certificando a sua integridade.

Com o eventual aumento do uso do VagasApp, pela adesão de estacionamentos e de motoristas, pode a vir a ajudar a amenizar o trânsito nas cidades, diminuindo o tempo em que motoristas ficam vagando pelas ruas a procura de um lugar para deixar seus veículos, com isso reduzirá o número de automóveis nas ruas, contribuindo para a mobilidade urbana, que hoje é um sério problema, principalmente nos centros movimentados.

REFERÊNCIAS

ALVES, Júlio. **Como funciona o React Native**. Disponível em: https://www.housecursos.com/blog/como-funciona-react-native/. Acesso em: 15 mar. 2019.

BASTOS, Geraldo Castro. **Qual arquitetura mobile escolher?**. Disponível em: https://dextra.com.br/pt/blog/qual-arquitetura-mobile-escolher/. Acesso em: 16 mar. 2019.

CRUZ, Fábio. **Scrum e Agile em Projetos Guia Completo**, Brasport Livros e Multimídia Ltda, Rio de Janeiro, 2015.

EISENMAN, Bonnie. **Learning React Native** Building Native Mobile Apps With JavaScript, 2ª Edição, O'Reilly Media, California, 2017.

FIREBASE 1. **Firebase Realtime Database**. Disponível em: https://firebase.google.com/docs/database/?hl=pt-br. Acesso em: 20 ago. 2019.

FIREBASE 2. **Firebase Authentication**. Disponível em: https://firebase.google.com/docs/auth/. Acesso em: 20 ago. 2019.

FIREBASE 3. **Google Analytics para Firebase**. Disponível em: https://firebase.google.com/docs/analytics/. Acesso em: 20 ago. 2019.

GASPERIN, Carlos Alberto. **Firebase: O Que é e Como Funciona**. Disponível em: http://micreiros.com/firebase-o-que-e-e-como-funciona/. Acesso em: 16 mar. 2019.

GUEDES, Gilleanes T. A. **UML 2** Uma Abordagem Prática, 2ª Edição, Novatec, São Paulo, 2011.

LAMAS, Júlio. **Estacionamentos, os novos vilões da mobilidade urbana**, Disponível em: https://exame.abril.com.br/brasil/estacionamentos-os-novos-viloes-da-mobilidade-urbana/. Acesso em: 06 mar. 2019.

MARTINS, Laura. **Vagas de estacionamento reservadas à pessoa com deficiência: informações importantes**. Disponível em: http://cadeiravoadora.com.br/vagas-de-estacionamento-reservadas-pessoa-com-deficiencia/. Acesso em: 11 mar. 2019.

MELO, João Eduardo. **Estacionamento privado ou público como diferenciar**, Disponível em: https://www.rotaryeclubdistrito4500.com.br/joomla30/index.php/462-Estacionamento-privado-ou-p%C3%BAblico-como-diferenciar. Acesso em: 09 mar. 2019.

MICROSOFT. **Getting Started**. Disponível em: https://code.visualstudio.com/docs. Acesso em: 21 ago. 2019.

NASCIMENTO, Ítalo. **Como funciona o estacionamento rotativo?**, Disponível em: https://www.tricurioso.com/2018/11/14/como-funciona-o-estacionamento-rotativo/. Acesso em: 09 mar. 2019.

NODEJS. **Introduction to Node.js**. Disponível em: https://nodejs.dev. Acesso em: 20 ago. 2019.

NUNES, Filipe. **Android MVC x MVP x MVVM qual Pattern utilizar – Parte 1**. Disponível em: https://medium.com/@FilipeFNunes/android-mvc-x-mvp-x-mvvm-qual-pattern-utilizar-parte-1-3defc5c89afd. Acesso em: 16 mar. 2019.

OLIVEIRA, Bruno Souza de. **Métodos ágeis e gestão de serviços de TI**, Brasport Livros e Multimídia Ltda, Rio de Janeiro, 2018.

PACHECO, Priscila. **Disponibilidade de estacionamento contribui para o aumento da motorização**, Disponível em: http://thecityfixbrasil.com/2016/01/20/disponibilidade-de-estacionamento-contribui-para-o-aumento-da-motorizacao/. Acesso em: 08 mar. 2019.

PRESSMAN, Roger S. **Engenharia de Software** Uma Abordagem Profissional, 7ª Edição, McGraw-Hill Companies, New York, 2011.

SHIMOSAKAI, Ricardo. **Vagas de estacionamento reservadas para pessoas com deficiência, idosos e gestantes**. Disponível em: https://turismoadaptado.com.br/vagas-de-estacionamento-reservadas/. Acesso em: 11 mar. 2019.

SILVA, Giancarlo. **O que é e como funciona a linguagem JavaScript?**. Disponível em: https://canaltech.com.br/internet/O-que-e-e-como-funciona-a-linguagem-JavaScript/. Acesso em: 16 mar. 2019.

SOMMERVILLE, Ian. **Engenharia de Software**, 9ª Edição, Pearson Prentice Hall, São Paulo, 2011.