Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Aplicadas
Departamento de Engenharia de Computação

Trabalho de Conclusão de Curso

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL COM FOCO EM MOBILIDADE URBANA

Daniella Silva Paiva

João Monlevade, MG
2019

Daniella Silva Paiva

DESENVOLVIMENTO DE UM APLICATIVO MÓVEL COM FOCO EM MOBILIDADE URBANA

Trabalho de Conclusão de curso apresentado à Universidade Federal de Ouro Preto como parte dos requisitos para obtenção do Título de Bacharel em Engenharia de Computação pelo Instituto de Ciências Exatas e Aplicadas da Universidade Federal de Ouro Preto.
Orientador: Prof.^{o} George Henrique Godim da Fonseca, Dr.

Universidade Federal de Ouro Preto
João Monlevade
2019

P149d Paiva, Daniella Silva.
Desenvolvimento de um aplicativo móvel com foco em mobilidade urbana [manuscrito] / Daniella Silva Paiva. - 2020.

44f.:

Orientador: Prof. Dr. George Henrique Godim da Fonseca.

Monografia (Graduação). Universidade Federal de Ouro Preto. Instituto de Ciências Exatas e Aplicadas. Departamento de Computação e Sistemas de Informação.

1. Aplicativos móveis. 2. Software de aplicação. 3. Android (Recurso eletrônico). 4. Mobilidade urbana. I. Fonseca, George Henrique Godim da. II. Universidade Federal de Ouro Preto. III. Titulo.

CDU: 004.9

Catalogação: ficha.sisbin@ufop.edu.br

MINISTÉRIO DA EDUCAÇÃO
UNIVERSIDADE FEDERAL DE OURO PRETO
REITORIA
INSTITUTO DE CIENCIAS EXATAS E APLICADAS
DEPARTAMENTO DE COMPUTACAO E SISTEMAS

FOLHA DE APROVAÇÃO

Daniella Silva Paiva

DESENVOLVIMENTO DE UM APLICATIVO
MÓVEL COM FOCO EM MOBILIDADE
URBANA

Membros da banca

George Henrique Godim da Fonseca - Doutor em Engenharia Elétrica - UFOP
Bruno Cerqueira Hott - Mestre em Ciência da Computação - UFOP
Lucinéia Souza Maia - Doutora em Gestão e Organização do Conhecimento - UFOP

Versão final
Aprovado em 13 de Dezembro de 2019

De acordo

George Henrique Godim da Fonseca (orientador)

Documento assinado eletronicamente por **George Henrique Godim da Fonseca**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 06/01/2020, às 20:44, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0030622** e o código CRC **CED07CF2**.

Referência: Caso responda este documento, indicar expressamente o Processo nº 23109.000051/2020-81 SEI nº 0030622

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: - www.ufop.br

Dedico este trabalho a todos os seres de luz que me acompanharam nessa trajetória e a poderosa mulher selvagem que em mim habita e guia meus passos.

Resumo

Este trabalho apresenta o desenvolvimento de uma aplicação Android com foco em mobilidade urbana, voltada para mototaxistas que trabalham diariamente com transporte de pessoas, inicialmente aplicado na cidade de João Monlevade. O trabalho tem como objetivo especificar, projetar e implementar um aplicativo Android, com base nos conhecimentos adquiridos no curso que conecte um mototaxista ocioso ao passageiro que precisa se deslocar com praticidade. O resultado do trabalho é uma aplicação de interface de usuário amigável e prática que permite ao mototaxista receber uma notificação quando uma solicitação de nova corrida é requisitada. Os resultados alcançados foram satisfatórios e demonstram que as estratégias adotadas para o desenvolvimento da aplicação foram adequadas e que é possível a implementação do sistema proposto.

Palavras-chave: Mobilidade urbana. Android. Aplicativos móveis.

Abstract

This paper presents the development of an application focused on urban mobility, aimed at motorcycle taxi drivers working daily with people transportation, initially applied in the city of João Monlevade. The paper has the objective to specify, design and implement an Android application, based on the knowledge acquired during the undergraduate course, that connects an idle motorcycle taxi driver with the passenger who needs to travel with practicality. The result of the paper is a practical user-friendly interface application that allows the motorcycle taxi driver to receive a notification when a new ride is requested. The results achieved were satisfactory and demonstrate that the strategies adopted for the application development were adequate and that the proposed system implementation is possible.

Keywords: Urban mobility. Android. Mobile apps.

Lista de ilustrações

Lista de abreviaturas e siglas

API	*Application Programming Interface*
IU	*Interface de Usuário*
JSON	*JavaScript Object*
JVM	*Java Virtual Machine*
NoSQL	*No Structured Query Language*
SDK	*Software development kit*
XML	*Extensible Markup Language*

Sumário

1 Introdução

A mobilidade nos centros urbanos se tornou um desafio para governos, cidadãos e corporações. Em um planeta individualista com mais de 7 bilhões de pessoas, a forma com que nos transportamos se tornou insustentável, derivando problemas ambientais e sociais (FIGUEIRA, 2015). Além dos problemas ambientais e sociais, cerca de 96% das cidades brasileiras não têm um plano de transporte, com isso os municípios não levam em conta as reais necessidades dos usuários na hora de investir em mobilidade urbana, revela o IBGE (2013). Dessa forma, o transporte urbano coletivo se apresenta ineficiente de várias maneiras, impactando de forma negativa o passageiro que deseja se deslocar de um ponto ao outro. Em geral, os passageiros enfrentam problemas como tempo de espera e deslocamento, superlotação, além de ter que se locomover até o ponto de ônibus mais próximo.

Uma das alternativas para desviar dos problemas encontrados no transporte público e da necessidade de comprar um automóvel, é o novo modelo de negócio criado pela Uber em 2009, que rompeu barreiras culturais e tornou mais acessível o uso de transporte particular, oferecendo uma maneira simples de conectar um passageiro a um motorista disponível (UBER, 2009). Várias empresas seguiram o modelo de negócio proposto pela Uber e foram lançados no mercado em pouco tempo, por exemplo: Cabify, 99, Lyfti, Piloto 31, entre outros que podem ser encontrados na *Google Play*.

Diante da necessidade de locomoção das pessoas no dia a dia e o desenvolvimento crescente de aplicativos que propõem soluções rápidas e práticas para a mobilidade urbana, este trabalho tem como objetivo a implementação de um aplicativo de mobilidade urbana onde será desenvolvido apenas a aplicação para o mototaxista. O aplicativo móvel visa conectar o mototaxista ocioso com o passageiro que precisa se deslocar com praticidade, na cidade de João Monlevade.

1.1 Objetivos

O objetivo geral do presente trabalho consiste em desenvolver um aplicativo em Android com foco em mobilidade urbana para os mototaxistas, que conecta de maneira fácil e prática um mototaxista disponível ao passageiro que deseja se deslocar.

1.1.1 Objetivos Específicos

Os objetivos específicos do presente trabalho são:

- Aplicar os conhecimentos adquiridos ao longo do curso;

- Estudar sobre mobilidade urbana;
- Especificar, projetar e implementar o aplicativo do mototaxista para Android, com base nos conhecimentos adquiridos no curso que possibilite um mototaxista ocioso aceitar uma solicitação de transporte de um passageiro.
- Integrar a aplicação com os serviços do *Google Maps* e *Firebase*;

1.2 Justificativa

O presente trabalho tem como justificativa desenvolver uma plataforma tecnológica que proporcione aos mototaxistas um ambiente mais seguro e prático para receber chamadas de corridas na cidade de João Monlevade, onde ainda não existe nenhuma aplicação semelhante.

1.3 Estrutura do Trabalho

O conteúdo deste trabalho foi organizado em cinco capítulos: Introdução, Referencial Teórico, Metodologia, Resultados, Considerações finais. O primeiro capítulo trata da introdução ao tema, a justificativa e os objetivos. No segundo capítulo, é apresentada uma revisão da literatura sobre o sistema operacional Android e os conceitos da coleção Android Jetpack. O terceiro capítulo descreve a metodologia apresentando a visão geral do sistema, as ferramentas e tecnologias utilizadas, os diagramas e o banco de dados, detalhando o processo de concepção do aplicativo. O quarto capítulo apresenta os resultados obtidos com o trabalho e por fim, no quinto capítulo são apresentadas as considerações finais e as melhorias subsequentes para os trabalhos futuros.

2 Referencial Teórico

Este capítulo contempla a fundamentação teórica para a implementação do trabalho, discorrendo sobre conceitos, ferramentas e tecnologias utilizadas.

2.1 Sistema Operacional Android

O sistema operacional Android é um sistema Linux multiusuário que implementa o princípio do privilégio mínimo, ou seja, cada aplicativo do sistema Android é um usuário diferente, que por padrão, tem acesso somente aos componentes necessários para a execução do seu trabalho, tornando o ambiente mais seguro.

O comportamento geral da aplicação é definido a partir da construção dos componentes de aplicativo. Cada componente representa uma entidade independente e cada tipo de componente tem um ciclo de vida específico. O ciclo de vida define a forma pela qual o componente será criado e destruído quando não for mais necessário, ou quando o sistema precisar recuperar memória para outros aplicativos. Como ilustra a Figura 1, existem quatro tipos de componentes no Android:

- Atividades: representam uma tela única com uma interface do usuário. Funciona como uma porta de entrada pelo qual o sistema pode acessar seu aplicativo. Ao contrário dos aplicativos na maioria dos outros sistemas, os aplicativos do Android não têm nenhum ponto de entrada único (não há a função main()).

- Serviços: são componentes executados em segundo plano para realizar operações de execução longa ou para realizar trabalho para processos remotos. Eles não apresentam uma interface do usuário.

- Provedores de conteúdo: gerenciam um conjunto compartilhado de dados do aplicativo. É possível armazenar os dados no sistema de arquivos, em um banco de dados SQLite ou em qualquer local de armazenamento persistente que o aplicativo possa acessar.

- Receptores de transmissão: são componentes que respondem a anúncios de transmissão por todo o sistema. Muitas transmissões se originam do sistema — por exemplo, uma transmissão que anuncia que uma tela foi desligada, a bateria está baixa ou uma tela foi capturada.

As Atividades, Serviços e Receptores de transmissão, são ativados por *intents* que conectam componentes individuais com outros componentes em tempo de execução. Um objeto *Intent* (intenção) define uma mensagem assíncrona para requisitar uma ação específica de outros componentes, podendo ser explícito ou implícito. Os Provedores de

Figura 1 – Blocos de construção do núcleo Android

Fonte: Luiz Tools (2011)

conteúdo são ativados por uma requisição de um *Content Resolver* (Resolvedor de conteúdo). O *Content Resolver*, a partir de uma camada abstrata, manipula todas transações diretas entre o componente que requisitou a informação e provedor de conteúdo.

Para que um componente de aplicativo seja iniciado pelo sistema Android, é necessário que o sistema saiba da existência do componente. Portanto, o aplicativo deve declarar todos os seus componentes no arquivo AndroidManifest.xml, que deve estar na raiz do diretório do projeto do aplicativo.

Além dos componentes de aplicativos, os aplicativos Android exigem recursos separados do código fonte, como imagens, cores, animações, arquivos de áudio e tudo o que se relaciona com a apresentação visual do aplicativo. O uso de recursos de aplicativo facilita a atualização de diversas características do aplicativo e diferentes configurações de dispositivos, por exemplo: troca de idiomas e tamanhos de tela diferentes. Os recursos de aplicativo e o layout das interfaces do usuário devem ser definidos em arquivos XML (GOOGLE DEVELOPERS, 2018a).

2.2 Android Jetpack

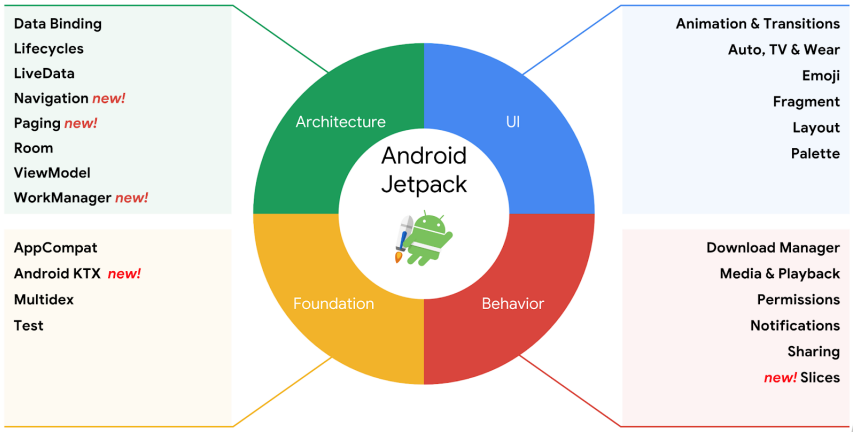
O Android Jetpack consiste em uma coleção de componentes de software Android, lançado em maio de 2018, para ajudar no desenvolvimento de aplicativos Android. A coleção Jetpack é um conjunto de componentes, ferramentas e orientações, oferecidos na forma de bibliotecas "não empacotadas" e construídos com base em práticas de projeto modernos, como separação de problemas e capacidade de teste.

A coleção Jetpack, apresentada na Figura 2, é constituída por quatro categorias de componentes:

- Base: fornecem os recursos principais do sistema e uma estrutura para testes de Interface de Usuário (IU) de unidade e tempo de execução.

- Arquitetura: classes que ajudam a gerenciar o ciclo de vida do componente de IU, lidar com a persistência de dados, navegação e tarefas em segundo plano.
- IU: simplificam a tarefa de tornar fácil e agradável o uso do aplicativo;
- Comportamento: ajudam no desenvolvimento de aplicativos robustos, testáveis e de fácil manutenção.

Figura 2 – Android Jetpack



Fonte: Google Developers (2018b)

Cada componente do Jetpack é adaptável individualmente e pode se encaixar aos demais como um quebra-cabeça, de acordo com a necessidade da aplicação. Para o desenvolvimento do presente trabalho, foram combinados Componentes de Arquitetura e de UI, seguindo as práticas recomendadas pelo Guia de Arquitetura, para garantir o desenvolvimento de um aplicativo modular e de fácil manutenção, com cada classe responsável por uma função bem definida.

2.2.1 Componentes de Arquitetura

Os Componentes de Arquitetura do Android Jetpack, fornecem orientação sobre a arquitetura de aplicativos, através de um conjunto de bibliotecas que estruturam o aplicativo de maneira robusta, testável e sustentável. As bibliotecas dos Componentes de Arquitetura seguem os princípios de separação de interesses, baixo acoplamento, o observador padrão e inversão de controle. A seguir, os componentes utilizados no desenvolvimento da aplicação:

- *Lifecycles*: permite criar componentes que reconhecem o ciclo de vida de uma atividade ou fragmento, e executem ações em resposta a mudança no estado do ciclo de vida que está sendo observado;
- *LiveData*: é uma classe de suporte de dados da biblioteca *Lifecycle* mantida pelo *ViewModel*, armazena dados observáveis e notifica os componentes observadores que

houve alteração nos dados, sempre respeitando o estado do ciclo de vida desses componentes.

- *Navigation*: ajuda a implementar a navegação gerenciando as interações que permitem aos usuários navegar, entrar e sair das diferentes partes do conteúdo do seu aplicativo de forma consistente e previsível;

- *View Model*: é responsável pela comunicação entre o Repositório e a IU, exercendo o princípio da responsabilidade única. Suas instâncias sobrevivem às mudanças na configuração da IU, sempre considerando o ciclo de vida do componente. Armazena o estado dos dados de uma *View* e gerencia os objetos *LiveData*.

2.2.2 Interface de Usuário

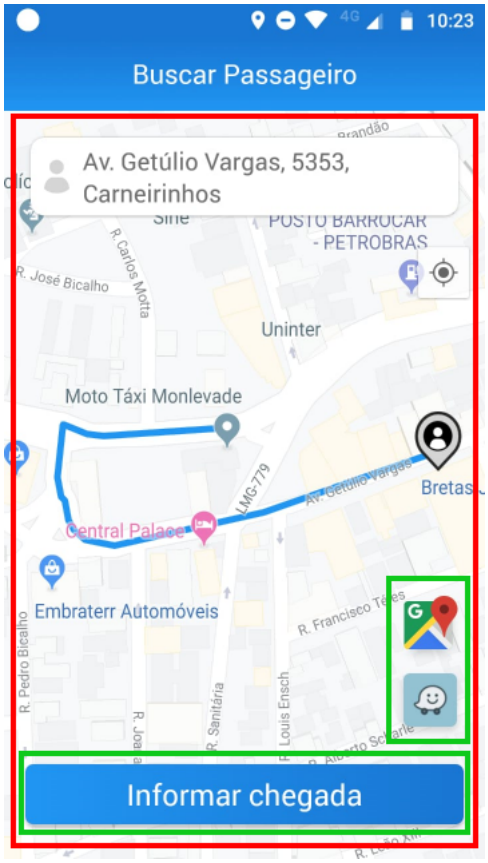
Os componentes de IU foram criados para simplificar o desenvolvimento da interface do usuário, além de fornecerem widgets e assistentes que tornam a aplicação fácil e agradável de usar. A seguir, os componentes de IU utilizados no desenvolvimento da aplicação:

- *Animation* e *Transitions*: As animações são utilizadas para fornecer *feedback* ao usuário quando a interface mudar de estado devido a alguma ação do mesmo, sendo possível adicionar pistas visuais que notificam caso um conteúdo novo estiver sendo carregado ou quando novas ações são disponibilizadas. Além de permitir animar facilmente as alterações entre duas hierarquias de exibição, as animações e transições adicionam uma aparência refinada ao aplicativo;

- *Fragment*: Os fragmentos foram introduzidos no Android, na API de nível 11, para suportar aplicações de IU flexíveis e dinâmicas. Um *Fragment* é usado para representar o comportamento ou uma parte da interface do usuário, sendo possível combinar vários fragmentos em uma mesma interface. No presente trabalho, a tela principal contém dois fragmentos: o mapa de visualização do motorista em um fragmento, e as ações disponíveis mais o *status* da corrida em outro fragmento que sobrepõe o mapa. Na Figura 3, o retângulo vermelho representa o fragmento que infla o mapa, enquanto os retângulos verdes representam o fragmento que mostra ao usuário as ações disponíveis.

- *Layout*: é usado para definir a estrutura de uma interface com os elementos que o usuário pode interagir. É possível declarar o *layout* de duas formas: instanciando os elemento em ambiente de execução ou declarando os elemento da IU no XML. O vocabulário XML, facilita a projeção e a criação de *layouts* diferentes para diferentes orientações e tamanhos de tela. É possível também declarar os *layouts* da aplicação em XML e modificar no ambiente de execução, para melhor apresentação da IU.

Figura 3 – Fragmentos



Fonte: elaborado pela autora

A implementação desses componentes evita vazamento de memória, manipulação manual do ciclo de vida, falha devido a atividades interrompidas e garante que a interface do usuário sempre corresponda ao estado de dados mais recente.

2.3 Trabalhos Relacionados

Nesse capítulo são apresentados alguns trabalhos relacionados existentes que foram utilizados como referência para elencar as principais funcionalidades da solução proposta.

2.3.1 Uber

Fundada em Março de 2009, a *Uber Technologies Inc* é a pioneira no desenvolvimento de soluções móveis para serviço de transporte. A empresa conta com mais de 500.000.000 instalações do aplicativo de passageiro e mais de 50.000.000 do aplicativo de motorista, ambos na *Play Store*.

Passageiro: com um cadastro simples, o passageiro consegue logar no aplicativo através do seu número de celular. Para solicitar uma corrida é necessário inserir a origem, o

destino final e escolher o método de pagamento - cartão, dinheiro, *Paypal* e Uber pré-pago. Após esses passos, basta confirmar e aguardar o motorista.

Motorista: com um cadastro mais burocrático, o motorista precisa aguardar a aprovação para começar a efetuar corridas. Após estar logado, é necessário habilitar o botão de receber solicitações para as notificações de nova corrida. Quando em viagem, o motorista pode escolher a forma que deseja navegar: *Waze*, *Google Maps* ou o próprio aplicativo da Uber. O mapa próprio é o grande diferencial da empresa, o que os dá liberdade para otimizá-lo da maneira mais conveniente para seus usuários.

Ambos os aplicativos são rápidos e simples, porém quando em situações de baixa qualidade de conexão de internet ou aparelhos de configurações inferiores, o aplicativo perde produtividade. Para contornar a sistuação, a empresa lançou o Uber Lite, que para reduzir o consumo de recursos do dispositivo, o mapa não é carregado quando a aplicação está em execução.

2.3.2 Piloto31

O aplicativo Piloto31, foi lançado pela primeira vez em outubro de 2017, na cidade de Belo Horizonte, MG. Assim como o aplicativo Uber, o aplicativo Piloto31 é uma plataforma intermediadora para prestação de serviços de mobilidade urbana, porém os serviços são feitos por mototaxistas. O aplicativo Piloto31 atua entre o motociclista e o cliente, que pode solicitar transporte de pessoas ou entrega de encomendas. O aplicativo do motociclista parceiro, conta com mais de 1000 instalações na *Play Store*.

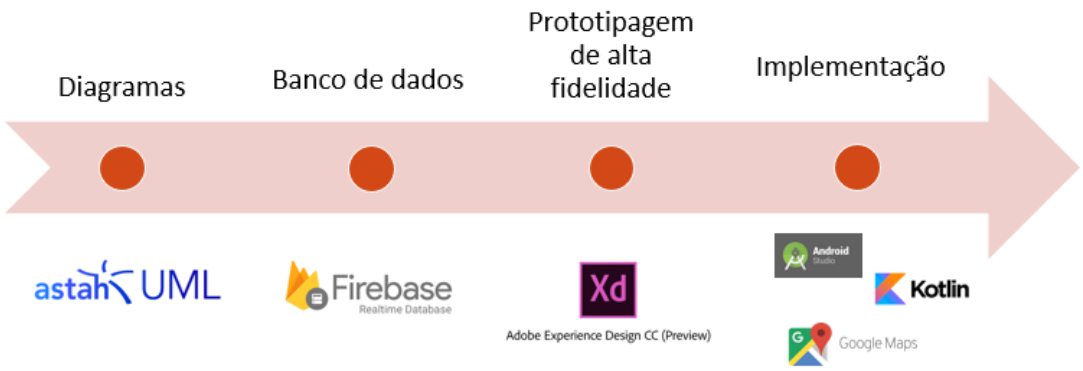
O cliente deve selecionar qual serviço deseja solicitar: entrega ou viagem. Logo após o cliente insere os endereços de origem e destino, e caso esteja solicitando uma entrega, o cliente pode adicionar até 4 paradas. Em seguida o cliente confirma a solicitação e os pilotos mais próximos são acionados.

O aplicativo do motociclista funciona da seguinte maneira: o motociclista escolhe em qual categoria vai atuar no momento do cadastro, mesmo que atue nas duas, só é possível estar ativo em uma modalidade por vez. Após selecionar a modalidade, o motociclista deve ficar *online* e aguardar novas corridas. Ao chegar uma nova corrida, o motociclista pode aceitar ou cancelar. Após aceitar a corrida, basta seguir as instruções do aplicativo. É possível, quando em viagem, navegar utilizando o *Google Maps* ou o *Waze*.

3 Metodologia

Neste capítulo serão apresentados os métodos e procedimentos adotados para o desenvolvimento do trabalho. A Figura 4 apresenta a cronologia dos processos para implementação da aplicação. Primeiramente foram desenhados os diagramas, para melhor visualização do sistema. Em seguida, o processo de modelagem do banco de dados e a prototipação de alta fidelidade. Por fim, o desenvolvimento do aplicativo.

Figura 4 – Desenvolvimento



Fonte: elaborado pela autora

3.1 Visão geral do sistema

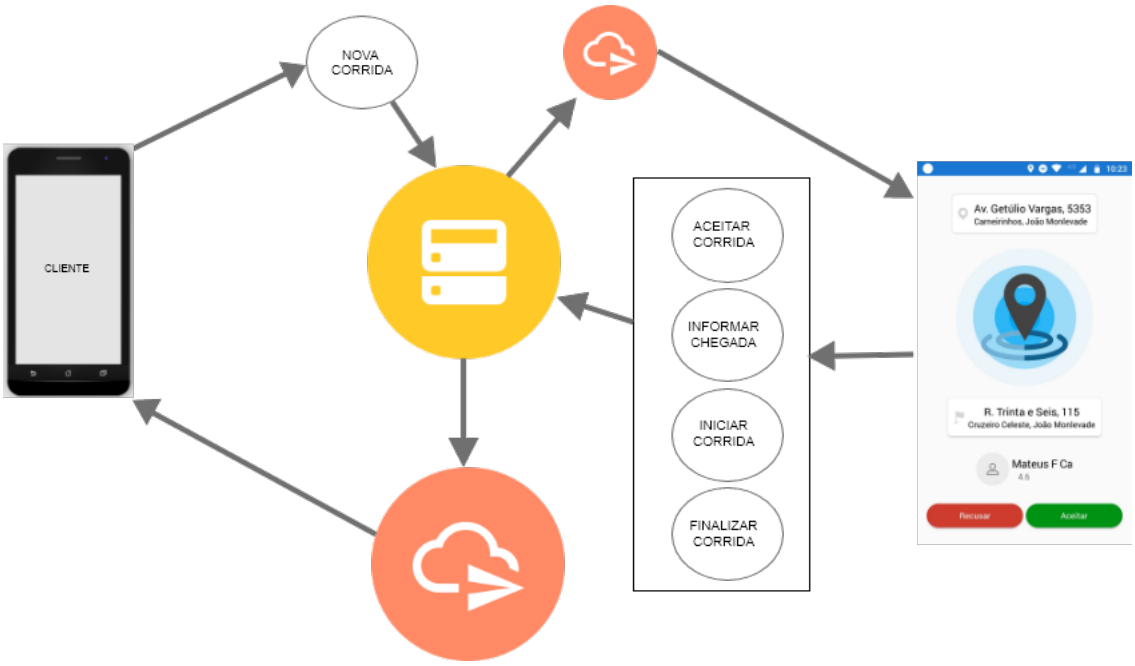
O sistema como um todo é constituído de três partes: o servidor, a aplicação do passageiro e a aplicação do mototaxista. O servidor é responsável pela manipulação das regras de negócio da aplicação, por exemplo: validação de cadastro, o gerenciamento de motoristas *online* e aptos a receberem uma nova corrida, o valor final da corrida. A aplicação do passageiro, permite que um usuário se cadastre e solicite uma nova corrida. O passageiro deve informar os endereços de origem e destino, confirmar a solicitação e aguardar até que um motorista aceite sua corrida.

A aplicação do mototaxista, que será implementada no presente trabalho, possui os módulos de Cadastro e Autenticação, Dados financeiros, Nova corrida e Histórico de corridas. O módulo de cadastro e autenticação é responsável pela coleta dos dados do mototaxista que deseja fazer parte da plataforma. Após preencher o formulário de cadastro, seus dados são salvos no *Realtime Database* e o mototaxista passa a ter acesso à plataforma. O módulo de Dados Financeiros é responsável por coletar os dados bancários do mototaxistas e salvar no banco de dados, para que o mesmo possa solicitar o resgate do

valor arrecadado com as corridas. O módulo de Histórico de corridas, gerencia as corridas já realizadas pelos mototaxistas, recuperando do banco de dados os detalhes de cada corrida, como: data, trajeto, avaliação, valor e passageiro.

O módulo Nova Corrida, apresentado na Figura 5, representa todo o fluxo de corrida realizado pelo mototaxista. O fluxo é iniciado quando um passageiro solicita uma nova corrida. Um *push* dessa solicitação chega para o motorista. O motorista pode aceitar ou cancelar a corrida. Ao aceitar a corrida, o mototaxista deve se deslocar ao encontro do passageiro e levá-lo ao seu local de destino. Ao final da viagem, tanto o mototaxista quanto o passageiro, podem avaliar a corrida realizada. Após finalizar uma corrida, o mototaxista se torna apto a receber novas solicitações.

Figura 5 – Visão geral do fluxo de uma nova corrida



Fonte: elaborado pela autora

3.2 Ferramentas e tecnologias utilizadas

3.2.1 Kotlin

O Kotlin é um projeto de código aberto e gratuito da licença do Apache 2.0, desenvolvido pela *Jetbrains*. É uma linguagem JVM, baseada na programação funcional e estaticamente tipada, com uma sintaxe mais expressiva e concisa do que Java. A linguagem utiliza de conceitos funcionais, como expressões lambda e por ser multi-paradigma, o desenvolvedor pode utilizar tanto o paradigma funcional como o orientado a objetos (KOTLIN ORG, 2019).

A linguagem Kotlin foi oficialmente anunciado no *Google I/O* de 2019, sendo fortemente recomendada para o desenvolvimento de aplicativos Android (TRIPATHI, 2019). O Kotlin possui algumas características que a diferem de Java, como mostra a Tabela 1.

Compatibilidade	o Kotlin é totalmente compatível com o JDK 6, garantindo que os aplicativos Kotlin possam ser executados em dispositivos Android mais antigos sem problemas.
Desempenho	Com o suporte do Kotlin para funções embutidas, o código usando lambdas geralmente é executado ainda mais rápido que o mesmo código escrito em Java.
Interoperabilidade	O Kotlin é 100 % interoperável com Java, permitindo usar todas as bibliotecas Android existentes em um aplicativo Kotlin.
Segurança	Valida valores nulos em tempo de compilação, para evitar exceções em tempo de execução.
Curva de aprendizado	Para um desenvolvedor Java, é muito fácil começar com o Kotlin utilizando o conversor automatizado de Java para Kotlin. O Kotlin Koans oferece um guia sobre os principais recursos do idioma com uma série de exercícios interativos.

Tabela 1 – Vantagens da linguagem Kotlin (KOTLIN, 2019)

Grandes empresas adotaram o Kotlin e algumas delas fizeram a conversão 100% do código, diminuindo o número de linhas de código e facilitando a vida do programadores, como por exemplo: *Pinterest*, *Basecamp*, *App Lock da Keepsafe*, entre outras. Por fim, Kotlin é uma linguagem moderna e modelada estaticamente, o que permite o aumento da produtividade do desenvolvedor e diminui a escrita de códigos clichês.

3.2.2 *Parse SDK*

O *Parse Server* é uma versão de código aberto do *back-end* do *Parse* que pode ser implantada em qualquer infraestrutura que possa executar o Node.js. Um *back-end* para governar todas as APIs e serviços de nuvem para aplicativos iOs, Android e *Windows*, de forma rápida e com esforço mínimo (PARSE, 2019).

Com a *API Parse*, um aplicativo consegue armazenar objetos de dados e arquivos na nuvem do *Parse*, enviar e receber notificações *push*, gerenciar usuários, lidar com dados de localização geográfica e usar plataformas de mídia social como *Twitter* e *Facebook*. No presente trabalho, a *API Parse* foi usada para controlar a sessão de um usuário logado e para salvar as imagens enviadas no cadastro. O *ParseFile* permite enviar um Bitmap e obter uma url correspondente de retorno, que será salva juntamente com os dados do usuário no *Firebase Realtime Database*.

3.2.3 *Google Maps*

Neste trabalho será utilizado o SDK do *Google Maps*, que permite adicionar mapas, marcadores, sobreposições de solo e de bloco, polilinhas, polígono e alterar a exibição do usuário em uma determinada área do mapa, possibilitando a modelagem do mapa de acordo com a necessidade da aplicação. A exibição do mapa, *download* de dados, a resposta aos gestos do usuário e o acesso ao servidores do *Google Maps*, são gerenciados automaticamente pela API (GOOGLE DEVELOPERS, 2019).

Para utilizar o *Google Maps* na aplicação, é necessário passar por algumas etapas no *Google Clound Plataform Console*, são elas:

1. Criar uma conta de cobrança: usada para rastrear os custos associados aos seus projetos;
2. Criar um projeto: base para gerenciar serviços, credenciais, cobrança, APIs e SDKs;
3. Habilitar a *API Maps SDK* para Android: fornece a chave de acesso aos serviços da API, que deve ser adicionada à aplicação para a configuração do mapa.

A chave de acesso é um identificador exclusivo que deve ser adicionada à aplicação para a configuração do mapa. A chave da API é usada para autenticar as solicitações associadas a aplicação para fins de uso e cobrança.

3.2.4 *Firebase*

O Firebase é uma plataforma móvel da Google para desenvolvimento de aplicativos móveis e Web. O Firesbase é um *Bass (Backend as a Service)*, ou seja, serviços como: autenticação, notificações, banco de dados, armazenamento e outros serviços, estão prontos para ser integrado com a aplicação desejada. Além da gama de serviços oferecidos, o Firebase disponibiliza alguns produtos gratuitamente e de acordo com o uso e a necessidade da aplicação, é possível aderir a outros planos. A Figura 6 apresenta os planos dos serviços integrados ao presente trabalho.

3.2.4.1 *Firebase Realtime Database*

A ferramenta *Firebase Realtime Database*, foi utilizada para a persistência de dados da aplicação, como: dados dos mototaxistas (dados pessoais, nota de avaliação, localização) e das corridas em andamento (origem, destino, valor total, mototaxista, cliente, *chat*). O *Realtime Database* é um banco de dados NoSQL hospedado na nuvem e usa a sincronização de dados em tempo real para todos os dispositivos que compartilham uma instância do banco, sempre que algum dado é alterado no servidor do *Firebase*, como mostra a Figura 7. Outra característica dessa ferramenta, é que os dados são mantidos localmente e no caso do aplicativo ficar offline, os dados locais são recuperados e quando a conexão for

Figura 6 – Planos de preço Firebase

Produtos	Plano Spark Limites generosos para amadores **Gratuito**	Plano Flame Preço fixo para apps em expansão **US$ 25/mês**	Plano Blaze Calcule o preço de apps em escala **Pagamento por utilização** ✓ Uso gratuito do plano Spark incluso*
Autenticação			
Phone Auth - EUA, Canadá e Índia	10 mil/mês	10 mil/mês	US$ 0,01/verificação
Phone Auth - todos os outros países	10 mil/mês	10 mil/mês	US$ 0,06/verificação
Outros serviços de autenticação	✓	✓	✓
Cloud Messaging (FCM)	Gratuito		
Realtime Database			
Conexões simultâneas	100	200k	200k/database
GB armazenados	1 GB	2,5 GB	US$ 5/GB
GB de download	10 GB/mês	20 GB/mês	US$ 1/GB
Vários bancos de dados por projeto	✗	✗	✓

Fonte: *Firebase (2019)*

restabelecida, o *Realtime Database* instantâneamente atualiza o dispositivo cliente e mescla os conflitos automaticamente, tornando a experiência mais rápida e fluída.

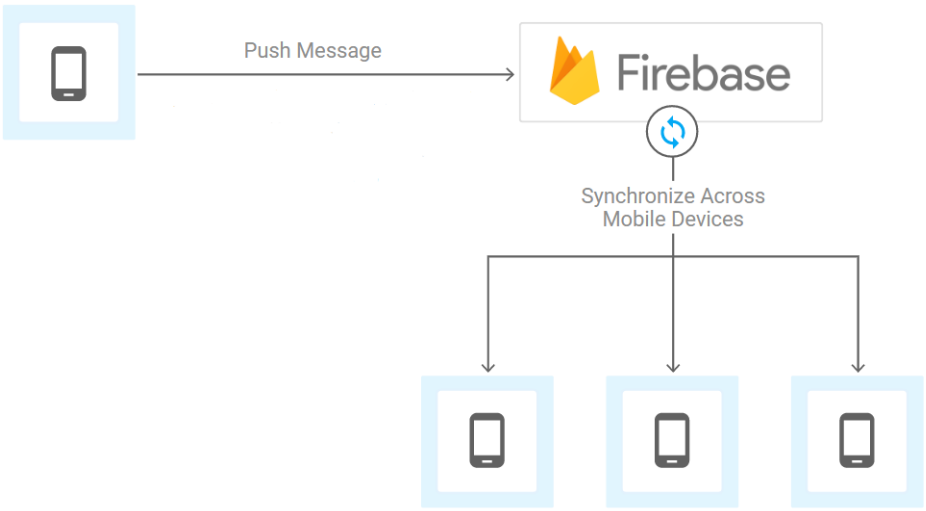
Quando em corrida, o mototaxista e o passageiro podem se comunicar através de mensagens dentro do aplicativo. Como mencionado anteriormente, toda modificação no banco é refletida em questão de milisegundos em todos os dispositivos conectados, isso torna possível a implementação do bate-papo prática e eficiente.

3.2.4.2 *Firebase Cloud Messaging*

Segundo Shi (2018), o *Firebase Cloud Messaging (FCM)* é uma solução de mensagens entre plataformas que permite o envio confiável de notificações sem custo. É possível interagir com as aplicações conectadas através de notificações que podem ser exibidas para o usuário ou mensagens de dados que podem determinar o que acontece com a aplicação. Outra característica, é a possibilidade de distribuir mensagens para um grupo de dispositivos, como apresentado na Figura 8.

No presente trabalho, o *Firebase Cloud Messaging* será usado para enviar notifi-

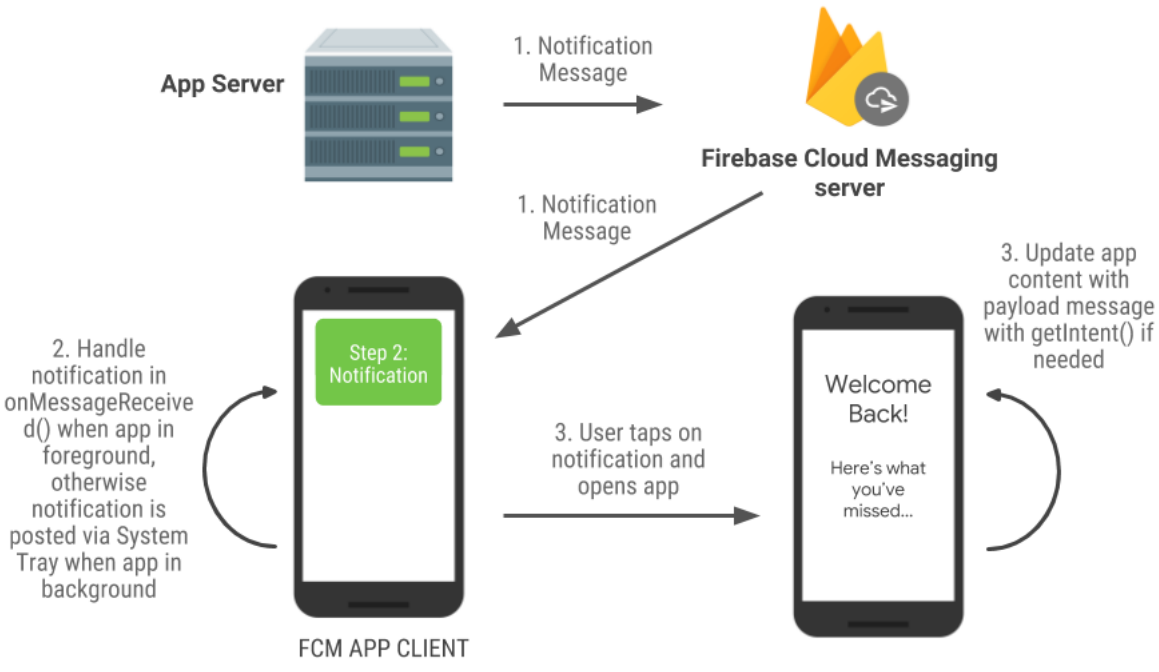
Figura 7 – *Firebase Realtime Database*



Fonte: Google Cloud (GOOGLE CLOUD, 2019)

cações aos mototaxistas informando sobre: *status* do cadastro, uma nova corrida e nova mensagem no *chat*. A notificação de uma nova corrida utiliza do serviço de segmentação versátil de mensagem para enviar a mensagem de forma simultânea para todos os mototaxistas que estiverem aptos a receber uma nova corrida.

Figura 8 – *Firebase Cloud Messaging*



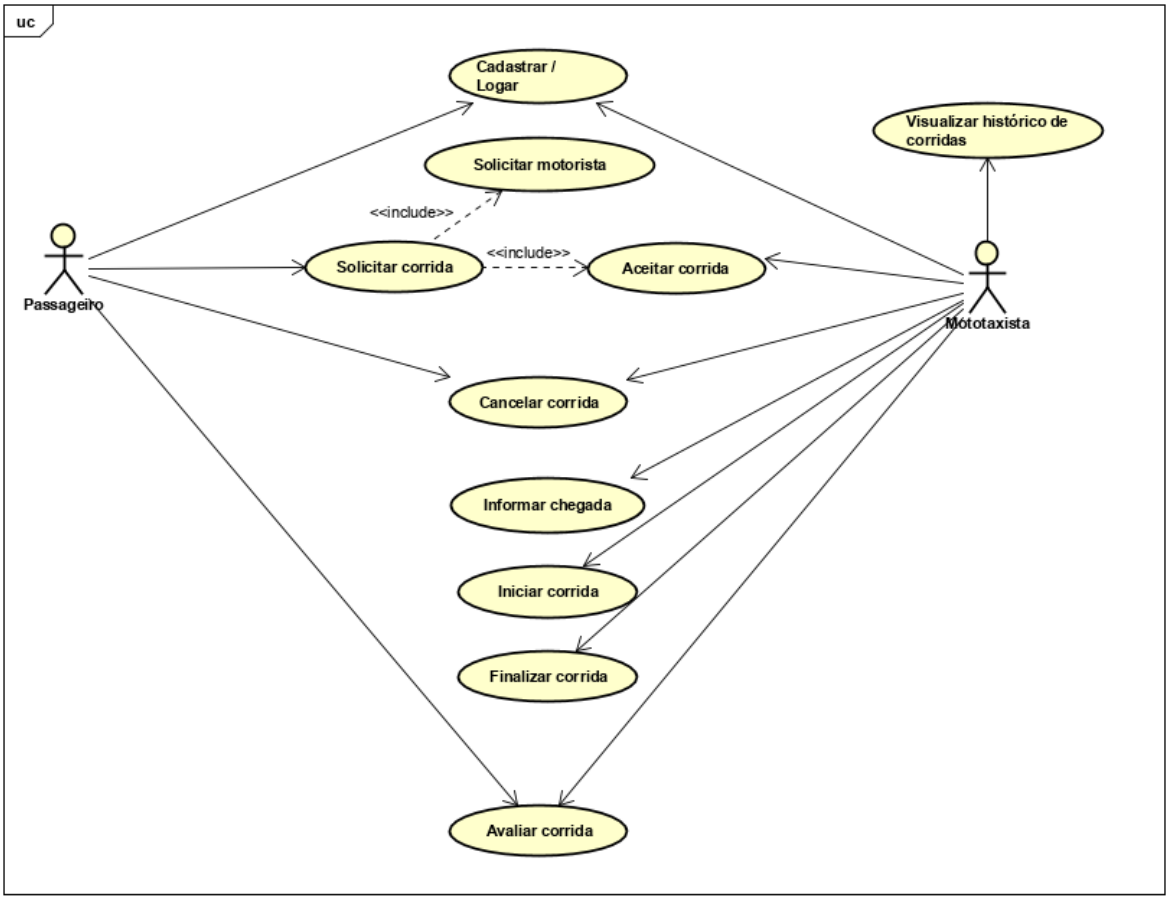
Fonte: Firebase (SHI, 2018)

3.3 Diagramas de caso de uso

Após o levantamento dos requisitos, foi criado o diagrama de caso de uso com o objetivo de modelar o comportamento do sistema, identificando as interações entre a aplicação e seus atores. O diagrama apresenta a visão externa geral das funções e do sistema, definindo o que o sistema faz e não como faz.

A Figura 9 a seguir representa o diagrama de caso de uso geral do sistema, criado utilizando a ferramenta Astah.

Figura 9 – Diagrama de caso de uso



Fonte: elaborado pela autora

3.4 Diagramas de atividades

Os diagramas de atividades apresentados nas figuras 10 a 15, juntamente com as estórias textuais, foram elaborados durante a fase de requisitos com o objetivo de ilustrar os casos de uso da Figura 9, apresentada na seção anterior. Os diagramas de atividades a seguir, foram desenhados utilizando a ferramenta Astah.

3.4.1 Caso de uso - Cadastrar

A Tabela 2 detalha o fluxo que o usuário deve seguir para se cadastrar na aplicação e se tornar um mototaxista vinculado ao sistema proposto.

Tabela 2 – Caso de uso - Cadastrar

Caso de uso: Cadastrar
Objetivo: Permitir que um mototaxista realize seu cadastro no aplicativo e se torne apto a receber corridas. Ator: Mototaxista Pré-condições: • Aplicativo instalado no dispositivo.
Cenário de sucesso principal: 1. O usuário abre o aplicativo. 2. O usuário clica em Cadastrar. 3. O usuário preenche o formulário com suas informações pessoais. 4. O usuário clica em Continuar e é direcionado para a próxima etapa. 5. O usuário tira foto dos documentos solicitados. 6. O usuário clica em Finalizar Cadastro. 7. O usuário é direcionado para dentro do aplicativo. Extensões: 4a, 6a. Falha de conexão com a internet 1. Uma mensagem de falha na conexão é exibida ao usuário juntamente com uma opção clicável de Tentar Novamente. 2. O usuário clica em Tentar Novamente e a aplicação tenta enviar os dados novamente.

Fonte: elaborado pela autora

3.4.2 Caso de uso - Aceitar corrida

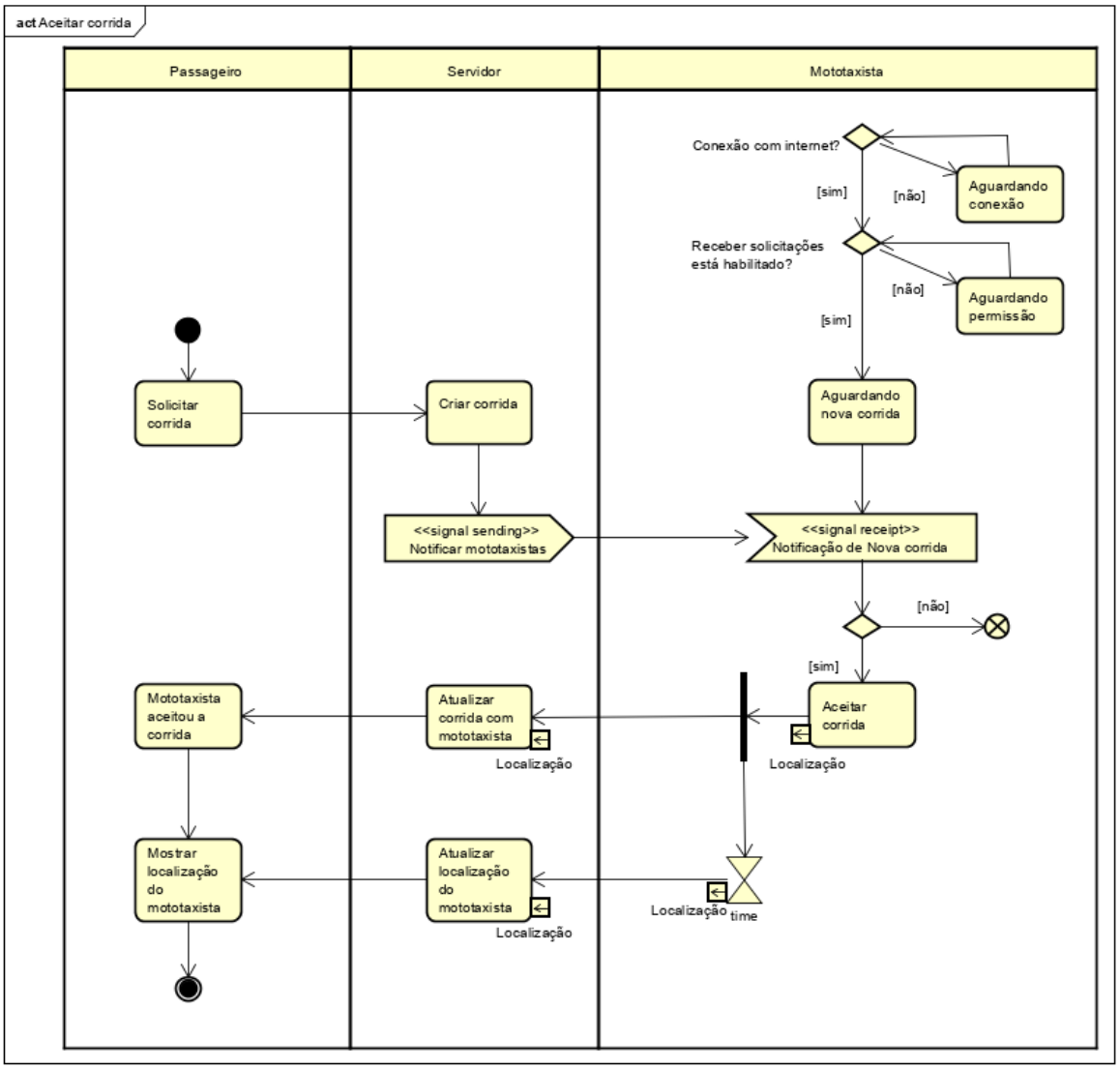
Na Tabela 3 está descrito o cenário ideal do caso de uso aceitar corrida. Em seguida o diagrama de atividades respectivo, apresentado na Figura 10.

Tabela 3 – Caso de uso - Aceitar corrida

Caso de uso: Aceitar corrida
Objetivo: Permitir que o mototaxista receba uma notificação de solicitação de uma nova corrida e possa aceitá-la. Ator: Mototaxista Pré-condições: • Aplicativo instalado no dispositivo. • Mototaxista deve estar logado no aplicativo. • Aplicativo deve estar conectado à internet. • A opção de Receber solicitações deve estar habilitada.
Cenário de sucesso principal: 1. O usuário recebe uma notificação de uma nova corrida. 2. O usuário clica em Aceitar corrida. 3. O usuário envia sua localização de tempos em tempos para o servidor. 4. Sistema notifica passageiro. Extensões: 2a. Erro ao enviar a solicitação para o servidor. 1. Uma mensagem de erro é exibida. 2. O estado da tela não muda.

Fonte: elaborado pela autora

Figura 10 – Diagrama de atividades - Aceitar corrida



Fonte: elaborado pela autora

3.4.3 Caso de uso - Cancelar corrida

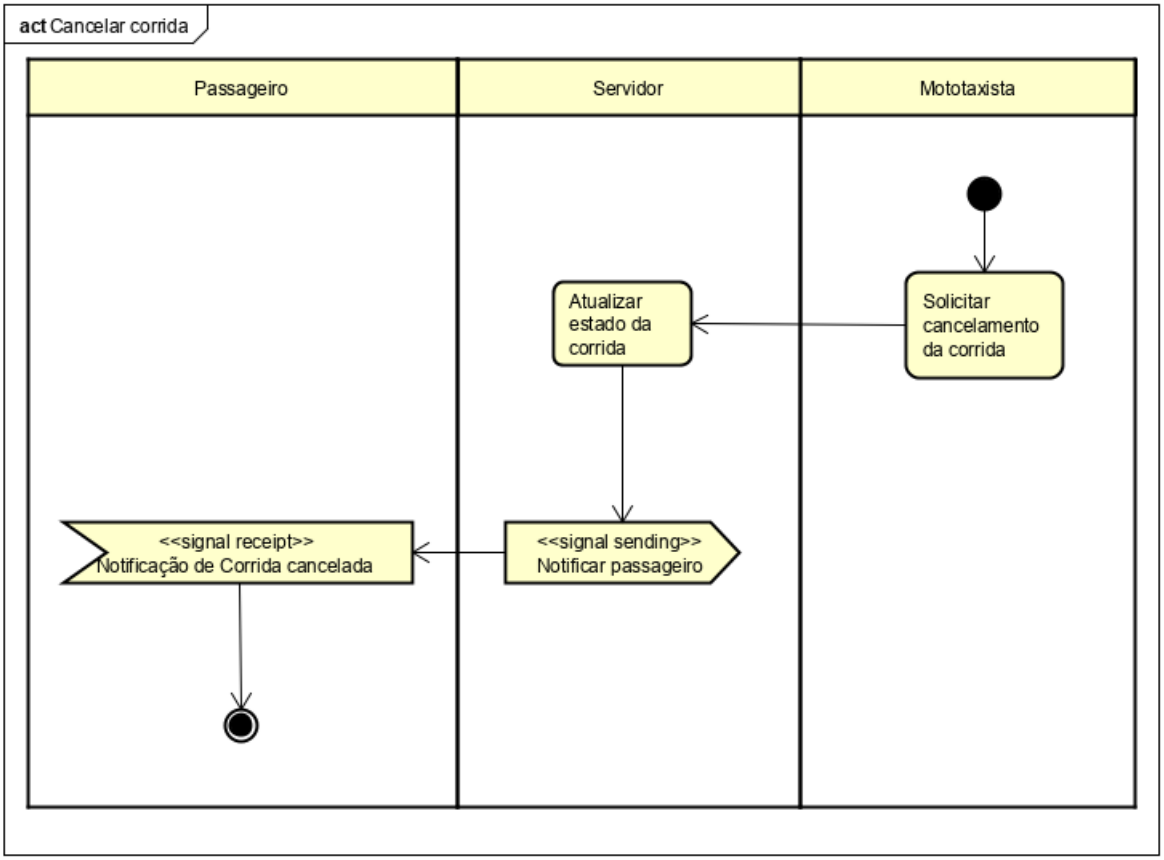
A Tabela 4 detalha o caso de uso para cancelar uma corrida já aceita. Em seguida, na Figura 11, o diagrama de atividades para esse caso.

Tabela 4 – Caso de uso - Cancelar corrida

Caso de uso: Cancelar corrida
Objetivo: Permitir que o mototaxista possa cancelar uma corrida. Ator: Mototaxista Pré-condições: • Aplicativo instalado no dispositivo. • Mototaxista deve estar logado no aplicativo. • Aplicativo deve estar conectado à internet. • Mototaxista deve estar em uma corrida.
Cenário de sucesso principal: 1. O usuário clica em Cancelar. 2. Uma mensagem de confirmação é exibida ao usuário. 3. O usuário clica em Sim e o mapa inicial é exibido. 4. Sistema notifica passageiro. Extensões: 3a. Erro ao enviar a solicitação para o servidor. 1. Uma mensagem de erro é exibida. 2. O estado da tela não muda.

Fonte: elaborado pela autora

Figura 11 – Diagrama de atividades - Cancelar corrida



Fonte: elaborado pela autora

3.4.4 Caso de uso - Informar chegada

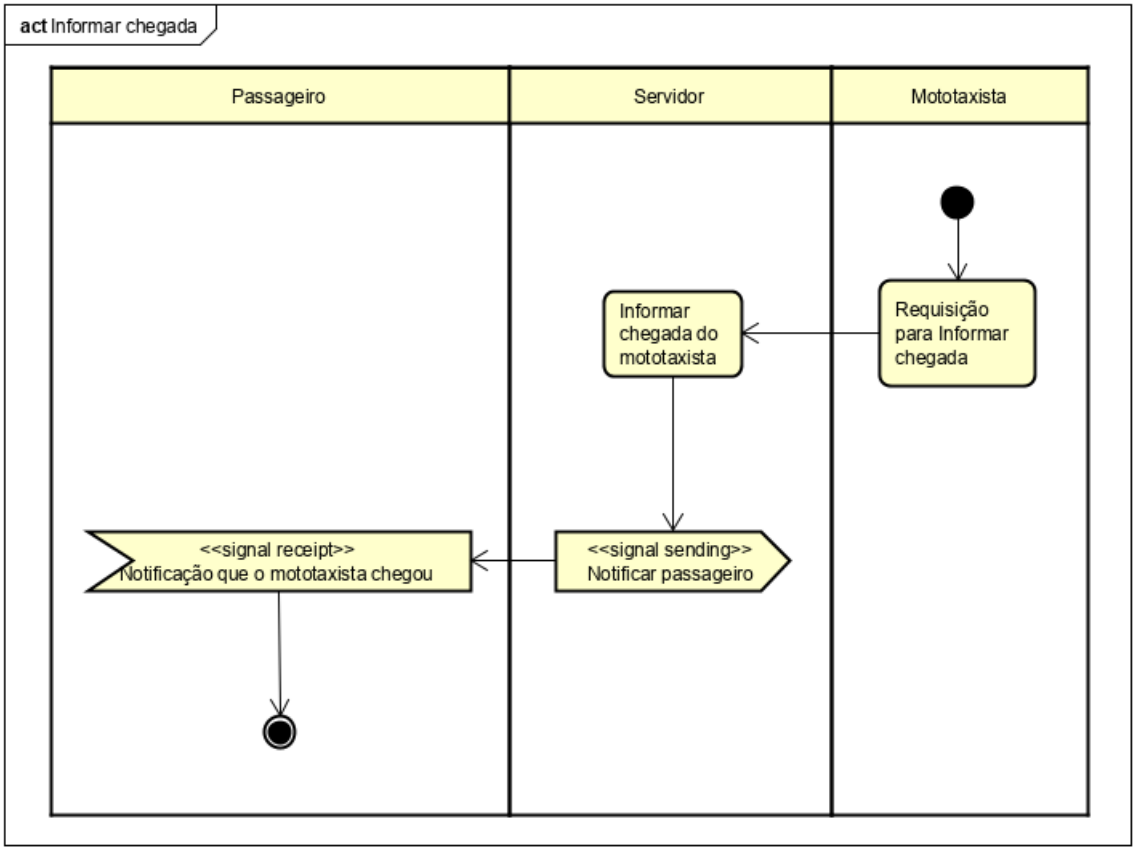
A Tabela 5 e a Figura 12, contextualizam o caso de uso na qual o mototaxista informa a sua chegada ao local de encontro do passageiro.

Tabela 5 – Caso de uso - Informar chegada

Caso de uso: Informar chegada
Objetivo: Permitir que o mototaxista informe a chegada e o cliente receba uma notificação. Ator: Mototaxista Pré-condições: • Aplicativo instalado no dispositivo. • Mototaxista deve estar logado no aplicativo. • Aplicativo deve estar conectado à internet. • Mototaxista deve estar em uma corrida.
Cenário de sucesso principal: 1. O usuário clica em Informar chegada. 2. Uma mensagem de confirmação é exibida. 3. O usuário clica em Sim. 4. O trajeto até o destino final é exibido no mapa. 5. Sistema notifica passageiro. Extensões: 3a. Erro ao enviar a solicitação para o servidor. 1. Uma mensagem de erro é exibida. 2. O estado da tela não muda.

Fonte: elaborado pela autora

Figura 12 – Diagrama de atividades - Informar chegada



Fonte: elaborado pela autora

3.4.5 Caso de uso - Iniciar corrida

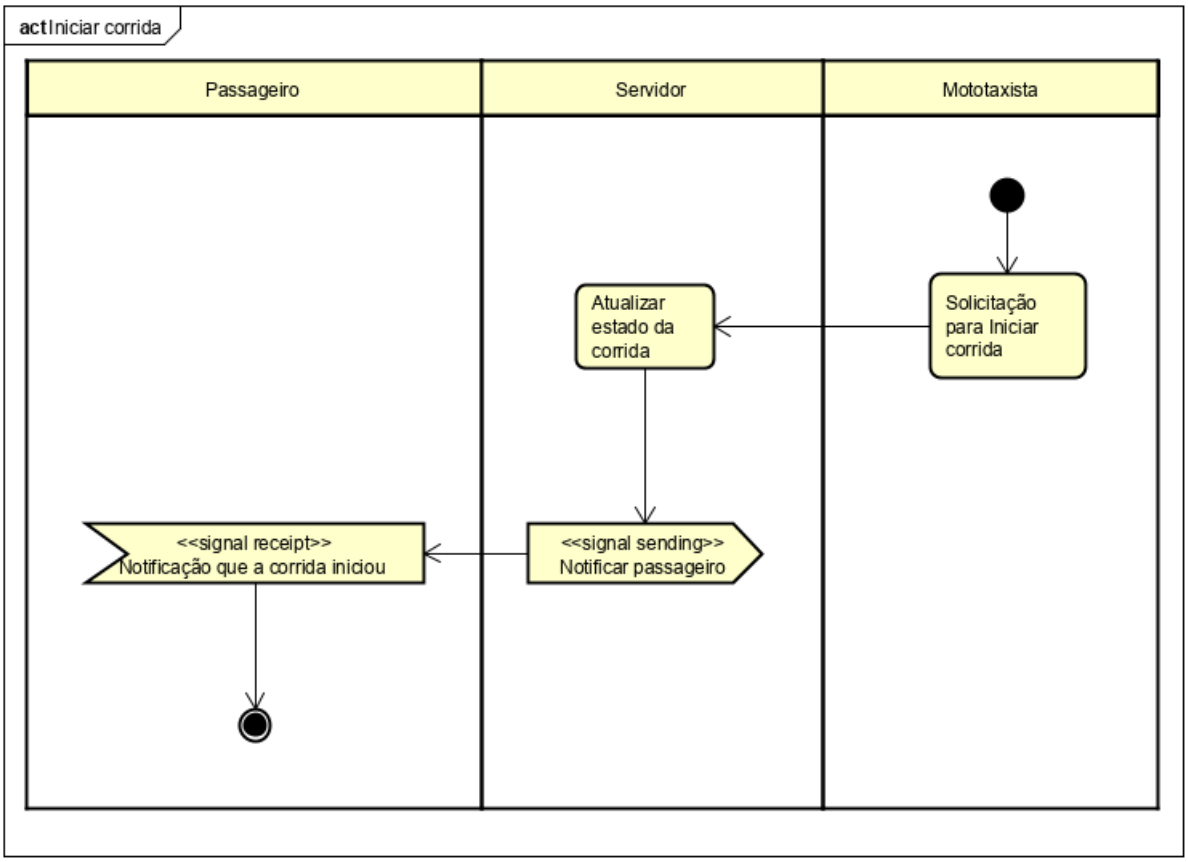
Na Tabela 6, seguido da Figura 13, está descrito o caso de uso que permite um mototaxista iniciar um corrida.

Tabela 6 – Caso de uso - Iniciar corrida

Caso de uso: Iniciar corrida
Objetivo: Permitir que o mototaxista inicie uma corrida. Ator: Mototaxista Pré-condições: • Aplicativo instalado no dispositivo. • Mototaxista deve estar logado no aplicativo. • Aplicativo deve estar conectado à internet. • Mototaxista deve ter informado a chegada.
Cenário de sucesso principal: 1. O usuário clica em Iniciar corrida. 2. O estado da tela muda. 3. Sistema notifica passageiro. Extensões: 1a. Erro ao enviar a solicitação para o servidor. 1. Uma mensagem de erro é exibida. 2. O estado da tela não muda.

Fonte: elaborado pela autora

Figura 13 – Diagrama de atividades - Iniciar corrida



Fonte: elaborado pela autora

3.4.6 Caso de uso - Finalizar corrida

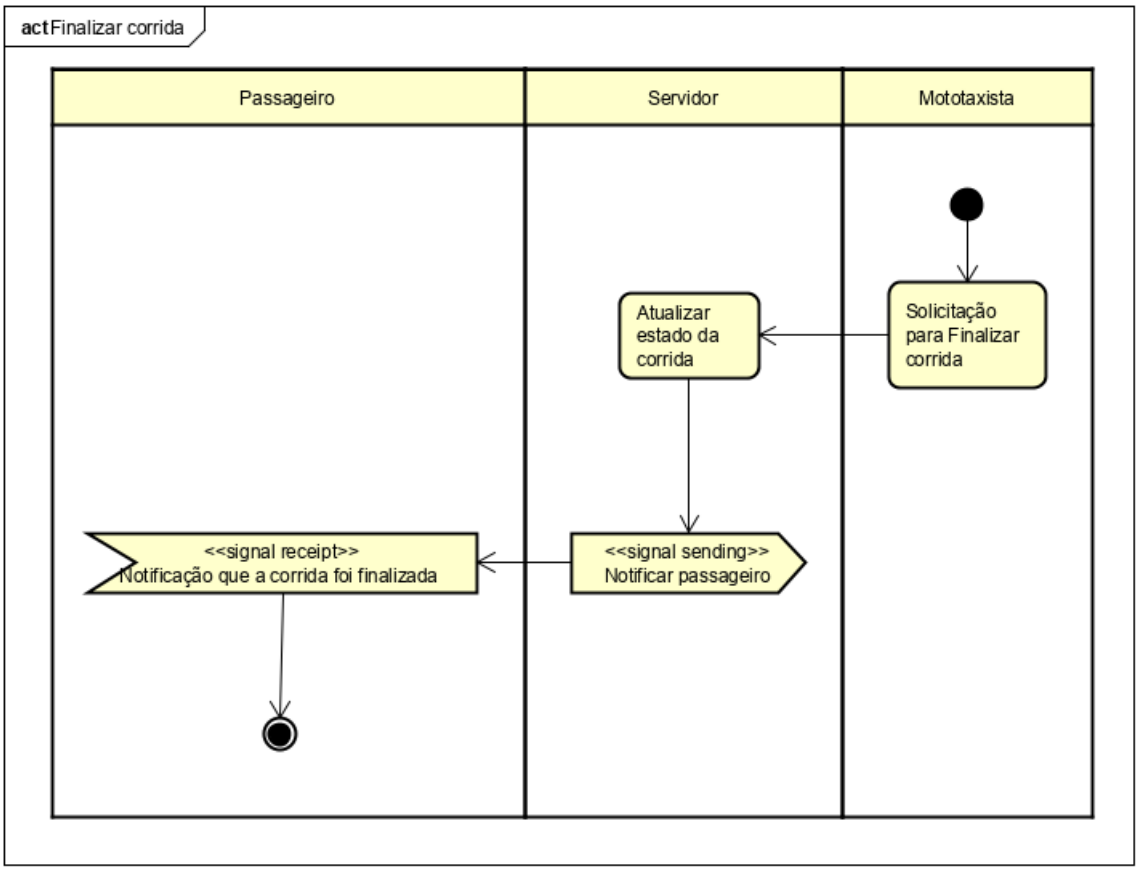
A seguir, a Tabela 7 e a Figura 14, descrevem o caso de uso para finalizar uma corrida.

Tabela 7 – Caso de uso - Finalizar corrida

Caso de uso: Finalizar corrida
Objetivo: Permitir que o mototaxista finalize uma corrida completada. Ator: Mototaxista Pré-condições: • Aplicativo instalado no dispositivo. • Mototaxista deve estar logado no aplicativo. • Aplicativo deve estar conectado à internet. • Mototaxista deve ter iniciado uma corrida.
Cenário de sucesso principal: 1. O usuário clica em Finalizar corrida. 2. A tela de avaliar corrida é exibida. 3. Sistema notifica passageiro. Extensões: 1a. Erro ao enviar a solicitação para o servidor. 1. Uma mensagem de erro é exibida. 2. O estado da tela não muda.

Fonte: elaborado pela autora

Figura 14 – Diagrama de atividades - Finalizar corrida



Fonte: elaborado pela autora

3.4.7 Caso de uso - Avaliar corrida

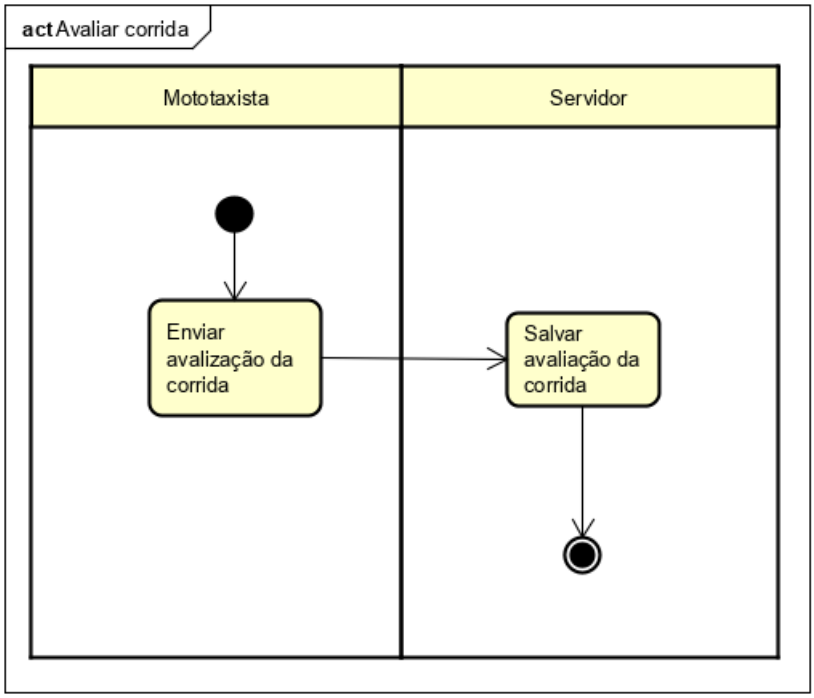
Após finalizar uma corrida, o mototaxista pode avaliá-la. A Tabela 8 a seguir, apresenta o caso de uso referente a avaliar uma corrida. A Figura 15 apresenta o diagrama do caso de uso avaliar corrida.

Tabela 8 – Caso de uso - Avaliar corrida

Caso de uso: Avaliar corrida
Objetivo: Permitir que o mototaxista avalie o passageiro que foi transportado durante a corrida. Ator: Mototaxista Pré-condições: • Aplicativo instalado no dispositivo. • Mototaxista deve estar logado no aplicativo. • Aplicativo deve estar conectado à internet. • Mototaxista deve ter finalizado a corrida.
Cenário de sucesso principal: 1. O usuário avalia a corrida. 2. O usuário clica em Avaliar. 3. A aplicação volta para a o mapa inicial. 4. O usuário está apto apto a corrida. Extensões: 2a. Erro ao enviar a solicitação para o servidor. 1. Uma mensagem de erro é exibida.

Fonte: elaborado pela autora

Figura 15 – Diagrama de atividades - Avaliar corrida



Fonte: elaborado pela autora

3.4.8 Caso de uso - Visualizar histórico de corridas

A Tabela 9, detalha a estória do caso de uso para visualizar o histórico de corridas feitas pelo mototaxista.

Tabela 9 – Caso de uso - Visualizar histórico de corridas

Caso de uso: Visualizar histórico de corridas
Objetivo: Permitir que o mototaxista tenha acesso a listagem de todas as corridas realizadas por ele. Ator: Mototaxista Pré-condições: • Aplicativo instalado no dispositivo. • Mototaxista deve estar logado no aplicativo. • Aplicativo deve estar conectado à internet.
Cenário de sucesso principal: 1. O usuário abre o menu lateral. 2. O usuário clica em Minhas corridas. 3. A lista de corridas feitas pelo usuário é exibida. Extensões: 3a. Falha de conexão com a internet 1. Uma mensagem de falha na conexão é exibida ao usuário juntamente com uma opção clicável de Tentar Novamente. 2. O usuário clica em Tentar Novamente e a aplicação faz a requisição dos dados novamente.

Fonte: elaborado pela autora

3.5 Banco de dados

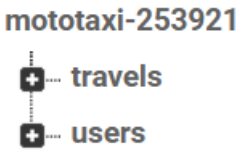
Como descrito na Seção 3.3.4, o banco de dados utilizado será o Firebase Realtime Database. Se trata de um banco de dados NoSQL, ou seja: não relacional, possuindo otimizações e funcionalidades diferentes de um banco de dados relacional. Os dados são armazenados no formato de uma árvore JSON, portanto ao adicionar um novo dado, ele se torna um nó da árvore que pode ser identificado a partir de uma chave única associada. O banco de dados da aplicação, pode ser consultado e alterado diretamente à partir do console do Firebase.

A Figura 16 apresenta a estrutura geral do banco de dados da aplicação, que possui apenas dois nós principais. A Figura 17 detalha os nós principais, sendo um nó correspondente as corridas em andamento/realizadas e outro nó que representa os usuários do aplicativo, mototaxistas e passageiros.

A Figura 17a representa o nó responsável por armazenar as informações de cadastro do usuário e os atributos necessários para realizar uma corrida, por exemplo os atributos: *isDriver*, que indica se o usuário é um mototaxista ou um passageiro, e *isAvailable*, que indica se o usuário está disponível para realizar uma corrida.

Já a Figura 17b representa o nó responsável por armazenar as informações da corrida em andamento/realizada como endereço de origem e destino, os usuários envolvidos, mototaxista e passageiro, o *status* e o valor total da corrida.

Figura 16 – Banco de dados - Nós principais



Fonte: elaborado pela autora

Figura 17 – Banco de dados

(a) Nó *users*

(b) Nó *travels*

Fonte: elaborado pela autora

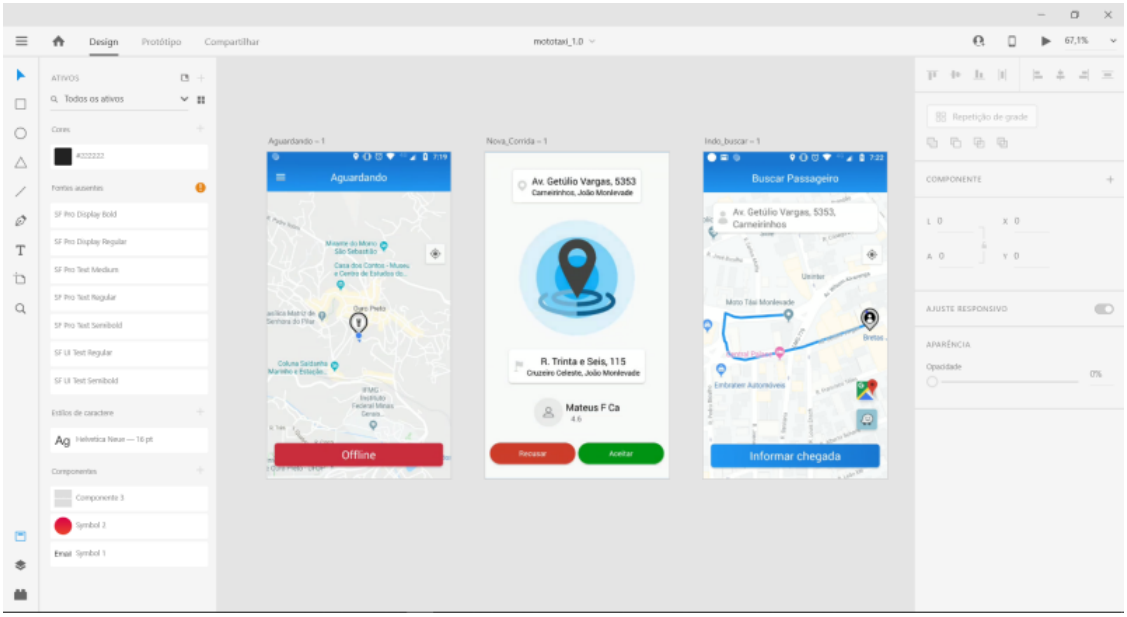
4 Resultados

Neste capítulo será apresentado o aplicativo do mototaxista desenvolvido à partir da metodologia descrita no capítulo anterior, detalhando as funcionalidade e a execução da aplicação.

4.1 Prototipagem de alta fidelidade

Todas as telas implementadas na aplicação passaram antes pela fase de prototipagem, com o objetivo de criar e validar os elementos da interface e o fluxo de navegação dentro do aplicativo. Os elementos visuais necessários para a concepção das telas foram levantados a partir dos casos de uso e logo após foram tomadas as decisões de *design* da arte, com por exemplo cores, disposição e tamanhos dos elementos na tela, estilos, *feedbacks* e animações. A Figura 18 apresenta algumas telas desenvolvidas na ferramenta Adobe XD.

Figura 18 – Protótipos

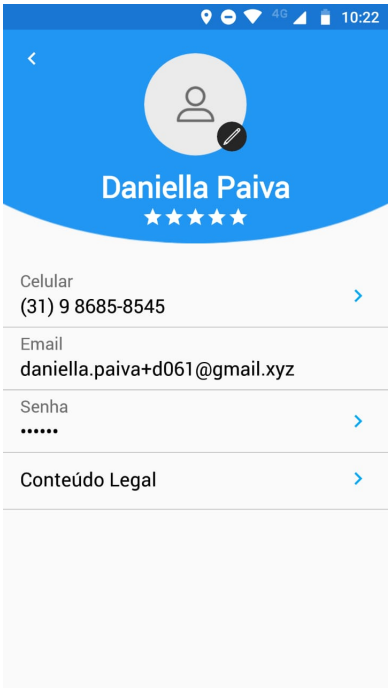


Fonte: elaborado pela autora

4.2 Perfil

Através do módulo Perfil, apresentado na Figura 19, o mototaxista pode visualizar e alterar suas informações de cadastro, como: foto do perfil, nome, celular e sua senha do aplicativo.

Figura 19 – Perfil



Fonte: elaborado pela autora

4.3 Histórico

A Figura 22 apresenta o recurso Histórico, disponibilizado para que o mototaxista possa visualizar as viagens realizadas através do aplicativo juntamente com as informações sobre o trajeto percorrido, data e hora da viagem, e o valor recebido. Ao clicar em uma viagem realizada apresentada na lista, uma nova tela com os detalhes da viagem, é apresentada ao usuário.

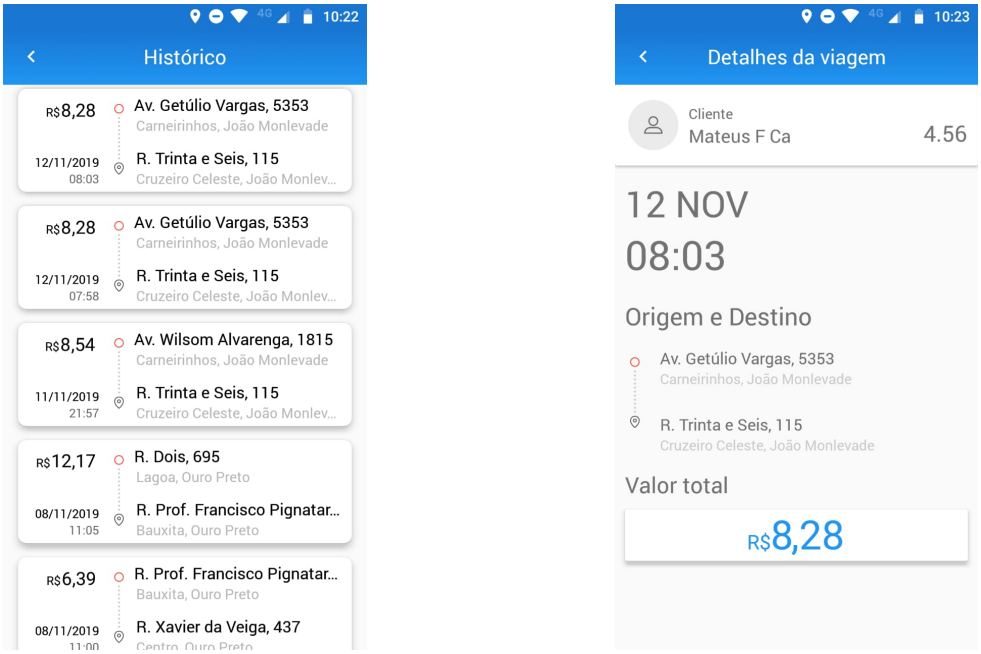
4.4 Relatórios

A funcionalidade Relatórios, apresentada na Figura 21, permite ao mototaxista acessar o relatório das corridas efetuadas e os valores arrecadados por dia e por semana. Para facilitar a compreensão do relatório, as informações são apresentadas na forma de um gráfico de barras onde as viagens são agrupadas semanalmente.

4.5 Ajuda

Este módulo tem como objetivo esclarecer possíveis dúvidas do usuário no dia a dia, conforme ilustra a Figura 22.

Figura 20 – Histórico



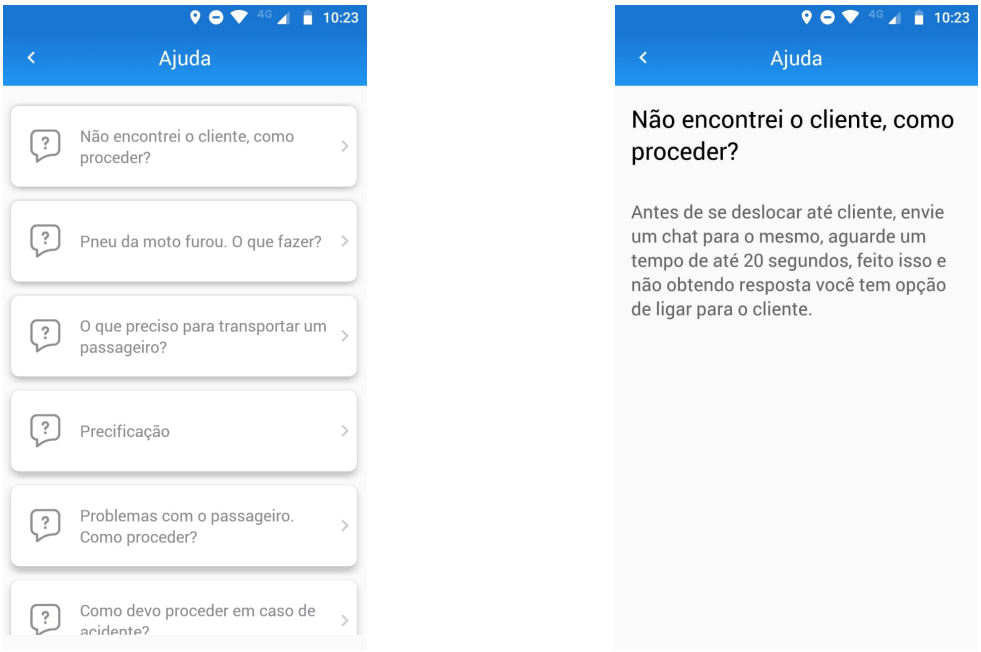
(a) Lista de viagens (b) Detalhe da viagem

Fonte: elaborado pela autora

Figura 21 – Relatórios

Fonte: elaborado pela autora

Figura 22 – Ajuda



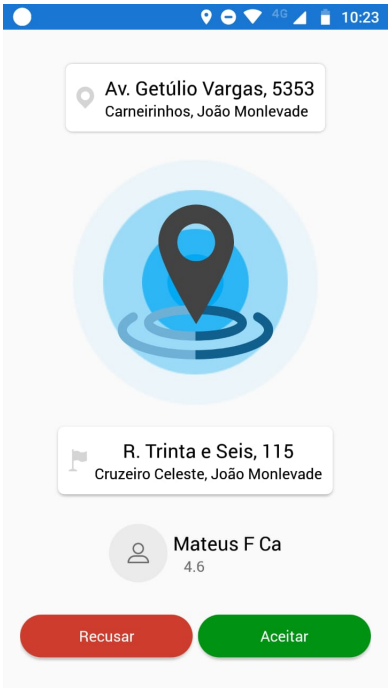
(a) Lista de ajuda (b) Detalhe da ajuda

Fonte: elaborado pela autora

4.6 Nova corrida

Quando uma nova corrida chega para o mototaxista, uma tela com as informações de endereço de origem, endereço de destino, nome e avaliação do passageiro, são exibidas. O mototaxista tem a opção de aceitar ou recusar a viagem, como mostra a Figura 23 a seguir:

Figura 23 – Nova corrida



Fonte: elaborado pela autora

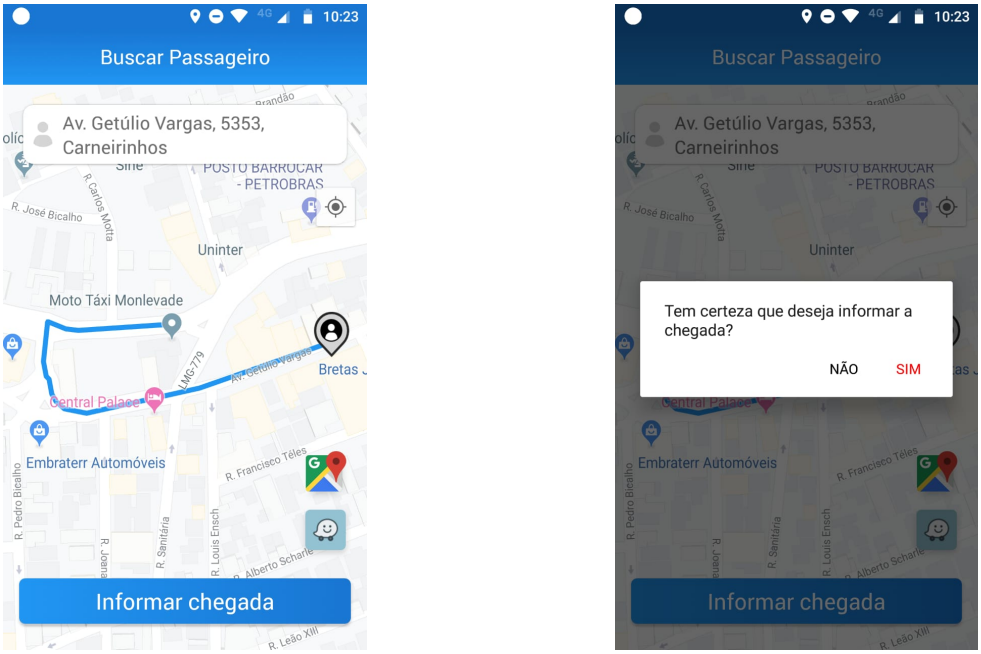
4.7 Opções de navegação

Após aceitar uma solicitação de viagem, o mototaxista tem acesso à rota da sua posição atual até o local de encontro com o passageiro. Visando melhorar a experiência do usuário, a aplicação oferece duas opções de navegação pelo mapa durante a viagem: um botão de acesso a rota pelo Google Maps e outro para o Waze.

4.8 Informar chegada

Essa etapa permite ao mototaxista informar ao passageiro que chegou para buscá-lo. Ao clicar no botão de Informar chegada, um alerta é exibido para que o usuário confirme sua ação, conforme a Figura 24.

Figura 24 – Informar chegada



(a) Informar chegada (b) Alerta

Fonte: elaborado pela autora

4.9 Iniciar corrida

Ao encontrar com o passageiro, o mototaxista deve iniciar a viagem, como indica a Figura 25. Nesse momento, a rota da sua localização até o destino final, já está sendo exibida no aplicativo.

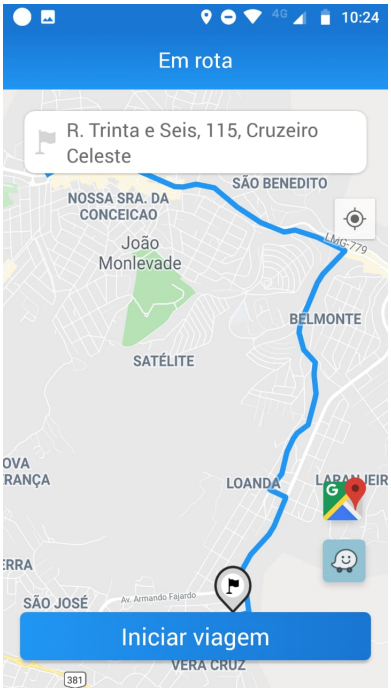
4.10 Finalizar corrida

Ao chegar no local de destino selecionado pelo passageiro, o mototaxista deve finalizar a viagem e confirmar sua ação, como mostra a Figura 26.

4.11 Avaliar corrida

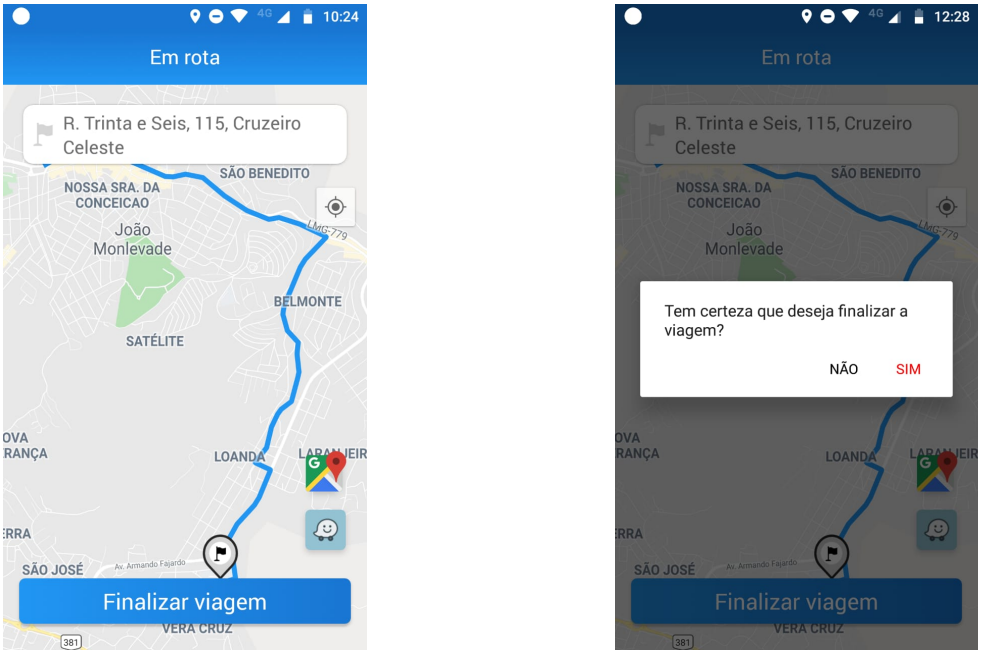
A etapa de avaliação, permite ao mototaxista avaliar o passageiro transportado através de uma nota de 1 a 5 e um texto de observação. A avaliação não é obrigatória e para finalizar o fluxo de corrida, basta clicar na seta de voltar no canto esquerdo superior da tela, ilustrado na Figura 27.

Figura 25 – Iniciar corrida



Fonte: elaborado pela autora

Figura 26 – Finalizar corrida



(a) Finalizar corrida (b) Alerta

Fonte: elaborado pela autora

Figura 27 – Avaliar corrida

Fonte: elaborado pela autora

5 Considerações finais

Apesar dos grandes avanços da sociedade, a mobilidade urbana muitas vezes ainda é um desafio no dia a dia das pessoas e muito explorado em termos de soluções ágeis e práticas. A cidade de João Monlevade, por exemplo, apesar de ter uma quantidade significativa de linhas de ônibus, possui duas universidades públicas que funcionam em período integral e afastadas do centro, o que torna necessário a utilização de meios de transporte para acessá-las com mais rapidez.

O presente trabalho, teve como principal objetivo apresentar o desenvolvimento de um aplicativo de mobilidade urbana, que visa conectar o mototaxista ocioso a um novo passageiro. Para o desenvolvimento do trabalho, foi necessário conhecimento multidisciplinar, envolvendo estudos sobre: *design* e experiência de usuário para a criação da interface; integração com APIs externas, como: Google Maps e Firebase; utilização do GPS do dispositivo para envio de coordenadas em tempo real; desenvolvimento com a linguagem Kotlin e utilização dos componentes de arquitetura Jetpack.

As principais dificuldades encontradas durante o desenvolvimento da aplicação, estão relacionadas as diferentes versões do Android e ao fato do sistema operacional (SO) ser *open source*, isso torna possível a alteração do *kernel* do SO pelas diferentes marcas produtoras de *smartphones*. Diante disso, foi necessário a implementação de um código que suportasse essas diferenças e partir da realização de testes em celulares distintos, foi possível concluir que as estratégias adotadas para o desenvolvimento foram adequadas e corroboram com uma solução satisfatória.

Os trabalhos futuros podem incluir:

- Melhorias no aplicativo desenvolvido, como monitoramento da conexão de rede e aplicação de filtro no histórico;
- Implementação de novas funcionalidades, como: métodos de pagamento, entrega de encomendas, *chat* com o passageiro;
- Desenvolvimento do aplicativo para o Passageiro;
- Desenvolvimento do *dashboard* para administração da aplicação;
- Desenvolvimento de uma API juntamente com as regras de negócio;

Referências

FIGUEIRA, G. M. *Mobilidade Colaborativa No Brasil: Um Estudo De Caso Sobre As Iniciativas De Carona Na Economia Colaborativa*. 2015. Disponível em: <http://www.inovarse.org/sites/default/files/T_15_027M_3.pdf>. Acesso em: 20 set. 2019. 4

FIREBASE. *Firebase Pricing*. 2019. Disponível em: <https://firebase.google.com/pricing/?hl=pt-br>. Acesso em: 15 dez. 2019. 16

GOOGLE CLOUD. *Build an Android App Using Firebase and the App Engine Flexible Environment*. 2019. Disponível em: <https://cloud.google.com/solutions/mobile/mobile-firebase-app-engine-flexible?hl=pt-br>. Acesso em: 30 set. 2019. 17

GOOGLE DEVELOPERS. *App Fundamentals*. 2018. Disponível em: <https://developer.android.com/guide/components/fundamentals>. Acesso em: 09 set. 2019. 7

GOOGLE DEVELOPERS. *Use o Android Jetpack para acelerar o desenvolvimento de aplicativos*. 2018. Disponível em: <https://developers-br.googleblog.com/2018/05/use-o-android-jetpack-para-acelerar-o.html>. Acesso em: 09 set. 2019. 8

GOOGLE DEVELOPERS. *Maps SDK for Android*. 2019. Disponível em: <https://developers.google.com/maps/documentation/android-sdk/intro>. Acesso em: 30 set. 2019. 15

IBGE. *Pesquisa de Informações Básicas Municipais 2013*. [S.l.], 2013. Disponível em: <ftp://ftp.ibge.gov.br/Perfil_Municipios/2013/munic2013.pdf>. Acesso em: 16 set. 2019. 4

KOTLIN. *Using Kotlin for Android Development*. 2019. Disponível em: <https://kotlinlang.org/docs/reference/android-overview.html>. Acesso em: 30 set. 2019. 14

KOTLIN ORG. *Kotlin Lang*. 2019. Disponível em: <https://kotlinlang.org/>. Acesso em: 30 set. 2019. 13

LUIZ TOOLS. *Fundamentos de Aplicações Android*. 2011. Disponível em: <http://www.luiztools.com.br/post/fundamentos-de-aplicacoes-android/>. Acesso em: 09 set. 2019. 7

PARSE. *Parse Server Guide*. 2019. Disponível em: <https://docs.parseplatform.org/parse-server/guide/#getting-started>. Acesso em: 10 oct. 2019. 14

SHI, J. *Handle FCM messages on Android*. 2018. Disponível em: <https://firebase.googleblog.com/2018/09/handle-fcm-messages-on-android.html>. Acesso em: 10 out. 2019. 16, 17

TRIPATHI, A. *What's new in Android @ I/O'19*. 2019. Disponível em: <https://medium.com/mindorks/whats-new-in-android-i-o-19-aabb0a59a3ae>. Acesso em: 30 set. 2019. 14

UBER. *História da Uber*. 2009. Disponível em: <https://www.uber.com/pt-BR/newsroom/historia/>. Acesso em: 16 set. 2019. 4