



UNIVERSIDADE FEDERAL DE OURO PRETO
ESCOLA DE MINAS
COLEGIADO DO CURSO DE ENGENHARIA DE
CONTROLE E AUTOMAÇÃO - CECAU



TULIO DE OLIVEIRA PARREIRAS

DESENVOLVIMENTO DE DOIS APLICATIVOS IOS TIPO UBER

MONOGRAFIA DE GRADUAÇÃO EM ENGENHARIA
DE CONTROLE E AUTOMAÇÃO

Ouro Preto, 2019

TULIO DE OLIVEIRA PARREIRAS

DESENVOLVIMENTO DE DOIS APLICATIVOS IOS TIPO UBER

Monografia apresentada ao Curso de Engenharia de Controle e Automação da Universidade Federal de Ouro Preto como parte dos requisitos para obtenção do Grau de Engenheiro de Controle e Automação.

Orientador: Prof. Dr. Ricardo Augusto Rabelo Oliveira

Coorientadora: Prof^a. Dr^a. Luciana Gomes Castanheira

Ouro Preto

Escola de Minas – UFOP

Julho/2019

P259d Parreiras, Tulio de Oliveira.
Desenvolvimento de dois aplicativos iOS tipo Uber [manuscrito] / Tulio de Oliveira Parreiras. - 2019.

95f.: il.: color.

Orientador: Prof. Dr. Ricardo Augusto Rabelo Oliveira.
Coorientadora: Prof^a. Dr^a. Luciana Gomes Castanheira.

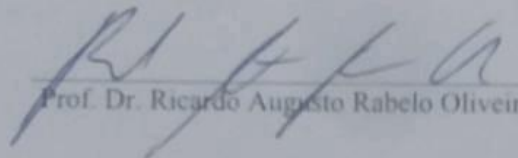
Monografia (Graduação). Universidade Federal de Ouro Preto. Escola de Minas. Departamento de Engenharia de Controle e Automação e Técnicas Fundamentais.

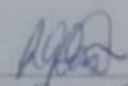
1. Mobilidade urbana. 2. Smartphone. 3. iOS. 4. Uber. 5. Desenvolvimento de software. I. Oliveira, Ricardo Augusto Rabelo. II. Castanheira, Luciana Gomes. III. Universidade Federal de Ouro Preto. IV. Título.

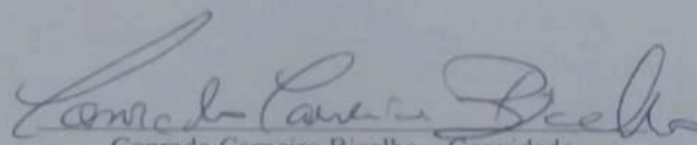
CDU: 681.5

Catálogo: ficha.sisbin@ufop.edu.br

Monografia de título Desenvolvimento de Aplicativo tipo uber defendida e aprovada, em 15 de julho de 2019, pela comissão avaliadora constituída pelos professores:


Prof. Dr. Ricardo Augusto Rabelo Oliveira - Orientador


Profa. Dra. Luciana Gomes Castanheira - Co-orientadora


Conrado Carneiro Bicalho - Convidado

AGRADECIMENTOS

Agradeço primeiramente a minha família, principalmente meus pais e meus irmãos que sempre me apoiaram. A minha noiva Juliana, por estar ao meu lado e me apoiar. Agradeço a UFOP e seus professores pelos ensinamentos, em especial ao meu orientador Ricardo Rabelo e minha co-orientadora Luciana Gomes por me guiarem ao longo desse trabalho. Gostaria de agradecer também a Usemobile pelas oportunidades e aprendizados, especialmente ao Conrado Carneiro e sua forte liderança.

RESUMO

Mobilidade urbana consiste nos meios utilizados pela população de um perímetro urbano para se deslocar dentro do mesmo, que, devido ao crescente desenvolvimento de grandes cidades é um desafio contínuo. Novas ideias e soluções surgem constantemente para suprir essa necessidade de locomoção, como o serviço oferecido pela empresa Uber e seus derivados, que têm oferecido uma opção prática, confiável e acessível para suprir a demanda.

Analisou-se os aplicativos motorista e passageiro da Uber e da concorrente brasileira 99, acompanhando como é funcionamento de ambos e seu desempenho. Com base nos estudos realizados foi definido o que compõe um aplicativo de mobilidade urbana e desenvolvido dois protótipos semelhantes, um aplicativo passageiro e um aplicativo motorista, de modo que ambos interagissem e fosse possível fazer uso de smartphones para requisitar viagens pagando através do próprio aplicativo passageiro.

Palavras-Chave: Dispositivos móveis, *smartphones*, iOS, iPhone, desenvolvimento de *software*, Uber, mobilidade urbana.

ABSTRACT

Urban mobility consists of the means used by the population of an urban perimeter to move within it, thus, because of the growing of the big cities it's a continuous challenge. New ideas and solutions emerge constantly to fill this need of locomotion, like the service provided from the company Uber and it's derivatives, who has been offering a practical, reliable and accessible option to fill that demand.

Has been analyzed the driver and passenger applications from the Uber and his Brazilian competitor 99, following how both of applications work and it's performance. Based on the realized studies has been defined what composes an urban mobile application and developed two similar prototypes, a passenger application and a driver application, in a way that both interact and be able to make use of a smartphone to request travels paying for the service within the passenger application.

Keywords: Mobile devices, smartphones, iOS, iPhone, software development, Uber, urban mobility.

LISTA DE FIGURAS

Figura 3.1 - Esquema, relação aplicativo, servidor e banco de dados	21
Figura 3.2 - Habilitando push notifications em seu projeto Xcode	30
Figura 4.1 - Tela de carregamento	33
Figura 4.2 - Tela inicial	34
Figura 4.3 - Login	35
Figura 4.4 - Cadastro.....	36
Figura 4.5 - Recuperar senha.....	37
Figura 4.6 - Mapa.....	38
Figura 4.7 - Busca por endereço	39
Figura 4.8 - Rota definida	40
Figura 4.9- Buscando motorista.....	41
Figura 4.10 - Viagem aceita por motorista.....	42
Figura 4.11 - Corrida aceita.....	43
Figura 4.12 - Avaliação do motorista	44
Figura 4.13 - Menu lateral.....	45
Figura 4.14 - Perfil.....	46
Figura 4.15 - Editar e-mail	47
Figura 4.16 - Editar celular.....	48
Figura 4.17 - Editar senha	49
Figura 4.18 - Histórico	50

Figura 4.19 - Detalhes do histórico.....	51
Figura 4.20 - Detalhes do histórico (ajuda).....	52
Figura 4.21 - Detalhes do histórico (recibo).....	53
Figura 4.22 - Pagamento	54
Figura 4.23 - Adicionar cartão.....	55
Figura 4.24 - Suporte e ajuda	56
Figura 4.25 - Suporte e ajuda (detalhes)	57
Figura 4.26 - Fale conosco	58
Figura 4.27 - Sair	59
Figura 4.28 - Carregamento (motorista).....	60
Figura 4.29 - Login (motorista)	61
Figura 4.30 - Tela inicial (motorista).....	62
Figura 4.31 - Recuperar senha (motorista).....	63
Figura 4.32 - Cadastro (motorista).....	64
Figura 4.33 - Consentimento legal.....	65
Figura 4.34 - Dados pessoais.....	66
Figura 4.35 - Categoria	67
Figura 4.36 - Dados do veículo	68
Figura 4.37 - Habilitação.....	69
Figura 4.38 - Mapa (motorista).....	70
Figura 4.39 - Nova corrida	71

Figura 4.40 - Corrida aceita.....	72
Figura 4.41 - Iniciar viagem.....	73
Figura 4.42 - Avaliar passageiro.....	74
Figura 4.43 - Menu lateral (motorista).....	75
Figura 4.44 - Perfil (motorista).....	76
Figura 4.45 - Cartões.....	77
Figura 4.46 - E-mail.....	78
Figura 4.47 - Telefone.....	79
Figura 4.48 - Alterar senha (motorista).....	80
Figura 4.49 - Financeiro.....	81
Figura 4.50 - Relatórios.....	82

SUMÁRIO

1	INTRODUÇÃO	16
1.1	Objetivo geral	16
1.2	Objetivos específicos	17
1.3	Justificativa do trabalho	17
1.4	Metodologia proposta	17
1.5	Estrutura do trabalho	17
2	APLICATIVOS DE MOBILIDADE URBANA	18
2.1	Aplicativo motorista.....	20
2.2	Aplicativo passageiro.....	21
2.3	Aplicativos de mobilidade urbana	21
2.3.1	Uber.....	21
2.3.2	99	22
2.4	Uberização.....	23
2.4.1	Uberização em outros negócios.....	24
3	DESENVOLVIMENTO IOS	25
3.1	Ambiente de desenvolvimento	25
3.2	Linguagem de programação	25
3.3	Padrão de projeto	25
3.4	Comunicação com servidor	28
3.4.1	Alamofire	29
3.5	Push Notifications.....	30
3.6	Autenticação	33

3.6.1	Parse	33
3.7	Sincronização de dados	34
3.7.1	Realtime Database (Firebase)	34
3.8	Mapa	36
3.8.1	Google Maps	36
3.9	Corrida	37
3.9.1	Places SDK para iOS	37
3.9.2	Directions API.....	38
3.9.3	Corrida no app passageiro.....	38
3.9.4	Corrida no app motorista	39
3.10	Chat.....	40
3.11	Menu	40
3.11.1	Menu lateral passageiro	41
3.11.2	Menu lateral motorista	42
4	PRODUTO	44
4.1	Aplicativo passageiro.....	44
4.1.1	Tela de carregamento.....	44
4.1.2	Tela inicial.....	45
4.1.3	Login.....	46
4.1.4	Cadastro.....	47
4.1.5	Recuperar senha	48
4.1.6	Mapa	49
4.1.7	Busca por endereço.....	50

4.1.8	Rota definida	51
4.1.9	Buscando motorista	52
4.1.10	Viagem aceita por motorista	53
4.1.11	Corrida iniciada	54
4.1.12	Avaliar o motorista	55
4.1.13	Menu.....	56
4.1.14	Perfil.....	57
4.1.15	Editar e-mail.....	58
4.1.16	Editar celular	59
4.1.17	Editar senha.....	60
4.1.18	Histórico.....	61
4.1.19	Detalhes do histórico – detalhes.....	62
4.1.20	Detalhes do histórico - ajuda.....	63
4.1.21	Detalhes do histórico - recibo	64
4.1.22	Pagamento.....	65
4.1.23	Adicionar cartão	66
4.1.24	Suporte e ajuda	67
4.1.25	Suporte e ajuda - detalhes	68
4.1.26	Fale conosco.....	69
4.1.27	Sair.....	70
4.2	Aplicativo motorista.....	71
4.2.1	Tela de carregamento.....	71
4.2.2	Tela inicial.....	72

4.2.3	Tela de login.....	73
4.2.4	Tela de recuperar senha	74
4.2.5	Tela de cadastro.....	75
4.2.6	Tela de consentimento legal.....	76
4.2.7	Tela dados pessoais	77
4.2.8	Categoria.....	78
4.2.9	Dados do veículo	79
4.2.10	Habilitação	80
4.2.11	Mapa.....	81
4.2.12	Tela nova corrida.....	82
4.2.13	Corrida aceita	83
4.2.14	Iniciar viagem.....	84
4.2.15	Avaliar passageiro	85
4.2.16	Menu lateral	86
4.2.17	Perfil.....	87
4.2.18	Perfil - Cartões	88
4.2.19	Perfil - Email	89
4.2.20	Perfil - Telefone.....	90
4.2.21	Perfil - Alterar senha.....	91
4.2.22	Financeiro.....	92
4.2.23	Relatórios... ..	93
5	CONSIDERAÇÕES FINAIS	94
5.1	Sugestões para trabalhos futuros	94

REFERÊNCIAS BIBLIOGRÁFICAS.....	96
---------------------------------	----

1 INTRODUÇÃO

Em dezembro de 2008 dois amigos encontram dificuldades para conseguir um taxi em uma manhã de neve em Paris, o que pode parecer um fato cotidiano acabou acarretando no começo de uma ideia que hoje contagia o mundo inteiro, o aplicativo Uber. Travis e Garret tiveram naquele dia a ideia de criar um aplicativo para *smartphone* em que fosse possível requisitar uma viagem com o simples gesto de acionar um botão, e então em março de 2009 os dois empreendedores começaram uma empresa chamada até então, UberCap. (UBER, 2019?)

Em 5 de julho de 2010, o UberCap conecta seu primeiro usuário com um carro particular preto em São Francisco. A partir desse ponto a empresa foi mudando e crescendo, alterou seu nome para o que hoje conhecemos como Uber. Em abril de 2014 alcançou a marca de 100 cidades sendo atendidas pelo serviço. Hoje em dia a empresa já conta com números impressionantes, são 75 milhões de usuários, 10 bilhões de viagens, 65 países em uso e mais de 15 milhões de viagens por dia. (UBER, 2019?)

Esse sucesso tem feito com que diversas empresas tenham lançado no mercado produtos com uma essência parecida, oferecendo um serviço parecido, de prover uma opção de renda para algumas pessoas e um transporte alternativo para outras.

Mas o que é necessário para que um produto desse tipo seja lançado no mercado e seja de fato utilizado? São diversas as variáveis envolvidas no desenvolvimento de um projeto similar ao Uber, que vão desde uma pesquisa no mercado, desenvolvimento de um sistema e promoção do mesmo. Para que assim seja possível chegar a um produto sólido que irá atingir e impactar os usuários.

Ao explorar a elaboração e criação de um sistema, podemos destrinchá-lo em dois produtos, um aplicativo que irá fornecer ao usuário uma ferramenta para encontrar os profissionais disponíveis, que é o aplicativo voltado ao passageiro. E também um aplicativo em que as pessoas poderão oferecer seus serviços e assim encontrar pessoas dispostas a pagar pelos seus serviços, sendo esse o aplicativo para o motorista.

1.1 Objetivo geral

Desenvolver dois aplicativos com o foco de oferecer um serviço de transporte alternativo.

1.2 Objetivos específicos

Criar dois aplicativos voltados para a plataforma iOS, que consistem em um aplicativo para passageiro e um para motorista, que irão comunicar entre si, oferecendo um serviço de transporte entre dois locais definidos pelo passageiro.

1.3 Justificativa do trabalho

A necessidade de se deslocar diariamente pelas mais diversas razões que existem, seja ir ao trabalho, uma consulta ou lazer, é muitas vezes um desafio para quem vive nas cidades. Engarrafamentos, aumento em tarifas, precariedade do transporte público, são alguns dos exemplos de fatores que podem tornar a mobilidade um problema. Pensando nesse cenário, é cada vez mais necessário buscar novas alternativas de transporte acessível e eficiente.

1.4 Metodologia proposta

Estudo, desenvolvimento e criação de dois aplicativos para plataforma iOS utilizando como linguagem de programação Swift e ambiente de desenvolvimento Xcode.

1.5 Estrutura do trabalho

Este trabalho foi dividido em cinco capítulos. O primeiro é responsável por apresentar e introduzir o leitor sobre o trabalho desenvolvido. O segundo capítulo apresenta um estudo sobre os temas pertinentes a parte prática. No terceiro capítulo é apresentado os módulos que compõe a aplicação enquanto os mesmos são explicados e detalhados. O quarto capítulo apresenta os resultados obtidos no desenvolvimento da aplicação. No quinto capítulo são apresentadas a considerações finais assim como as sugestões para trabalhos posteriores.

2 APLICATIVOS DE MOBILIDADE URBANA

Um aplicativo que facilite uma pessoa sair de um local e ir para outro pode ser considerado como um aplicativo de mobilidade urbana. Para o presente trabalho serão considerados apenas os aplicativos que envolvem uma terceira parte que consiste em um motorista em um carro. (GALANTE, 2019)

Atualmente, apenas cerca de 10% da população brasileira faz uso de algum aplicativo de mobilidade urbana. Esse mercado pode ser ampliado ainda mais pelos erros cometidos pela Uber, tendo em vista que a mesma atua apenas em cidades grandes, agora outras empresas têm surgido para suprir essa demanda das cidades menores. (GALANTE, 2019)

Nas seções subsequentes será apresentada uma breve descrição sobre os aplicativos de mobilidade urbana e seu cenário atual.

2.1 Mobilidade urbana

As cidades possuem como objetivo principal facilitar o câmbio de bens e serviços, conhecimentos e cultura entre sua população, porém isso só ocorre se existirem condições de mobilidade condizentes para seus habitantes. Tendo isso em vista, a mobilidade é uma propriedade relacionada à cidade e equivale à capacidade de deslocamento de bens e pessoas no perímetro urbano. A mobilidade consiste nas relações entre as pessoas e o ambiente no qual os mesmos habitam, com os meios e objetos utilizados pelos mesmos para se locomover e com as outras pessoas que compõem a sociedade. (COSTA, 2008)

A mobilidade consiste na capacidade de pessoas em participar de atividades diferentes em lugares diferentes e na possibilidade de acessar as atividades que estão relacionadas a sua produção e comercialização, que ocorrem constantemente em lugares diferentes. Com relação ao transporte de pessoas, essas atividades compõem a residência, o trabalho, lazer, compras e educação. As cidades devem dar suporte à mobilidade de maneira que cumpra sua função social e proporcione crescimento econômico, da mesma maneira que devem buscar frear o crescimento do tráfego motorizado e seus resultados negativos sobre a população e o meio ambiente. (COSTA, 2008)

O cuidado e atenção constantes as questões relacionadas ao desenvolvimento sustentável têm encorajado e incentivado o estudo e implantação de ações, em diferentes áreas, que possam contribuir para com a sustentabilidade em regiões urbanas. Essas preocupações

quando voltadas a área de transporte, podem ser observadas através de ações que promovem uma busca por uma mobilidade urbana mais sustentável. Essa busca por uma mobilidade urbana mais sustentável, deve seguir como base o conceito de desenvolvimento sustentável de modo que se procura definir estratégias que contemplam um conjunto de questões: econômicas, sociais e ambientais. Deve-se estar atento também ao mais conhecido princípio do desenvolvimento sustentável, que diz que o desenvolvimento sustentável é uma forma de desenvolvimento que tem como objetivo suprir as necessidades da geração atual de modo que não comprometa a possibilidade de que as gerações futuras possam satisfazer as suas necessidades. (CAMPOS, 2006)

2.1.1 Desafios na mobilidade urbana

Cidades inteligentes têm como papel fornecer soluções de mobilidade que sejam eficientes e que devem também incentivar a inovação de modo a tornar mais acessível um sistema de colaboração com objetivos sustentáveis. Esses desafios fazem parte de um cenário de mudanças no cenário da mobilidade urbana que têm acontecido de uma forma rápida. (GLASCO, 2019)

Existem poucos lugares em que o cenário da mobilidade urbana atende as expectativas das pessoas em termos de segurança, confiabilidade, limpeza e formas acessíveis de se locomover entre dois pontos. Um constante desafio de grandes cidades é propor maneiras de melhorar a mobilidade enquanto se reduz o número de acidentes, congestionamentos e poluição. (GLASCO, 2019)

Existem quatro principais problemas em transporte urbano que necessitam de quatro soluções separadas, e é importante que as pessoas não confundam a solução para um problema com a solução de um problema diferente. (MCMAHON, 2018)

1. Comunicação: uma tensão é gerada entre sistemas de trânsito e usuários quando os usuários não têm as informações das quais precisam e quando precisam das mesmas. Esse problema tem sido resolvido pelo uso de aplicativos de transporte, que fornecem informações em tempo real para os usuários.
2. Emissões e eficiência energética: a solução para esse problema tem sido os carros elétricos, que não geram poluentes durante seu uso.
3. Mão-de-obra e segurança: a mão-de-obra é o principal custo de operação para passageiros de transportes, e também o principal fator envolvido na questão da

segurança, em termos de acidentes de trânsito. Para esse problema a solução é o investimento feito em carros autônomos que irão remover a necessidade de um motorista ao mesmo tempo em que irá fornecer uma alternativa mais segura de transporte.

4. Espaço: o quarto problema é a questão do uso eficiente do espaço, tendo em vista que os veículos automotivos demandam de um grande espaço por serem grandes. Esse problema pode ser resolvido através do transporte de bicicleta, a pé ou então no uso de transportes públicos. (MCMAHON, 2018)

Quando falamos de transporte público temos a opção de ônibus e trem de rotas fixas, que são capazes de mover uma grande quantidade de pessoas em uma cidade em que o seu espaço se encontra congestionado. Porém as pessoas têm dado preferência ao transporte individual em que o mesmo vai onde você deseja e quando você deseja, atendendo as suas necessidades de uma maneira mais eficiente, porém não sendo capaz de mover um número grande de indivíduos ao mesmo tempo. Em contrapartida o transporte individual exerce uma vantagem com relação ao transporte público de modo que muitas vezes é possível ver um ônibus circulando vazio ou com poucos passageiros pela cidade, isso ocorre pela necessidade de atender a todas as pessoas até em localizações mais remotas, o que gera um custo mais elevado, pois ocorre de uma rota de ônibus ser definida para atender um número menor de pessoas. Quando falamos nesse problema, os meios de transporte individuais são uma boa opção pois ocupam um menor espaço, poluem menos e conseguem atender essas pessoas. O cenário ideal seria então um trabalho conjunto entre ambos de modo que em regiões com grande fluxo de pessoas a opção utilizada seria o transporte público e em locais em que há um fluxo menor de pessoas o transporte individual iria atender esse público. (MCMAHON, 2018)

2.2 Aplicativo motorista

A Uber e empresas que possuem serviços similares, fazem uso de aplicativos que são desenvolvidos para aparelhos *smartphones* e *tablets*, tanto para plataforma Android quanto iOS, que possuem como foco conectar usuários que são motoristas de automóveis particulares e oferecem um serviço de transporte que pode ser individual ou não, já existem categorias de transporte coletivo (UBER juntos), e remunerado de passageiros. Um proprietário de um automóvel que se interessar em oferecer esse serviço deve realizar o seu cadastro junto à empresa, e passar por um processo em que deve comprovar adequação a uma lista de requisitos, como sua habilitação como motorista e seus antecedentes criminais. Após o término

desse procedimento e sua validação, o mesmo pode então disponibilizar seus serviços na plataforma, ficando acessível aos passageiros ali presentes. A plataforma, em concordância com o motorista, procede o pagamento do serviço de transporte, realizando o desconto de maneira pré-definida sobre o valor pago pelo passageiro, que pode ser um percentual do montante ou um valor fixo.

2.3 Aplicativo passageiro

O passageiro, faz o *download* do *software* em seu dispositivo e realiza o cadastro, informando seus dados pessoais e de pagamento. Por meio do programa ele solicita um motorista credenciado na empresa quando necessita de fazer uso dos serviços de transporte fornecidos pela mesma. O pagamento pelo trabalho é feito de acordo com as opções previstas pela plataforma, que podem variar entre dinheiro, cartão ou até alguma rede de créditos/bônus fornecida pela mesma. O serviço em si compreende em encontrar o passageiro no local definido pelo mesmo e então leva-lo ao destino solicitado através do aplicativo. O preço pago é calculado com base em uma regra pré-definida pela companhia que está relacionada diretamente a distância percorrida e tempo gasto durante a viagem.

2.4 Aplicativos de mobilidade urbana

O avanço na tecnologia dos *smartphones* somado à crescente melhoria na facilidade de acesso à internet móvel, tem feito com que diversos aplicativos de transporte individual fossem lançados no mercado, se tornando parte do cotidiano da população. Foram escolhidos dois dos aplicativos mais utilizados na plataforma iOS para serem analisados e estudados. O parâmetro usado como escolha dos mesmos foi o recurso disponibilizado pela própria loja de aplicativos da *Apple* (*App Store*), que informa os aplicativos (apps) grátis mais usados, o Uber e 99.

2.4.1 Uber

A Uber é a pioneira no serviço de transporte via aplicações móveis, que consta como o aplicativo (app) nº 1 na categoria de viagens da *App Store* e conta com mais de 300 mil avaliações na loja em sua versão passageiro e cerca de 50 mil avaliações na versão motorista, que por sua vez ocupa a sexta posição de apps na categoria negócios. (Apple, 2019?)

O passageiro do aplicativo Uber, após finalizar o cadastro, pode realizar o pedido de uma corrida selecionando a caixa de endereço “Para onde?”, localizada no topo da tela principal do app e inserir um destino, sendo possível também editar o local de partida, assim com poucos

gestos já consegue solicitar um motorista para sua viagem. O cliente pode escolher entre pagar pelo serviço utilizando um cartão de débito ou crédito, que fica salvo em suas preferências para ser utilizado em viagens subsequentes. Quando o motorista aceita a solicitação de corrida, o usuário pode acompanhar a localização atual do motorista e o trajeto até seu encontro, sendo informado sobre alguns dados pertinentes como o tempo estimado do deslocamento ou até uma orientação relacionada ao ponto de encontro, como um portão de saída de um shopping. O passageiro pode também entrar em contato com o motorista pelo app através de mensagem de texto, facilitando a comunicação entre ambos. Após finalizada a corrida, a plataforma realiza a cobrança pelo serviço automaticamente e é disponibilizado para o passageiro a opção de avaliar o motorista pelo serviço prestado.

O motorista parceiro da Uber, após a finalização do cadastro, verificação e aprovação como parceiro, pode começar a atuar através do simples gesto de acionar um botão, com título “Iniciar”, que vai habilitar o mesmo a receber solicitações de passageiros. Ao receber um pedido de viagem, são disponibilizadas algumas informações ao motorista, como a categoria da corrida, distância e tempo estimado com relação ao ponto de encontro com o passageiro e também a média de avaliações do passageiro. Se o motorista optar por aceitar o pedido, é exibido o trajeto a ser percorrido no mapa e o recurso de navegação, que consiste em orientações para o mesmo conseguir chegar ao ponto de encontro e pode ser utilizado tanto dentro do próprio app, ou através de terceiros, como Waze ou Google Maps. O grande diferencial fica por conta do sistema próprio de navegação dentro do app, que não é disponível em concorrentes, possibilitando adaptações e ajustes de acordo com sua própria necessidade.

Dentro do proposto pela empresa, os aplicativos motorista e passageiro são eficientes e eficazes em cumprir seu objetivo, a realização de viagens de forma simples e acessível. Vale ressaltar que ambos apresentam desempenhos questionáveis quando em situações em que a conexão com a internet apresenta lentidão ou o dispositivo usado possui configurações mais inferiores. O que é compreensível, quando se é levado em conta a grande quantidade de informações e recursos disponíveis para o usuário durante o uso da plataforma.

2.4.2 99

O maior concorrente da Uber no Brasil, 99, é uma empresa fundada em 2012 que também propõe uma solução de transporte por aplicações móveis. A mesma já conta com 18 milhões

de passageiros, 600 mil motoristas e atuam em mais de mil cidades, apenas no Brasil. (99, 2019?)

Seu aplicativo para passageiro é o nº 2 na categoria de viagens na loja App Store, contando com mais de 900 mil avaliações. Já a versão voltada para motorista é a nº 5 na categoria negócios e possui mais de 1,5 mil avaliações. (Apple, 2019?)

O passageiro que utiliza o 99, após finalizar o processo de cadastro com sucesso, tem acesso ao mapa e pode então pedir sua corrida, de modo similar ao Uber. Esta plataforma tem como adicional o recurso chamado “Embarque rápido”, em que o cliente pode fazer uso da câmera para escanear o qr code de um motorista e iniciar uma corrida de forma rápida.

Por sua vez, o motorista da 99, tendo finalizado seu processo de cadastro e ser validado, pode tanto habilitar quanto desabilitar chamadas de corrida através de um botão no topo da tela, de modo prático e eficiente. O aplicativo motorista 99 tem como diferencial o recurso do motorista visualizar no mapa tanto a origem quanto destino de uma chamada, enquanto visualiza também sua própria localização, podendo decidir de forma melhor se a vagem é de seu interesse ou não.

Em termos de usabilidade, a plataforma 99 também cumpre bem seu objetivo tanto na versão motorista quanto passageiro, sendo afetada pelo mesmo problema que a Uber, que em situações não ideias de internet ou hardware, deixa a desejar um pouco.

2.5 Uberização

Quando a empresa Uber lançou seu aplicativo em 2010, em conjunto com esse lançamento se deu início a uma ideia, de disponibilizar para os usuários de aplicativos móveis uma forma de renda extra, através da prestação de um serviço. Com o sucesso da empresa e seu formato de negócio, surgiu um novo termo: uberização. Que seria um formato de negócios que faz uso de tecnologias móveis para conectar o consumidor ao fornecedor de um produto/serviço. No Brasil, esse tipo de serviço se tornou mais popular em 2016 e tem crescido cada vez mais. (GALANTE, 2018)

O grande diferencial desse modelo de trabalho é que o meio de produção é detido pelo próprio profissional, ou seja, se em algum momento a empresa fecha suas portas ou o próprio profissional decide não ser mais parceiro da empresa, ele pode buscar outra forma de prestar seu serviço sem ser prejudicado, pois ele continua tendo os meios para isso. No caso de

aplicativos de mobilidade urbana, o seu meio de trabalho seria o automóvel, e mesmo que a parceria entre a empresa e prestador deixe de existir, o prestador continua tendo seu veículo disponível para trabalhar (GALANTE, 2018)

2.5.1 Uberização em outros negócios

Este conceito da forma descrita, não fica preso apenas ao modelo de transporte, sendo aplicado a outros tipos de serviço, tais como:

- Serviços para casa: existem aplicativos em que é possível encontrar prestadores de serviço para tarefas de casa, tais como limpeza, reparos, jardinagem e outros. Um exemplo seria o aplicativo Sem Patrão;
- Delivery: atualmente temos uma grande popularização no uso de smartphones para facilitar o delivery, principalmente no setor alimentício, com aplicativos como iFood, aiqfome e Uber Eats;
- Hospedagem: os serviços de hospedagem, seja de um quarto ou uma casa, já aderiram também a esse modelo, sendo possível encontrar diferentes opções usando aplicativos como AirBnb. (GALANTE, 2018)

3 DESENVOLVIMENTO IOS

Será agora abordado sobre o modelo de desenvolvimento para a plataforma iOS, onde serão apresentados as ferramentas envolvidas na construção da aplicação, tais como técnicas, métodos e padrões utilizados para otimizar o processo. Também serão apresentados os módulos que irão compor a aplicação.

3.1 Ambiente de desenvolvimento

Para construir um projeto voltado para a plataforma iOS, é necessário primeiro escolher qual a ferramenta a ser utilizada para se gerenciar o mesmo. O mercado oferece algumas opções diferentes, cada uma com seus recursos e funcionalidades, como Xcode que é próprio da Apple, AppCode da JetBrains ou então o Atom, utilizado por quem usa Linux. Para a realização deste projeto foi selecionada a ferramenta Xcode, por uma questão de familiaridade com a mesma.

3.2 Linguagem de programação

Tendo escolhido o ambiente de desenvolvimento, também chamado IDE, é necessário então escolher qual linguagem de programação a ser utilizada para se escrever as linhas de código. O Xcode permite ao programador escolher entre duas, o Swift e o Objective-C. A linguagem adotada foi o Swift, que atualmente é a linguagem oficial para iOS mais adotada no mercado.

3.3 Padrão de projeto

Para a construção do projeto é importante ser definido um padrão a ser seguido, de modo a ter um código mais organizado, estruturado e reutilizável. Ou seja, a adoção de um padrão de projeto permite ao programador trabalhar de forma mais organizada e eficiente, permitindo reduzir a complexidade de criar um software e também de dar manutenção no mesmo. É possível encontrar na comunidade alguns padrões já definidos como o MVC (*Model-View-Controller*), MVVM (*Model-View-ViewModel*), MVP (*Model-View-Presenter*), VIPER (*View-Interactor-Presenter-Entity-Router*) e outros.

Foi decidido o uso do padrão de projeto MVVM, que é um padrão com foco de modularizar mais o projeto, dividindo o mesmo em grupos distintos cada um com uma única responsabilidade, tornando o projeto mais bem subdividido e organizando, facilitando até

mesmo correções, uma vez que o desenvolvedor já irá saber qual grupo procurar para realizar um ajuste. O MVVM é composto pelos seguintes módulos:

- *Model*: é responsável pela representação das informações/dados na aplicação;
- *View*: composta pela parte visual da aplicação, responsável por exibir as informações ao usuário e permitir interações do usuário com a aplicação;
- *ViewModel*: tem como responsabilidade intermediar a comunicação entre as camadas *Model* e *View*, notificando o modelo quando ocorrem interações na camada visual, de modo a alterar os dados, e também de informar a camada visual se o modelo foi atualizado pelo servidor, para que a mesma sempre apresente informações atualizadas e consistentes.

Usando como base a tela em que o usuário irá realizar o *login* na aplicação, que é composta pelos módulos abaixo:

- Duas entradas de texto (e-mail e senha);
- Botão para fazer a requisição de *login* (usando o e-mail e senha informados);
- Botão para acessar a tela de cadastro;
- Botão para acessar a tela de recuperar senha;

```
public class LoginViewController: UIViewController {
    lazy var loginView: LoginView = {
        let view = LoginView()
        view.delegate = self
        return view
    }()

    public override func loadView() {
        self.view = self.loginView
    }
    ...
}

class LoginView: UIView {
    weak var delegate: LoginViewDelegate?

    var email = ""
    var password = ""

    @objc func enterPressed(_ sender: UIButton) {
        self.endEditing(false)
        self.delegate?.loginView(self,
            didTapEnterFor: LoginViewModel(email: self.email,
```

```

        password: self.password))
    }

    @objc func forgotPasswordPressed(_ sender: UIButton) {
        self.endEditing(false)
        self.delegate?.loginView(self, didTapForgotPasswordFor: self.email)
    }

    @objc func registerPressed(_ sender: UIButton) {
        self.endEditing(false)
        self.delegate?.loginView(self,
            didTapRegisterFor: LoginViewModel(email: self.email,
                password: self.password))
    }
}
....
}

protocol LoginViewDelegate: class {
    func loginView(_ view: LoginView, didTapEnterFor viewModel: LoginViewModel)
    func loginView(_ view: LoginView, didTapForgotPasswordFor email: String)
    func loginView(_ view: LoginView, didTapRegisterFor viewModel: LoginViewModel)
}

public class LoginViewModel {

    public var email: String
    public var password: String

    public init(email: String,
        password: String) {
        self.email = email
        self.password = password
    }

    public func getJSON() throws -> [String: Any] {
        let passwordMinChar = 6
        if !self.email.isValidEmail {
            throw CustomError("Email inválido")
        }
        if self.password.count < passwordMinChar {
            throw CustomError("A senha deve conter ao menos \(passwordMinChar) caracteres")
        }
        let json: [String: Any] = ["login": self.email,
            "password": self.password]
        return json
    }
}
}

```

No código acima é apresentada a *LoginViewController* que tem como função iniciar a *LoginView* que é a *View* do padrão MVVM, sendo responsável por receber os eventos do usuário. A medida que o usuário interage com a *View*, caso o mesmo opte por realizar o *login*, é gerado uma instância de um *LoginViewModel* que irá conter os dados de e-mail e senha. Quando for realizado a requisição, esse *LoginViewModel* irá gerar um JSON (através do

método *getJSON*), que por sua vez é o *Model* do MVVM, que será então enviado para o servidor na requisição de *login*.

É importante destacar o padrão utilizado para fazer a comunicação entre as classes *LoginViewController* e a classe *LoginView*, representado pelo protocolo *LoginViewDelegate*.

Esse padrão é conhecido como *Delegate* e é muito utilizado durante o desenvolvimento de aplicações iOS. O mesmo tem como função realizar a comunicação entre duas classes diferentes, podendo passar informações entre as mesmas durante essa comunicação. Tendo como base o exemplo apresentado, temos que o *LoginViewDelegate* faz a comunicação entre as classes através de três funções que serão detalhadas abaixo:

- *didTapEnterFor viewModel*: essa função tem como objetivo da *LoginView* informar a *LoginViewController* de que o usuário apertou o botão de realizar *login*, e enviar o *LoginViewModel* que contém as informações preenchidas pelo usuário;

- *didTapForgotPasswordFor email*: desempenha o papel de informar a *viewController* de que o usuário deseja acessar a tela de recuperação de senha, passando como parâmetro o e-mail já informado na tela de *Login*, para que o mesmo não tenha que inserir o mesmo e-mail novamente;

- *didTapRegisterFor viewModel*: informa a *LoginViewController* de que o usuário selecionou o botão para ir a tela de cadastro, também envia as informações de e-mail e senha informados, para que o mesmo possa aproveitar esses dados já informados durante a etapa de cadastro;

Outro detalhe importante a ser observado pelo código é o uso do modificador *weak*, atribuído a variável *delegate*. Esse modificador é utilizado em projetos iOS para evitar ciclos de retenção. Uma vez que ao terminar de ser utilizada, uma classe deve ser desalocada da memória do celular, para que o aplicativo fique mais rápido. O uso desse modificador *weak* permite ao gerenciador de memória da *Apple* (ARC) que aquela instância da classe seja removida com sucesso da memória.

3.4 Comunicação com servidor

Uma etapa muito importante na construção de uma aplicação móvel é definir a comunicação com o servidor. A maioria das aplicações móveis necessita de um banco de dados externo e um servidor para funcionalidade que não podem ser executadas exclusivamente no próprio

dispositivo, como no caso do projeto em questão, a busca por um motorista. É necessário se ter um servidor por trás gerenciando os serviços disponíveis nos aplicativos.

O banco de dados é responsável por armazenar as informações pertinentes a aplicação, como dados dos usuários e histórico das corridas. O servidor contém as regras de negócio, serviços disponíveis (*endpoints*) e se comunica com o banco de dados, é responsável por retornar para os aplicativos as informações e recursos que serão disponibilizados para o usuário. Esse conjunto de regras de negócio e serviços é denominado API e no presente trabalho foi desenvolvida por Cláudio Madureira, que trabalha em conjunto com o time no desenvolvimento do projeto geral.

Foi optado pelo uso da biblioteca Alamofire para auxiliar a comunicação dos aplicativos motorista e passageiro com a API.

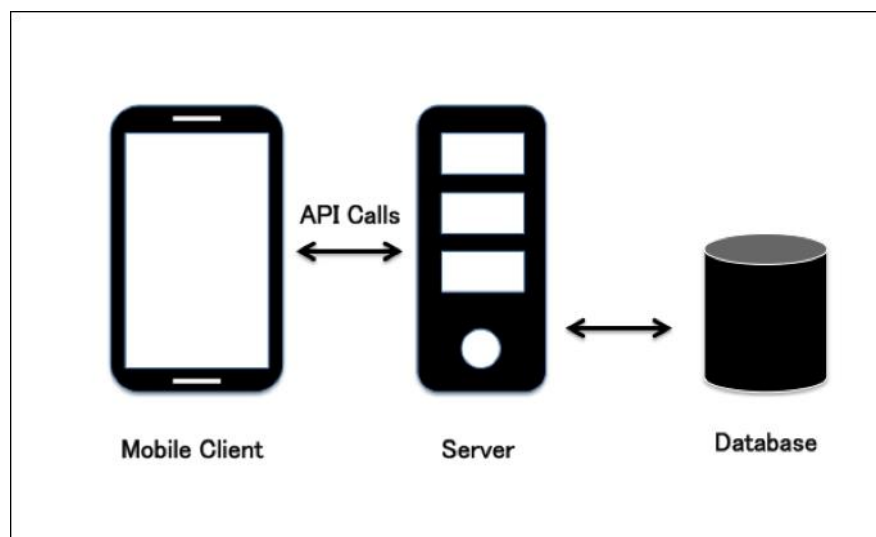


Figura 3.1 – Esquema, relação aplicativo, servidor e banco de dados

Fonte: KOTIPALLI, 2016

3.4.1 Alamofire

O Alamofire é uma biblioteca escrita em Swift de comunicação HTTP para iOS e macOS. Ela fornece ao desenvolvedor uma interface que simplifica as mais comuns tarefas de comunicação e trabalha com a estrutura JSON, que é uma formatação de troca de dados muito utilizada por aplicações Web e Mobile.

Um *endpoint* chamado através do Alamofire retorna um objeto JSON que é então serializado e transformado em um objeto Swift. Essa serialização é feita com o auxílio de outra biblioteca, o ObjectMapper, que vai mapear o JSON e formatar ele com base em uma

estrutura pré-definida pelo programador, buscando atributos relacionados a chaves que são definidas na estrutura. Por exemplo, ao requisitar os dados de um usuário, teremos campos como nome, e-mail e telefone, esses campos são identificados por chaves, que por convenção são definidas em inglês, o campo nome é identificado pela chave “name”, logo na estrutura teremos um atributo com o identificador “name”, para representar o nome do usuário.

```
public class API {
    public class func request(_ type: String, params: [String: Any], success: Success? = nil, failure: Failure? = nil {
        let link = API_LINK + "functions/" + type
        let header: [String: String] = ["Content-type": "application/json"]
        var parameters: [String: Any] = params
        parameters["_ApplicationId"] = APP_ID
        if let sessionToken = PFUser.current()?.sessionToken {
            parameters["_SessionToken"] = sessionToken
        }
        Alamofire.request(link, parameters: parameters, encoding: JSONEncoding.default, header: header)
            .responseJSON(completionHandler: { response in
                ....
            })
    })
}
```

A amostra de código acima exemplifica a base de uma requisição feita através do uso da biblioteca Alamofire. É informado um tipo (*type*) que é o identificador do *endpoint*, então a partir dele é gerado um link composto pela concatenação da *string* base da *url*, a *string* “*functions/*” que é adicionada para informar que está sendo acessada uma função, e o identificador *type*. Também é informado os parâmetros que serão enviados junto a requisição, que são atualizados adicionando o identificador da aplicação (*APP_ID*) e o identificador da sessão do usuário (*sessionToken*). É definido o cabeçalho da requisição que informa qual o tipo do conteúdo, no caso JSON. É feita a chamada da função “*request*” da biblioteca Alamofire sendo enviadas todas essas informações e então chamado a função “*responseJSON*” que irá fazer o envio para o servidor e aguardar pela resposta, que é definida dentro do campo “*completionHandler*”, onde o desenvolvedor informa o que será feito quando uma resposta for recebida do servidor, essa representada pela constante “*response*”, que pode conter um sucesso ou falha.

3.5 Push Notifications

Quando uma plataforma deseja informar um usuário de aplicações móveis sobre algo pertinente, uma notificação é disparada ao usuário, pelo aplicativo. Essa notificação é chamada *Push Notification (push)* e surgiu com o intuito de enviar alguma forma de informação ao usuário de forma simples e eficiente, para manter o mesmo acessando o

aplicativo. As empresas fazem uso das notificações para informar ao usuário sobre novos recursos, promoções ou qualquer informação que acreditem ser do interesse do usuário.

Nos aplicativos motorista e passageiro, os *pushs* são gerenciados de forma nativa, ou seja, informando ao próprio sistema operacional do dispositivo para exibir e esconder as mesmas, mantendo a experiência como a mais familiar possível ao usuário com a plataforma.

```
func requestNotificationAuthorization(_ application: UIApplication) {
    if available(iOS 10, *) {
        UNUserNotificationCenter.current().requestAuthorization(options: [.badge, .sound, .alert]){ (granted, error) in }
        application.registerForRemoteNotifications()
    } else {
        UIApplication.shared.registerUserNotificationSettings(UIUserNotificationSettings(types: [.badge, .sound, .alert],
categories: nil))
        UIApplication.shared.registerForRemoteNotifications()
    }
}

func application(_ application: UIApplication, didFinishLaunchingWithOptions launchOptions:
[UIApplication.LaunchOptionsKey: Any?] -> Bool {
    ...
    self.requestNotificationAuthorization(_ application: application)
    UNUserNotificationCenter.current().delegate = self
    return true
}
```

O código acima mostra como é configurado os *push notifications* dentro do aplicativo. Essa configuração é feita dentro do AppDelegate do projeto que consiste em uma que já vem definida dentro do *template* inicial de qualquer projeto. Essa classe tem como objetivo gerenciar todos os comportamentos compartilhados da aplicação, ela é a raiz do projeto e trabalha junto com a classe UIApplication para gerenciar algumas das interações da aplicação com o sistema operacional.

Dentro da classe AppDelegate, temos a função “*didFinishLaunchingWithOptions*” que é executada quando o aplicativo termina sua execução, sendo esse o momento para serem feitas algumas configurações gerais do projeto, assim como a definição dos *push notifications*. É o momento que é chamada a função “*requestNotificationAuthorization*” que desempenha o papel de pedir autorização ao usuário para que o aplicativo possa exibir notificações, dentro e fora do app. Já a linha de código “UNUserNotificationCenter.current().delegate = self” é utilizada para informar a aplicação que a mesma deve gerenciar o centro de notificações, sendo assim possível exibir os *pushs* de forma nativa quando o aplicativo estiver aberto. Quando o app está minimizado, o próprio sistema gerencia automaticamente todos os alertas emitidos por todos aplicativos instalados.

É importante ressaltar, que dentro da função “*requestNotificationAuthorization*” há uma condição “if” em que é verificado qual a versão do sistema operacional utilizado pelo usuário. Essa verificação é feita pois de acordo com o lançamento de novas versões e atualizações do sistema operacional (iOS), novos recursos e ferramentas são disponibilizadas para o desenvolvedor, fazendo com que as classes utilizadas possam ser diferentes ou possuir uma nova sintaxe.

Para que a funcionalidade de receber *pushs* funcione ainda há mais uma etapa a ser realizada, que é habilitar essa funcionalidade dentro do painel de funcionalidades nas configurações do projeto, como ilustrado na imagem abaixo:

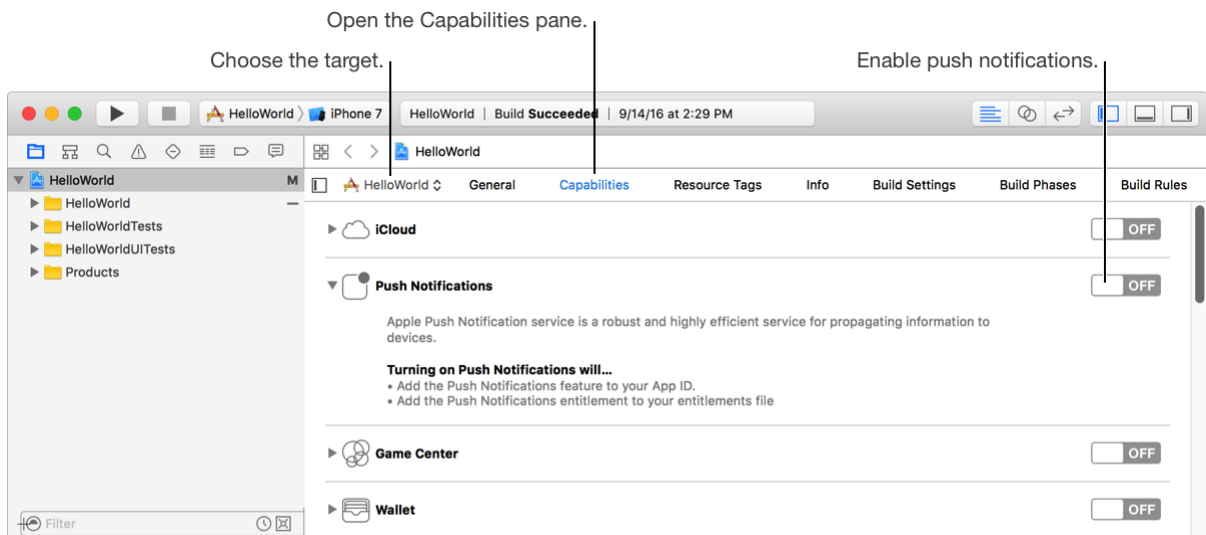


Figura 3.2 - Habilitando *push notifications* em seu projeto Xcode

Fonte: ACCENGAGE, 2015?

As notificações enviadas para os aplicativos são diferentes com base na versão utilizada, se o usuário estiver usando o aplicativo de passageiro, ele receberá alguns tipos de notificações, caso ele utilize o aplicativo de motorista ele receberá outros tipos de *push notifications*.

O usuário que acessa os aplicativos pode receber em algum momento alguma das seguintes notificações:

- Bloqueio: enquanto usuário da plataforma, uma pessoa pode ser bloqueada por algum motivo que possa ser relevante a quem gerencia a mesma. Quando esse caso ocorre, é enviado uma notificação informando ao usuário que ele foi bloqueado e informando quais providências o mesmo deve tomar;

- Chat: ao longo de uma corrida, o passageiro e o motorista podem ser comunicar através de um chat, sempre que uma mensagem é enviada, o destinatário é informado através de um *push* que uma mensagem nova foi recebida;

- Nova corrida: se um motorista estiver habilitado a receber chamadas e definir seu status como disponível, ele será informado sempre que uma pessoa buscar uma corrida através de um *push*, que contém algumas notificações sobre a viagem e exibe também uma tela com mais detalhes, fornecendo ao motorista também a forma de aceitar ou recusar a requisição;

- Atualização no status da corrida: toda corrida possui etapas, que consistem em quando o motorista vai buscar o cliente, encontra o cliente e deixa o mesmo no local de destino. Cada etapa é representada por um status e sempre que o mesmo muda o passageiro é informado através de uma notificação;

3.6 Autenticação

Já foi definido que tanto o aplicativo motorista quanto o passageiro possuem um fluxo de autenticação do usuário, ou seja, o mesmo deve realizar um cadastro e *login* para poder ter acesso as funcionalidades dos aplicativos. Esse *login* é armazenado dentro do app através de uma sessão, que é utilizada para o usuário informar ao servidor que está logado e pode acessar os serviços da plataforma.

Para fazer o gerenciamento da sessão foi optado pelo uso da biblioteca Parse.

3.6.1 Parse

O Parse é um framework que permite aos desenvolvedores criar um *backend* em poucos minutos e fazer o uso do mesmo através de aplicações móveis. O mesmo possui algumas bibliotecas voltadas para aplicações mobile que auxiliam na comunicação com o servidor, gerenciamento da sessão do usuário no app e até envio de arquivos para o servidor. Com o uso dessa biblioteca é possível manter a sessão do usuário ativa mesmo que ele feche o aplicativo. O Parse também é utilizado para auxiliar no envio da imagem de perfil dos usuários para o servidor, onde é enviado a imagem em formato de dados e retornado a url que representa a mesma no servidor.

3.7 Sincronização de dados

Para um aplicativo de mobilidade urbana, é interessante que durante o uso do mesmo, as informações estejam sempre atualizadas. Enquanto passageiro é do interesse do usuário estar acompanhando em tempo real a localização do motorista que está a caminho de busca-lo. Outras informações também devem estar sempre atualizadas tanto para o motorista quanto o passageiro, como a média de avaliação, sempre que uma avaliação é realizada, a média deve ser atualizada.

Para oferecer esse tipo de serviço de modo rápido, eficiente e sem consumir muita internet, é utilizado a ferramenta conhecida como Realtime Database, do Firebase.

3.7.1 Realtime Database (Firebase)

Firebase é uma plataforma voltada para o desenvolvimento de aplicações móveis e web que foi feita pela empresa Google e oferece diversos recursos, como o Realtime Database. O mesmo consiste em uma forma de armazenar e sincronizar informações com um banco de dados na nuvem, em tempo real.

Através do uso dessa ferramenta, toda vez que algum dado é alterado, seja pelo cliente ou servidor, é disparado para todos os usuários que estão observando o dado que o mesmo foi alterado. Se um aplicativo está observando uma corrida em tempo real, toda vez que a mesma sofrer alguma alteração, como o status da corrida, é enviado ao usuário essa nova informação. O principal uso desse recurso é durante o envio das coordenadas do motorista ao servidor, que é feito de 20 em 20 segundos, mantendo o servidor sempre atualizado para com a localização dos motoristas disponíveis para corridas ou com corridas ativas. Essa informação é então utilizada para ser informado ao motorista se há algum pedido de corrida próximo ao mesmo.

Um outro recurso utilizado dessa ferramenta é a persistência de dados, que consiste em um banco de dados local gerenciado pelo próprio Firebase, que possibilita ao usuário uma experiência mais fluida. Esse recurso funciona de forma que sempre que uma informação for recebida, ela é armazenada no app e quando feita outra requisição, essa informação armazenada é exibida até ser atualizada pela resposta referente a nova requisição. Dito isso, quando um usuário está acompanhando uma corrida, se ele fechar o aplicativo e abrir novamente, será exibido os mesmos dados que ele visualizava antes, enquanto é aguardado uma resposta do servidor com as informações atualizadas. Então mesmo que o usuário não

tenha uma boa conexão com a internet no momento de abrir o aplicativo, é exibido as informações que foram armazenadas localmente.

```
final class Firebase_Observe_User {
    static let shared = Firebase_Observe_User()
    var ref: DatabaseReference?

    class func startObserving(completion: @escaping(Firebase_Response) -> Swift.Void) {
        guard let userId = User.current()?.objectId, !userId.isEmpty else { return }
        guard Firebase_Observe_User.ref == nil else { return }
        Firebase_Observe_User.ref = Database.database().reference()
        Firebase_Observe_User.ref?.child("user").child(userId).observe(.value, with { (snapshot) in
            ...
        })
    }

    class func stopObserving() {
        guard let userId = User.current()?.objectId, !userId.isEmpty else { return }
        guard Firebase_Observe_User.ref != nil else { return }
        Firebase_Observe_User.ref?.child("user").child(userId).removeAllObservers()
        Firebase_Observe_User.ref = nil
    }
}
```

O exemplo acima ilustra como é observado as informações de um usuário utilizando o Firebase Realtime Database. A classe “Firebase_Observe_User” é responsável por iniciar o evento de observar um usuário pelo seu “objectId” que consiste em sua chave primária, utilizada para identificar o mesmo.

A função “startObserving” tem como objetivo iniciar o evento de observar o usuário no banco de dados, para isso é verificado qual o objectId do usuário e se o mesmo não é vazio, para garantir que realmente exista um usuário para ser observado. Após isso é verificado se a classe já possui uma referência ao banco de dados, para evitar que a mesma tente iniciar o evento mais de uma vez para o usuário, que não é permitido pela biblioteca. Passando as verificações, é atribuído a classe uma referência do banco de dados e então é iniciado o evento de observar o usuário.

A função “stopObserving” é responsável por encerrar a conexão com o banco de dados, finalizando o evento de observar o usuário. Para isso é verificado novamente o objectId do usuário para garantir que ainda há um usuário em sessão. Após isso, é verificado se a classe ainda possui uma referência do banco de dados, pois se a mesma não possuir significa que o evento já foi encerrado e as linhas de código seguintes não precisam ser executadas. Ao satisfazer todas as verificações, é encerrado o evento de observar o usuário e removido a referência ao banco de dados.

Dentro do escopo da classe “Firebase_Observe_User” é possível notar alguns padrões utilizados, como o Singleton, através da variável “shared” em que é definido uma instância global da classe, não sendo necessário instanciar a mesma durante seu uso em outras partes do projeto. O Singleton consiste em um padrão de projeto que garante que uma classe irá possuir apenas uma instância global dentro do projeto, de modo que possa ser utilizada em qualquer arquivo do mesmo.

Também vale ressaltar a condição “guard” utilizada, que consistem em uma verificação em que se a condição não for válida, qualquer linha de código abaixo da mesma não será executada. É uma verificação muito utilizada em projetos iOS para desembrulhar valores opcionais de um modo seguro, evitando que a aplicação quebre.

3.8 Mapa

É do interesse da plataforma disponibilizar para os usuários um mapa dentro dos aplicativos. Como passageiro, o usuário pode interagir com o mapa de modo a definir qual sua rota de interesse, podendo ter uma pré-visualização da mesma antes mesmo de requisitar um motorista e também acompanhar o mesmo antes e durante a corrida. Do ponto de vista do motorista é importante se ter um mapa disponível para visualizar onde encontrará o passageiro, saber por onde deve navegar e onde será o ponto de destino.

Em termos de bibliotecas existem diferentes opções disponíveis de mapas que podem ser usados em uma aplicação iOS, como o próprio mapa nativo da Apple, o Google Maps, MapBox e outros. Para o desenvolvimento do projeto foi optado pelo uso do Google Maps, que é a biblioteca mais utilizada para aplicativos de mobilidade urbana.

3.8.1 Google Maps

Com a SK de mapas do google para iOS (Maps SDK), o desenvolvedor pode facilmente adicionar um mapa, que faz uso da base de dados do Google maps, ao seu aplicativo. Essa biblioteca já tem como disponível os seguintes recursos:

- Gerenciamento automático de acesso aos servidores do Google Maps;
- Apresentação do mapa;
- Reconhecimento de gestos e interações no mapa, como clicar e arrastar;

- Adicionar pinos e marcadores ao mapa;
- Traçar rotas. (Google, 2019?)

Algumas dessas funcionalidades são feitas através de requisições ao servidor do Google Maps, que por sua vez passam por uma etapa de autenticação, para que a plataforma Google tenha controle na quantidade de acessos e usos de seus retornos. Essa identificação é feita através de uma chave de acesso que é gerada no site da Google, e então enviada junto as requisições feitas.

O uso dessa ferramenta permite ao passageiro escolher os pontos de origem e/ou destino de uma corrida através da navegação pelo mapa, centralizando o mesmo na posição deseje.

3.9 Corrida

O fluxo de corrida é o principal dentro dos aplicativos de mobilidade urbana e é composto por todas as etapas dentro do processo de iniciar, acompanhar e finalizar uma corrida. Algumas bibliotecas são utilizadas para compor esse fluxo, entre elas o Places SDK e Directions.

3.9.1 Places SDK para iOS

A SDK Places para iOS permite ao desenvolvedor adicionar a seu projeto o recurso de busca por endereços e locais baseados em contextos, palavras e coordenadas de uma localização. Este projeto fez uso dessa ferramenta para que o usuário possa definir uma rota de viagem buscando endereços por nome de ruas, bairros, estabelecimentos e outros. Permitindo que um passageiro possa de forma fácil definir os pontos de origem e destino de sua viagem. (GOOGLE, 2019?)

Esse recurso funciona de modo que ao digitar um texto para buscar um endereço, é feita uma requisição a API da Google, que irá retornar um vetor de endereços correspondentes ao texto buscado. Esses endereços podem possuir campos como rua, bairro, número, cidade, estado, CEP, país e outros. Os endereços são então exibidos ao usuário em formato de uma lista, com as informações disponíveis de cada endereço, permitindo ao passageiro escolher um deles para prosseguir e definir sua rota.

3.9.2 Directions API

A Directions API é um serviço utilizado para definir rotas entre duas localizações, através de uma requisição. Essas rotas podem ser definidas para diferentes tipos de transporte, carro, moto e caminhada. Os pontos usados como referência devem ser informados através de coordenadas ou textos.

A rota retornada pelo serviço é composta por informações como distância e tempo estimado, que são informados ao passageiro enquanto o mesmo aguarda pela chegada do motorista ao ponto de início de uma viagem. Também está contido na rota o conjunto de pontos que é utilizado para desenhar a mesma no mapa, que é utilizado tanto no aplicativo passageiro quanto no aplicativo motorista, para ambos acompanharem a corrida pelo mapa.

3.9.3 Corrida no app passageiro

O fluxo de corrida no app passageiro é composto pelas seguintes etapas:

- Definição da rota: o usuário deve começar definindo sua rota de viagem, para isso ele deve definir o ponto de partida, que pode ser a localização atual ou então definido através de busca por texto ou interação com o mapa, onde o mesmo escolhe um ponto e seleciona um botão para confirmar a origem. A escolha do ponto de destino é semelhante à origem, tendo disponível as mesmas opções;

- Escolha da categoria de viagem: tendo sido definida a rota de viagem, uma requisição é feita ao servidor para listar as opções de corrida disponíveis ao passageiro de acordo com os pontos de origem e destino escolhidos. É então desenhado no mapa a rota completa e apresentado a lista de opções de viagem, cada uma com seu valor. Caso não tenha nenhum motorista atuando no momento, é retornado um erro e apresentado um alerta ao passageiro, informando que não há motoristas disponíveis no momento;

- Acompanhamento da viagem: ao escolher uma categoria e apertar o botão de busca por motoristas, é exibido uma tela de busca enquanto algum motorista aceita a corrida. Caso algum motorista aceite, essa tela é dispensada e o mapa é exibido novamente já informando ao passageiro o tempo estimado de chegada do motorista e algumas informações sobre o mesmo (nome, foto e avaliação média), também são fornecidas informações sobre o veículo (placa, modelo, marca e cor). Também é disponibilizado ao passageiro as opções de entrar em contato com o motorista via chat ou cancelar a corrida. O passageiro também pode

acompanhar localização em tempo real do motorista (através do Firebase Realtime Database) e é desenhado a rota entre o ponto de partida da corrida e a localização do motorista. A medida que a corrida muda de status, a tela do mapa também atualiza, informando o passageiro quando o motorista chega ao ponto de encontro, quando a corrida foi inicializada e quando ele chega ao ponto de destino sendo finalizada a corrida;

- Avaliação do motorista: ao ser finalizada a corrida, é exibida uma tela em que o passageiro tem a opção de avaliar o motorista. Na tela são apresentadas informações referentes a corrida como valor e forma de pagamento, informações sobre o motorista (imagem e nome), nesse momento a se forma de pagamento escolhia for dinheiro, o cliente deve acertar o pagamento junto ao motorista, caso o método de pagamento escolhido for o cartão, o próprio sistema se encarrega de cobrar o valor. A avaliação é feita através de uma nota entre 1 e 5, sendo possível adicionar também um comentário a avaliação. Também fica disponível um botão de relatar problema, que redireciona o usuário para uma tela de contato com o suporte da plataforma, para reclamações ou sugestões.

3.9.4 Corrida no app motorista

O fluxo de corrida no app motorista é composto pelas seguintes etapas:

- Solicitação de corrida: quando habilitado a receber corridas, caso algum passageiro requisite um motorista, é exibido uma notificação para o motorista juntamente com uma tela contendo as informações do passageiro (nome, foto e avaliação média) e as informações referentes a corrida (origem, destino e distância). Então o motorista pode optar por aceitar ou recusar a corrida;

- Corrida aceita: caso o motorista opte por aceitar a corrida, a tela de informações é dispensada e o mapa é exibido novamente, contendo as informações referentes ao ponto de encontro com o passageiro, as informações do mesmo, um botão para entrar em contrato como passageiro, um para cancelar a corrida, informar chegada e os botões para abrir os aplicativos de navegação Waze e Google Maps, que se o usuário tiver instalado em seu celular os aplicativos respectivos, o mesmo é redirecionado ao app ao selecionar seu botão. Se o motorista não tiver instalado o aplicativo referente ao botão de navegação pressionado, é informado ao mesmo que deve instalar o app em questão;

- Informar chegada: ao informar a chegada, o botão “Informar chegada” muda seu título para “Iniciar corrida” e o passageiro é informado da chegada através de uma notificação;

- Iniciar corrida: ao pressionar o botão de iniciar corrida, a tela do aplicativo motorista é atualizada, removendo a opção de cancelar corrida, uma vez que o passageiro já se encontra no veículo, e o botão de iniciar corrida dá lugar ao “Finalizar viagem”. Ao mesmo tempo o mapa é atualizado, mostrando ao motorista agora o ponto de destino e desenhada a rota entre sua localização e o mesmo;

- Corrida finalizada: ao finalizar a corrida, é exibida uma tela contendo o valor final da mesma, informações do usuário e método de pagamento. Nesse momento caso a forma de pagamento seja dinheiro, o motorista deve se encarregar de recolher o valor da corrida, caso o pagamento seja via cartão, a própria plataforma cobra o valor direto do passageiro. É disponibilizado também ao motorista a opção de avaliar o passageiro com uma nota entre 1 e 5 e adicionar um comentário a avaliação se for do interesse do motorista. Também é possível relatar um problema junto à plataforma, através de um botão que redireciona o motorista a uma tela de contato com o suporte da empresa.

3.10 Chat

Durante o processo de corrida, tanto o passageiro quanto o motorista podem entrar em contato um com o outro através de um chat, enviando mensagens. Esse recurso é disponibilizado para possibilitar uma comunicação entre ambos, facilitando o processo de encontro ou alertando caso haja algum engano, como outra pessoa entrar no veículo do motorista.

Para desenvolver o chat, foi optado pelo uso do Firebase Realtime Data, para que ambos os aplicativos motorista e passageiro estejam sempre atualizados com relação a conversa em tempo real. Sempre que uma mensagem é enviada, tanto o remetente quanto o destinatário são informados e atualizados em um curto espaço de tempo.

3.11 Menu

O outro fluxo disponível para o usuário durante o uso dos aplicativos motorista e passageiro é o menu lateral. Através do menu o usuário tem acesso a outros módulos disponíveis no app que têm como função complementar a experiência do usuário. O passageiro tem acesso a informações como perfil, histórico de suas viagens, opções de pagamento e suporte. O motorista pode através do menu acessar funcionalidades como perfil, histórico, financeiro, relatórios e suporte.

3.11.1 Menu lateral passageiro

O menu lateral do aplicativo passageiro é composto pelos seguintes módulos:

- Perfil: através do perfil o passageiro pode acessar suas informações cadastradas como a sua foto, nome, email e telefone. Ao selecionar algum desses campos o mesmo pode atualizar as suas informações;

- Histórico: o módulo de histórico exhibe ao passageiro a lista de viagens que o mesmo já realizou, contendo algumas informações de cada viagem como o valor, avaliação dada ao motorista (caso tenha sido avaliado), origem e destino. Ao selecionar uma viagem é exibido uma tela contendo mais informações sobre a mesma. A tela de detalhes apresenta uma imagem de um mapa com o desenho da rota realizada na viagem, as informações sobre o motorista (foto, nome a avaliação média), as informações sobre o veículo (marca, modelo e placa), valor da corrida, detalhes da corrida (origem, destino, dia, horário, duração e distância) e caso o passageiro tenha avaliado o motorista, é exibido a nota, do contrário é informado ao que o mesmo não avaliou o motorista. Também é disponibilizado o recurso “Ajuda” em que são listadas perguntas frequentes e exibido um botão para entrar em contato com o suporte. Por fim o passageiro pode também visualizar os dados referentes ao pagamento da corrida, onde são exibidos o valor da viagem, taxa da plataforma e valor total, o passageiro pode também compartilhar o recibo da corrida através de um botão;

- Pagamentos: nesse fluxo o passageiro pode visualizar seus métodos de pagamentos cadastrados, bem como definir um método de pagamento primário, deletar um método e adicionar um novo método de pagamento (cartão de crédito);

- Suporte e ajuda: acessando esse módulo o passageiro tem acesso a uma lista de perguntas frequentes, e um botão de enviar mensagem para o suporte. Caso o passageiro selecione uma das perguntas frequentes, o mesmo é direcionado a uma tela em que é exibido uma resposta para a pergunta a pergunta selecionada, com orientações ao passageiro, e no fim da tela são exibidos dois botões para o passageiro informar se o conteúdo ali apresentado foi útil ou não, para que a plataforma possa ter um feedback dos usuários. Ao selecionar um dos botões um alerta é exibido perguntando ao usuário se ainda tem alguma dúvida e se o mesmo quer entrar em contato com o suporte, neste caso o mesmo é direcionado a tela de contato;

- Compartilhar: através dessa opção o passageiro tem a opção de compartilhar o aplicativo com outras pessoas através do modo que mais lhe for interessante, seja por email, mensagem, Whatsapp ou outros;

- Sair: ao selecionar essa opção o usuário encerra sua sessão e é enviado novamente ao fluxo de login do app;

3.11.2 Menu lateral motorista

O menu lateral do aplicativo motorista é composto pelos seguintes módulos:

- Perfil: pelo perfil, o motorista pode acessar algumas de suas informações definidas durante o processo de cadastro assim como atualizar algumas delas. Através do perfil o motorista tem acesso a seus documentos, cartões, a sua categoria (comum ou vip), email, telefone, data de nascimento, alterar senha, código de indicação e termos de uso. Ao selecionar uma dessas opções, o motorista é redirecionado a uma tela de detalhes da respectiva opção, caso seja uma informação de cadastro, o motorista pode então atualizar a mesma;

- Histórico: o histórico do motorista se assemelha ao do passageiro, onde o mesmo poderá visualizar todas as corridas que o mesmo já realizou através da plataforma, bem como acessar as informações referentes a mesma. No caso do motorista o mesmo irá visualizar as informações sobre o passageiro que requisitou a corrida, sendo possível ver sua imagem, nome e avaliação média. O motorista também terá acesso a avaliação que o mesmo tenha feito ao passageiro, caso ele tenha avaliado, caso contrário é informado que ele optou por não avaliar o passageiro;

- Financeiro: através do fluxo financeiro, o motorista tem acesso ao seu balanço financeiro, onde lhe é informado seu saldo disponível, o saldo devedor a plataforma, o saldo bloqueado pela plataforma e seus dados de conta bancária, que serão utilizados para sacar seu saldo disponível. Nessa tela o motorista pode adicionar ou editar seus dados de conta bancária através um botão, que o redireciona para uma tela que contém um formulário que contém os campos necessários para a criação de uma conta bancária. Tendo uma conta já cadastrada, um botão de saque é habilitado e o mesmo pode requisitar que seu saldo seja transferido para sua conta bancária;

- Relatórios: ao selecionar a opção de relatórios presente no menu lateral, o motorista é redirecionado a uma tela em que o mesmo pode acompanhar seus ganhos diários através de um gráfico, que apresenta os sete dias da semana, no topo do gráfico o motorista tem a opção de alterar a semana exibida, para voltar as semanas anteriores a semana atual, podendo ver todo seu rendimento ao longo do uso da plataforma;

- Suporte e ajuda: esse fluxo é exatamente igual ao suporte e ajuda do aplicativo do passageiro, composto por uma tela de perguntas frequentes, uma tela de detalhes de uma pergunta selecionada e uma tela de contato com o suporte da plataforma;

- Compartilhar: através do botão de compartilhar o motorista pode compartilhar o aplicativo com outras pessoas, fazendo uso da forma de compartilhamento que lhe for mais interessante;

- Sair: ao selecionar a opção de sair do menu lateral, o motorista encerra sua sessão e é direcionado a tela inicial do aplicativo, tendo que realizar login novamente caso queira oferecer corridas;

4 PRODUTO

Será agora apresentado os protótipos desenvolvidos para o presente trabalho, sendo eles compostos por um aplicativo passageiro e outro motorista.

4.1 Aplicativo passageiro

4.1.1 Tela de carregamento

Todo aplicativo ao ser iniciado apresenta uma tela enquanto o mesmo carrega suas informações.



Figura 4.1 – Tela de carregamento

4.1.2 Tela inicial

Ao terminar o carregamento do aplicativo, é apresentada a tela inicial, onde o usuário pode escolher entre realizar cadastro ou login.

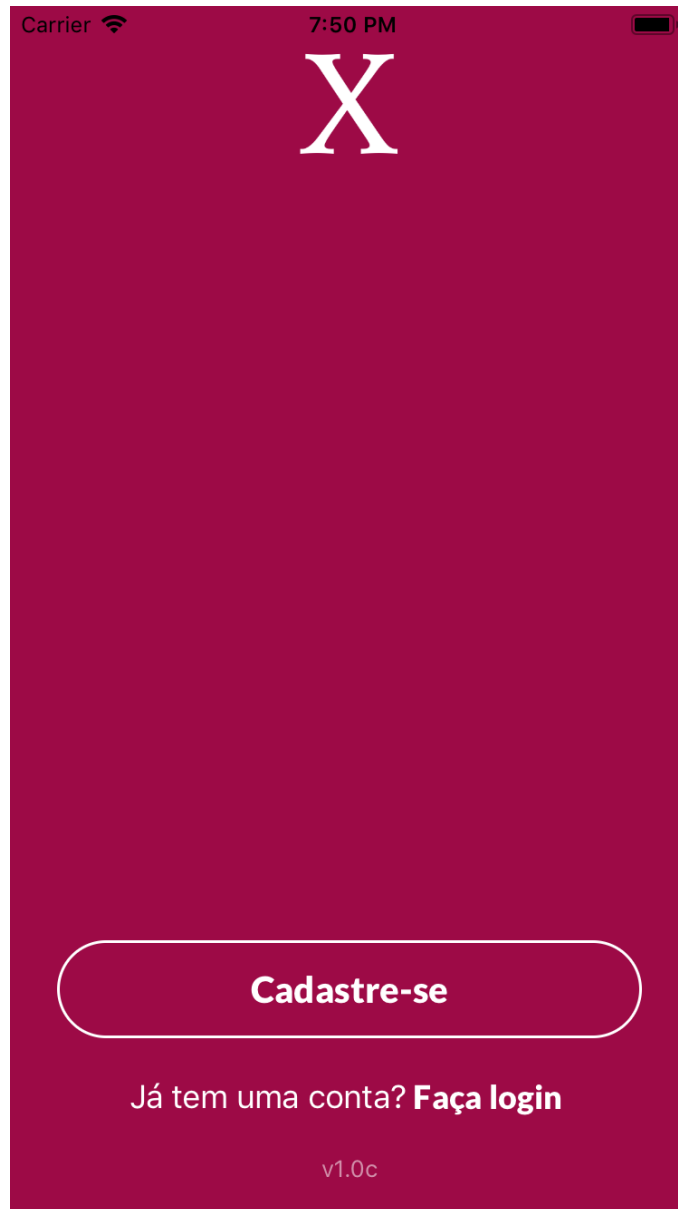


Figura 4.2 – Tela inicial

4.1.3 Login

Caso seja optado por realizar o login, é apresentada a tela de login, que é iniciada com o botão de entrar desabilitado, sendo habilitado após os dados de email e senha serem preenchidos.

Carrier 7:50 PM

<

X

E-mail

Senha

Entrar

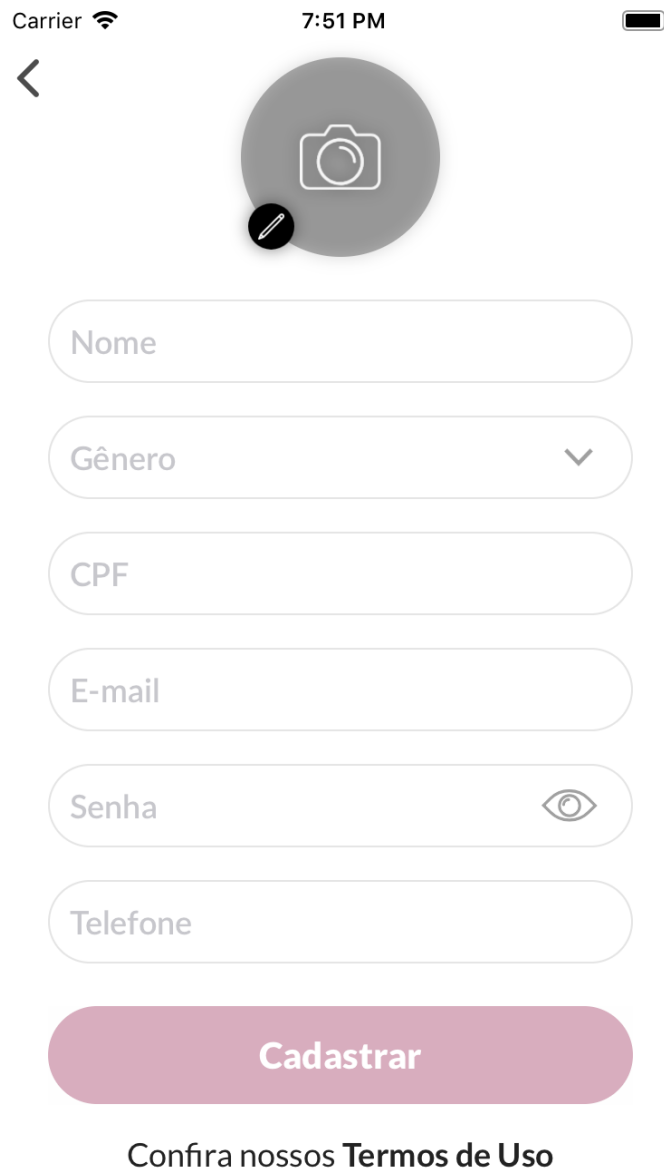
[Esqueci minha senha](#)

Faça seu **cadastro**

Figura 4.3 – Login

4.1.4 Cadastro

Caso selecionada a opção de cadastro é apresentado a tela de cadastro, que é iniciada com o botão de cadastrar desabilitado, que é habilitado após o usuário entrar com dados necessários para o cadastro.





The image shows a mobile registration screen. At the top, the status bar displays 'Carrier' with a Wi-Fi icon, the time '7:51 PM', and a battery icon. Below the status bar is a back arrow on the left and a circular profile picture placeholder in the center containing a camera icon and a small edit icon. The form consists of several input fields: 'Nome', 'Gênero' (with a dropdown arrow), 'CPF', 'E-mail', 'Senha' (with an eye icon for visibility), and 'Telefone'. At the bottom of the form is a large, rounded, pink button labeled 'Cadastrar'. Below the button, the text 'Confira nossos Termos de Uso' is displayed.

Figura 4.4 – Cadastro

4.1.5 Recuperar senha

Caso seja selecionado o botão “Esqueci minha senha”, a tela de recuperar senha é apresentada, com o botão de recuperação desabilitado, sendo habilitado após o email ser informado.



Carrier  7:51 PM 





Informe o email cadastrado

Recuperar

Figura 4.5 – Recuperar senha

4.1.6 Mapa

Após a realização do login ou cadastro, a tela do mapa é apresentada, onde é possível pedir uma corrida após traçar a rota.

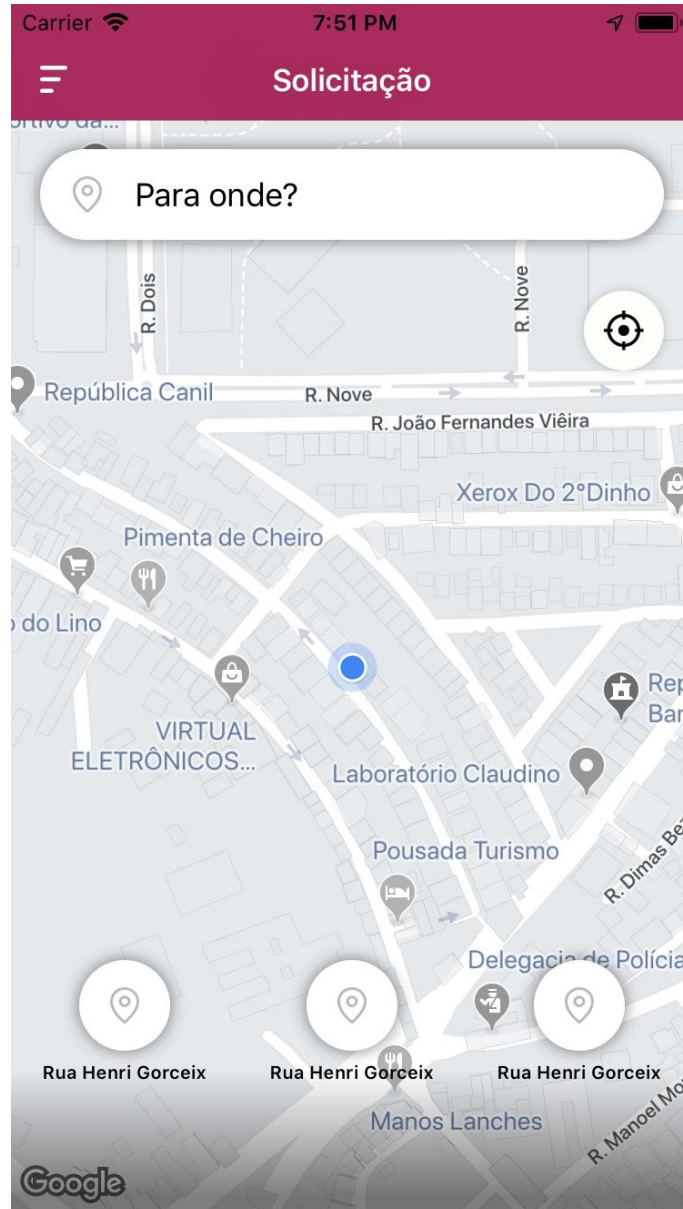


Figura 4.6 – Mapa

4.1.7 Busca por endereço

Ao selecionar o campo “Para onde?”, é apresentado o teclado e a opção de buscar por um endereço através de texto.

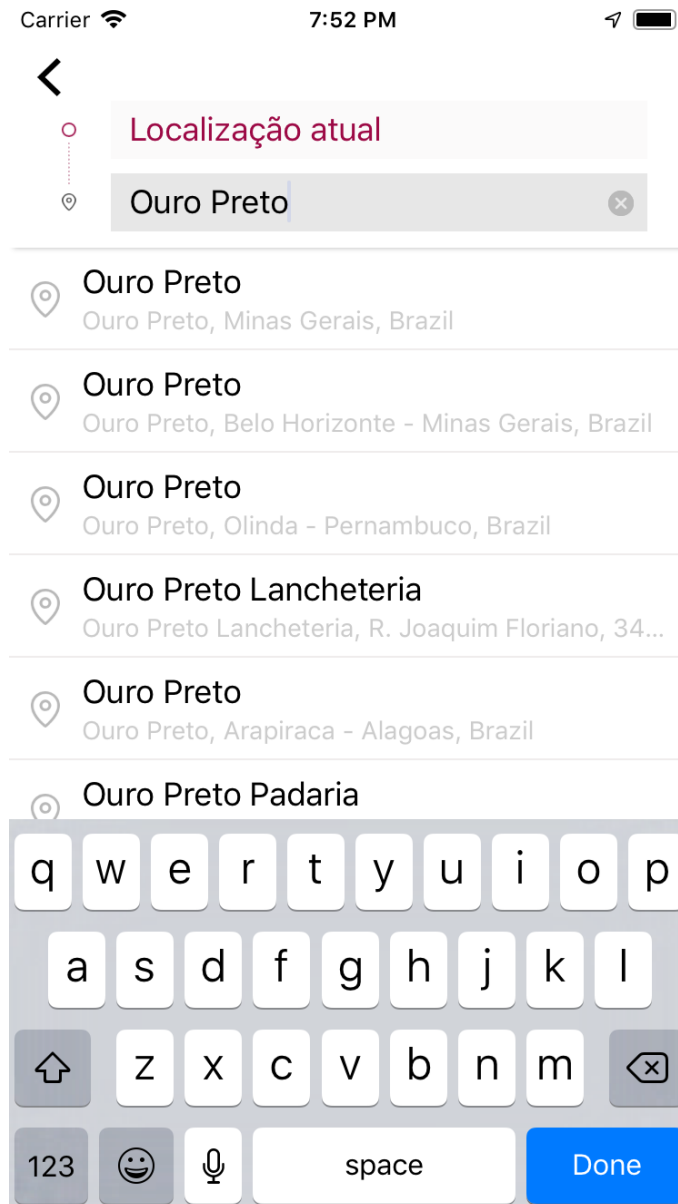


Figura 4.7 – Busca por endereço

4.1.8 Rota definida

Após a definição da rota, a mesma é desenhada no mapa e apresentado ao passageiro quais as categorias disponíveis.

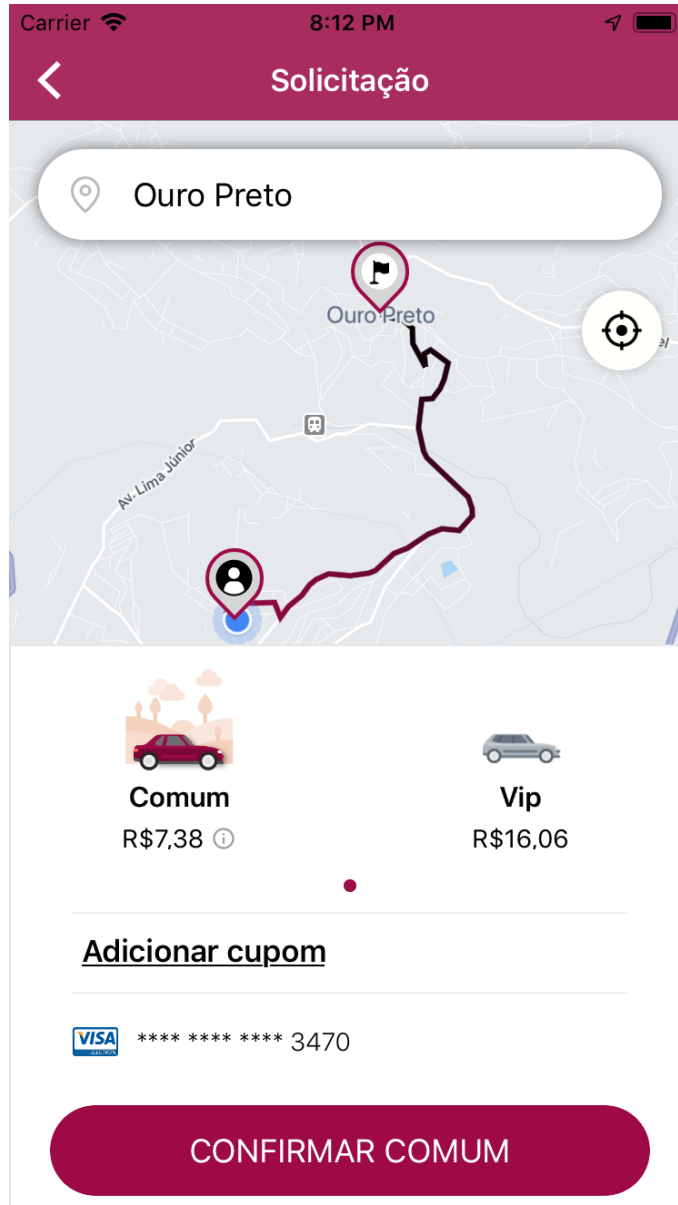


Figura 4.8 – Rota definida

4.1.9 Buscando motorista

Ao selecionar o botão de confirmar, a busca por um motorista é iniciada.



Figura 4.9 – Buscando motorista

4.1.10 Viagem aceita por motorista

Se a busca for aceita por um motorista, é exibido o mapa novamente, apresentando a rota entre a localização do motorista e o ponto de início da corrida.



Figura 4.10 – Viagem aceita por motorista

4.1.11 Corrida iniciada

Quando o motorista chega ao encontro do cliente e o mesmo entra no veículo, a corrida é iniciada e o mapa atualizado.

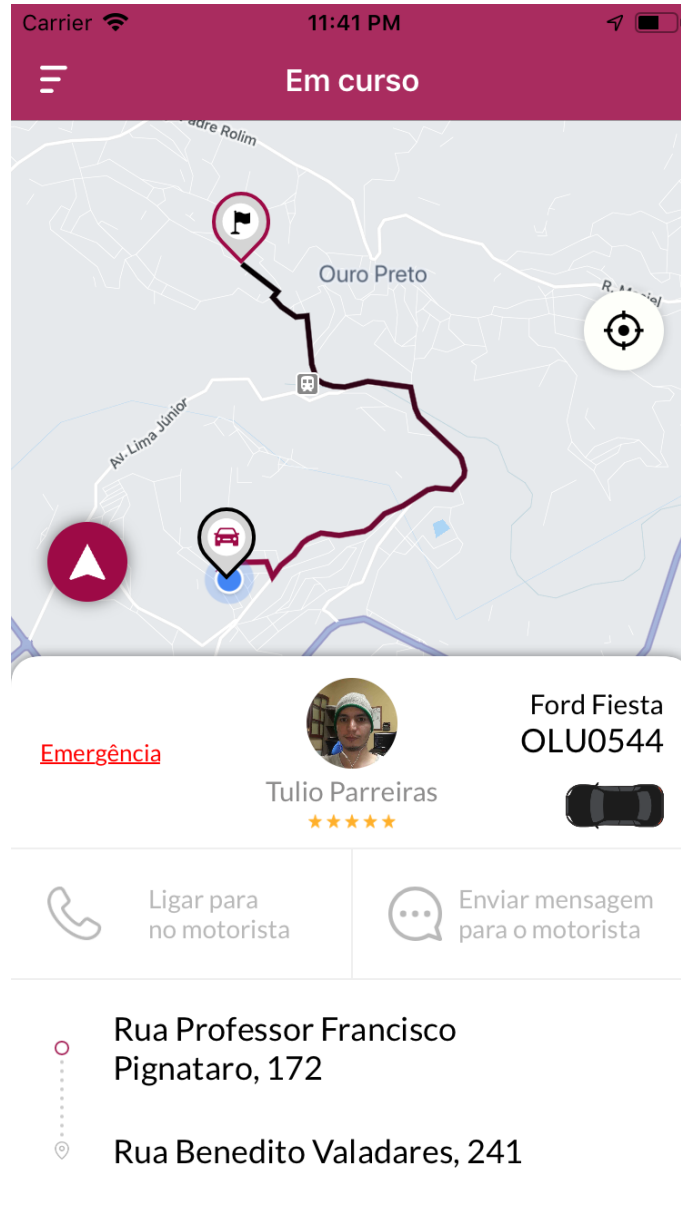


Figura 4.11 – Corrida iniciada

4.1.12 Avaliar o motorista

Ao finalizar a corrida, é exibido a tela de avaliação do motorista para o passageiro.



The screenshot shows a mobile application interface for evaluating a driver. At the top, there is a status bar with 'Carrier', signal strength, Wi-Fi, and battery icons, and a time of 11:42 PM. Below this is a dark red header with the word 'Resumo' and a close button (X). The main content area features a driver's profile: a circular profile picture of a man, the name 'Tulio Parreiras', and the car model 'Ford Fiesta - OLU0544'. To the right, it indicates the payment method as 'PAGAMENTO Cartão'. The fare is displayed as 'Valor R\$6,50'. Below the fare, the text 'Avalie sua viagem:' is followed by five grey stars. A text input field with the placeholder 'Escreva uma mensagem (opcional)' is provided for feedback. At the bottom, there is a large, rounded, light red button labeled 'Avaliar' and a link labeled 'Reportar problema'.

Figura 4.12 – Avaliação do motorista

4.1.13 Menu

Ao selecionar o botão de menu ao lado esquerdo da tela, o menu lateral é apresentado.

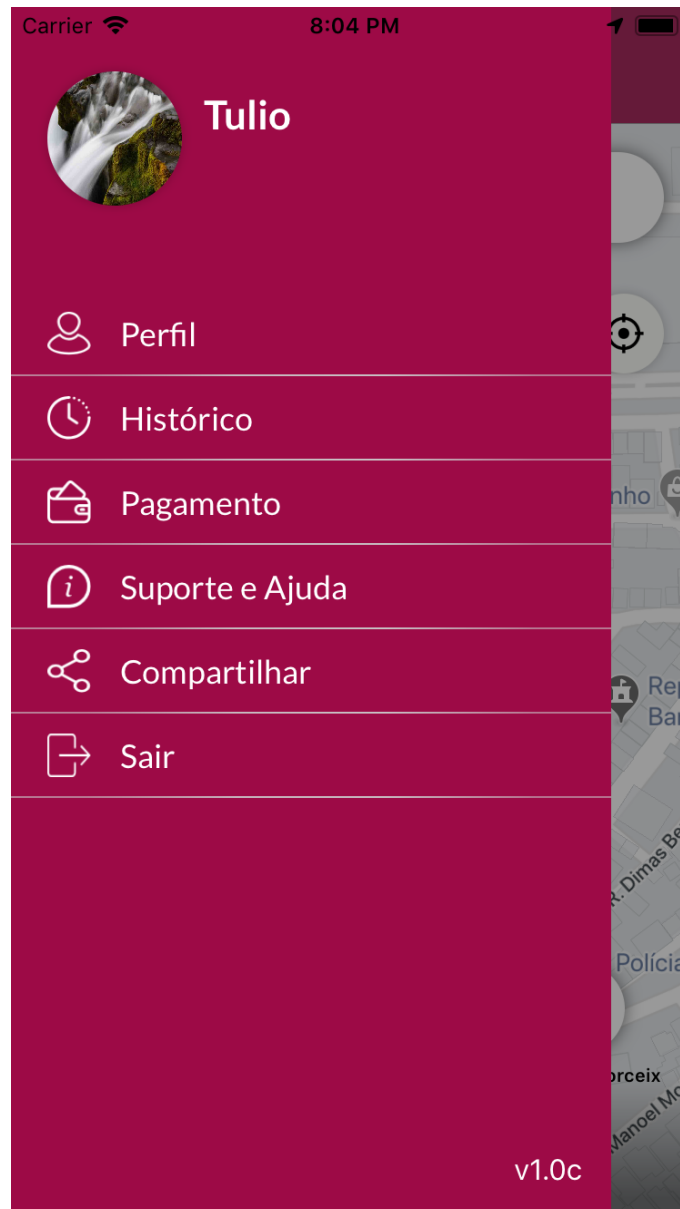


Figura 4.13 – Menu lateral

4.1.14 Perfil

Se selecionado a opção perfil no menu lateral, a tela de perfil é exibida.

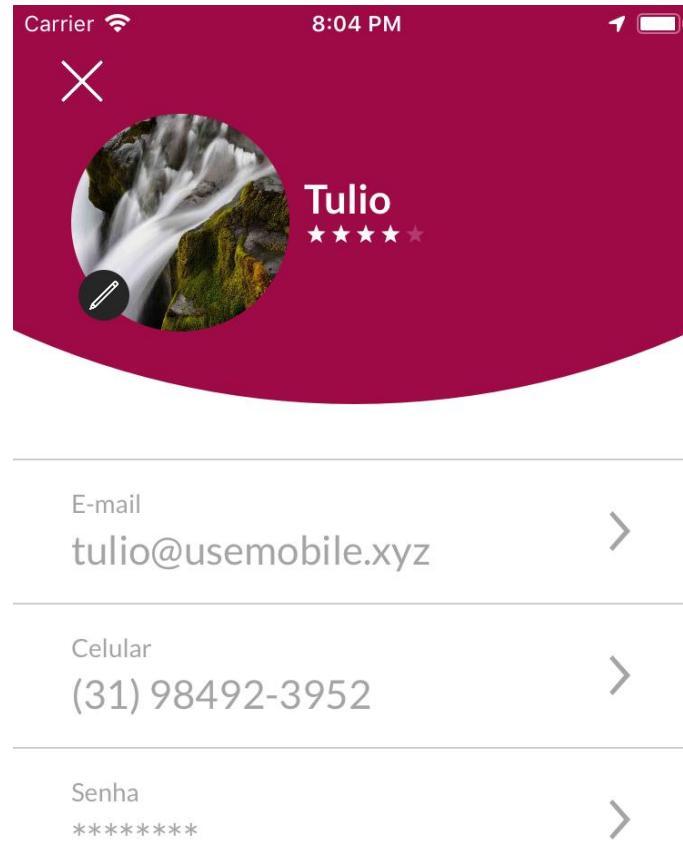


Figura 4.14 – Perfil

4.1.15 Editar e-mail

Ao selecionar o campo e-mail, a tela de editar e-mail é apresentada.

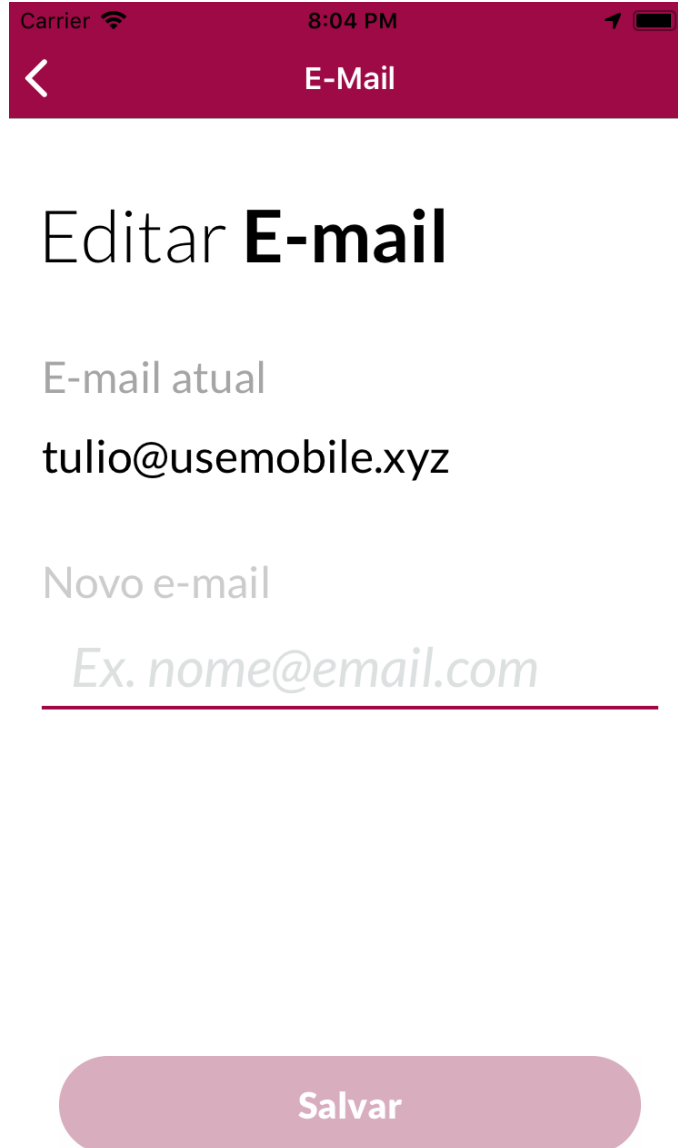


Figura 4.15 – Editar e-mail

4.1.16 Editar celular

Caso selecionado o campo celular, é carregado a tela de editar celular.

Carrier 8:04 PM

< Celular

Editar Celular

Celular atual
(31) 98492-3952

Novo celular
Ex. 31 987654321

Salvar

Figura 4.16 – Editar Celular

4.1.17 Editar senha


Se o campo senha for selecionado, a tela de edição de senha é apresentada.

Carrier 8:04 PM


< Senha

Preencha a sua
Nova senha

Senha atual

Senha atual 

Nova senha

Nova senha 

Salvar

Figura 4.17 – Editar senha

4.1.18 Histórico

Se a opção do histórico for selecionada, a tela de listagem do histórico de corridas é mostrada.

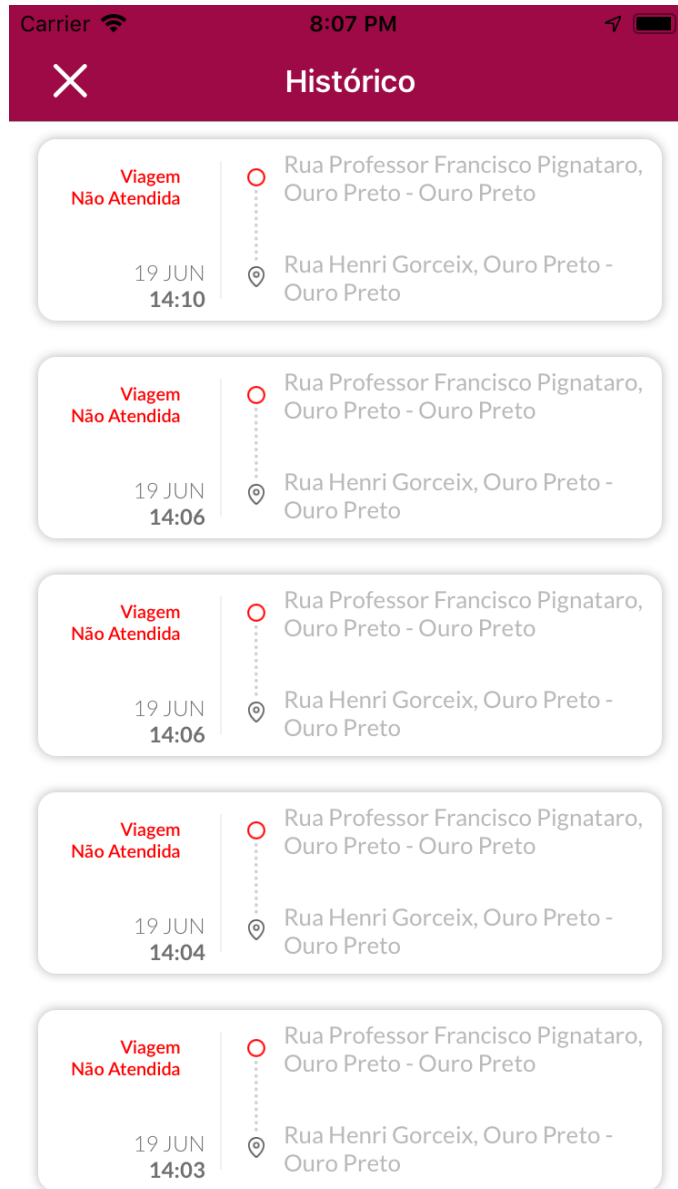


Figura 4.18 - Histórico

4.1.19 Detalhes do histórico – detalhes

Se um histórico for selecionado, é apresentado os detalhes do mesmo.



Figura 4.19 – Detalhes do histórico

4.1.20 Detalhes do histórico - ajuda

Se a aba ajuda for clicada, é apresentado uma lista de opções de ajuda.



Figura 4.20 – Detalhes do histórico (ajuda)

4.1.21 Detalhes do histórico - recibo

Se a aba recibo for selecionada, é apresentado os dados de recibo da corrida.



Figura 4.21 – Detalhes do histórico (recibo)

4.1.22 Pagamento

Caso a opção pagamento do menu lateral seja clicada, a tela de listagem dos métodos de pagamento é carregada.

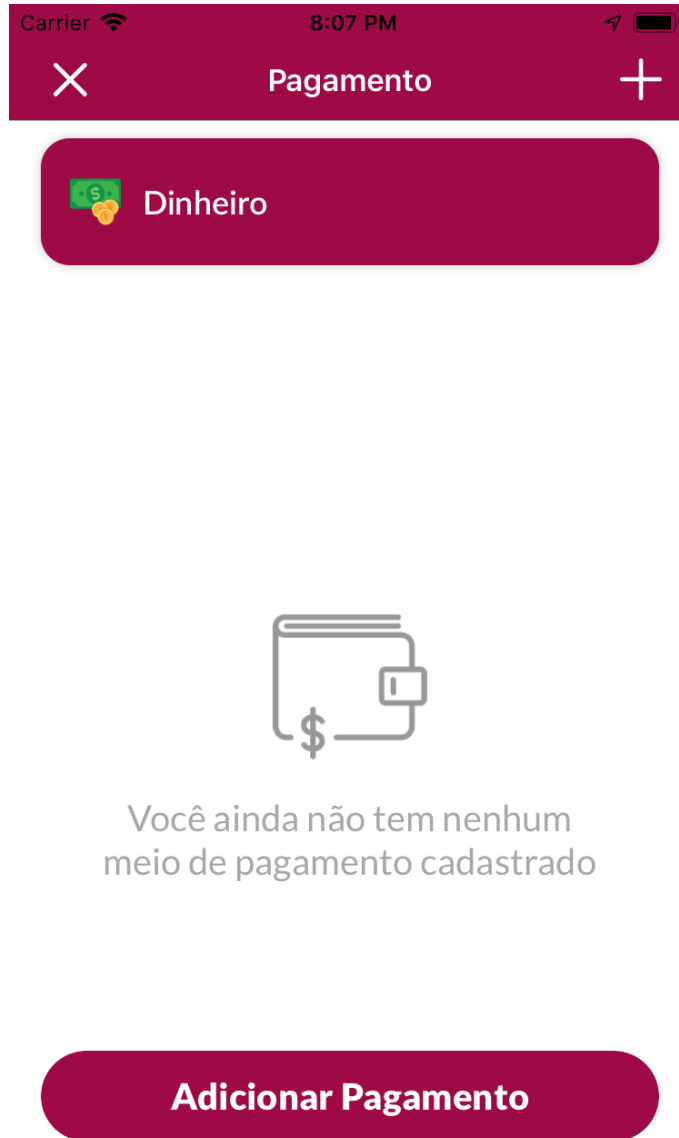


Figura 4.22 - Pagamento

4.1.23 Adicionar cartão

Ao selecionar o botão de adicionar cartão, a tela de cadastrar cartão é apresentada.



The screenshot shows a mobile application interface for adding a card. At the top, there is a status bar with 'Carrier', signal strength, '8:08 PM', and battery level. Below this is a dark red header with a white back arrow and the text 'Pagamento'. The main content area features a dark grey card template with a world map background. The card has a placeholder for a logo in the top left, and labels 'CARD HOLDER' and 'EXPIRES' in the bottom left and right respectively. Below the card template are four input fields: 'Número do cartão' and 'MM/AA' (Month/Year) on the first line, 'Nome do titular' and 'Código' (Code) on the second line, and 'CPF' (Brazilian Tax ID) on the third line. At the bottom of the screen is a large dark red button with the white text 'CONFIRMAR'.

Figura 4.23 – Adicionar cartão

4.1.24 Suporte e ajuda

Se a opção de suporte e ajuda do menu lateral for selecionada, é apresentada a tela de suporte e ajuda.

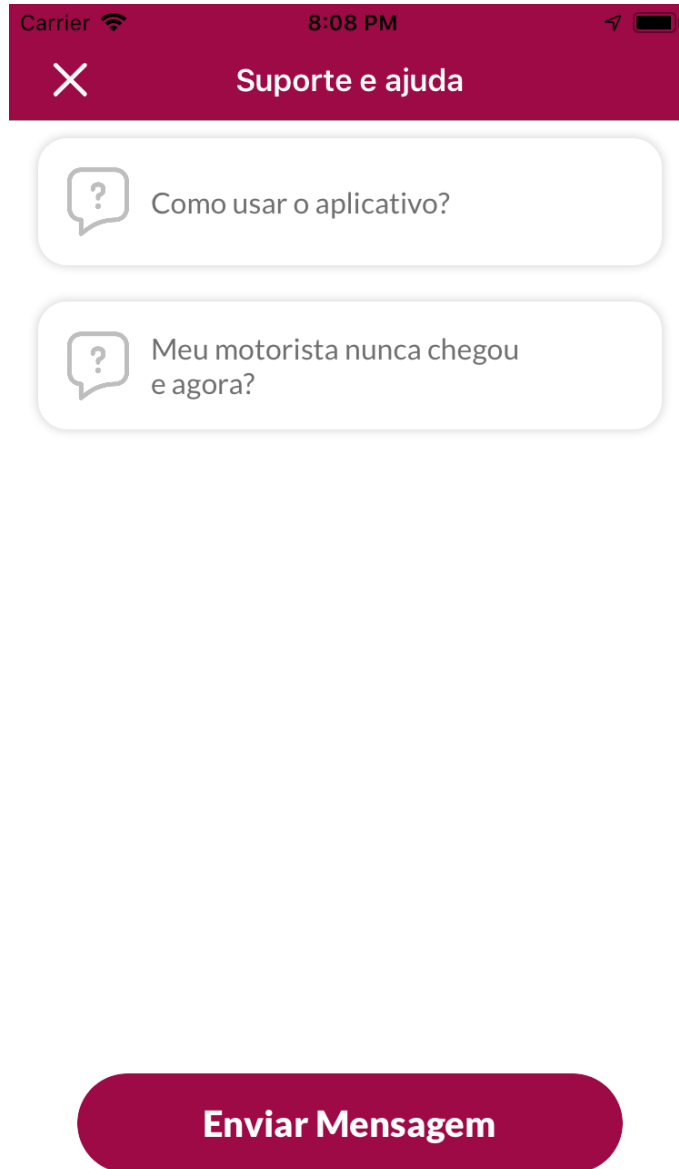


Figura 4.24 – Suporte e ajuda

4.1.25 Suporte e ajuda - detalhes

Caso um item de suporte seja clicado, são apresentados os detalhes do mesmo.

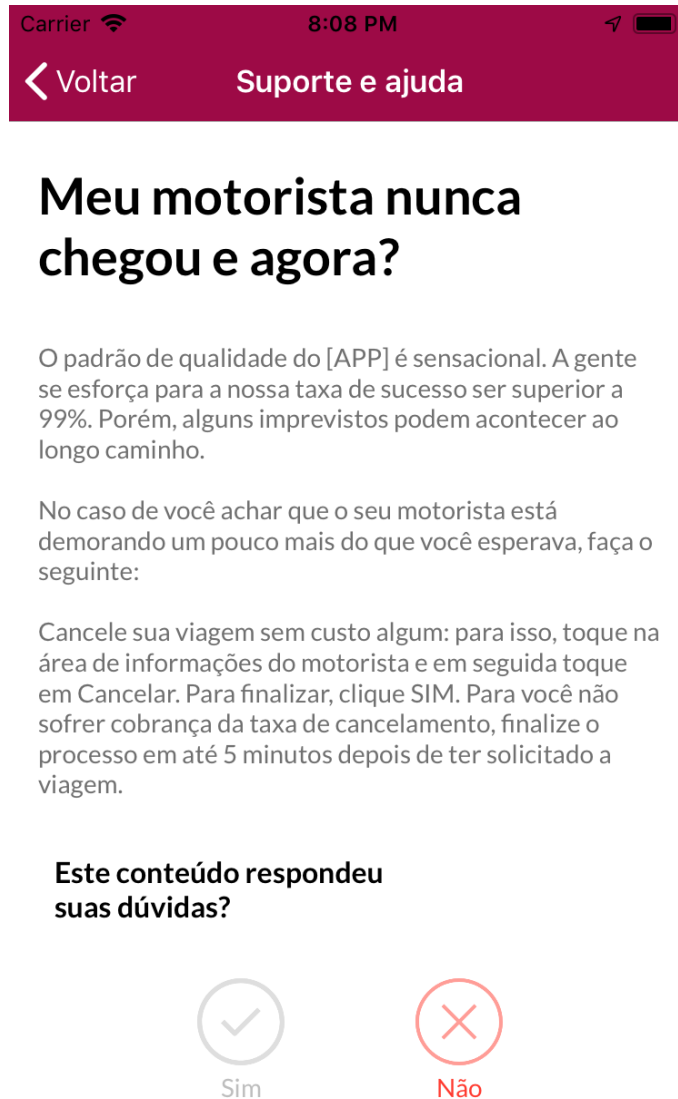
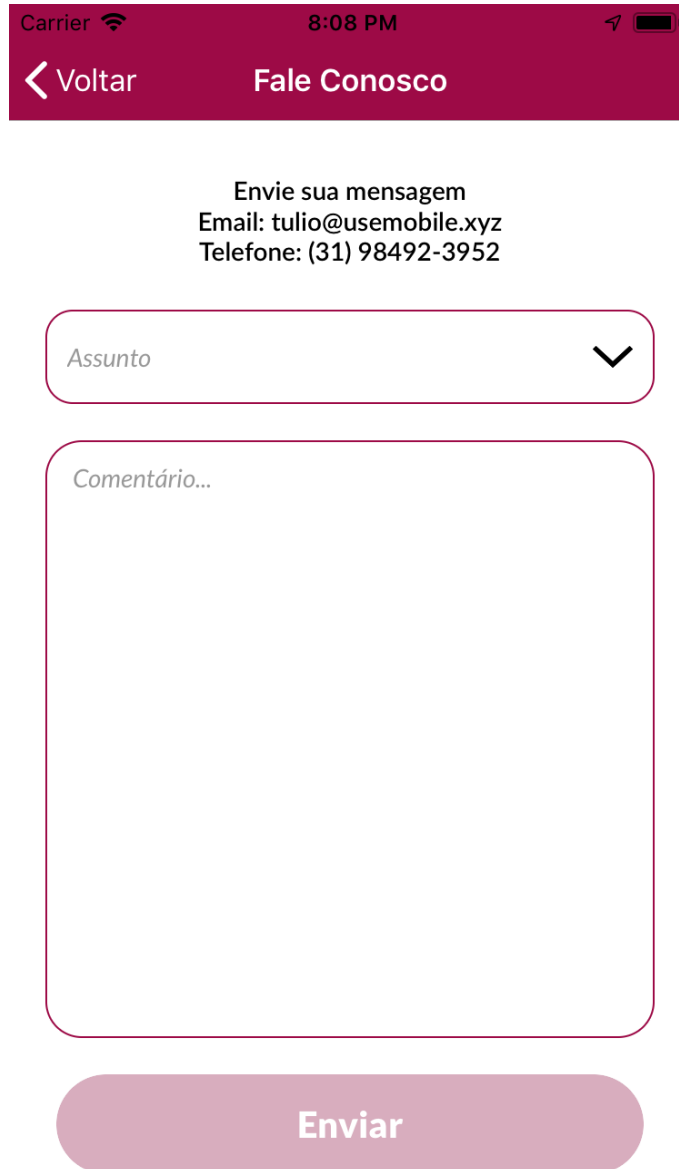


Figura 4.25 – Suporte e ajuda (detalhes)

4.1.26 Fale conosco

Caso seja selecionado o botão de enviar mensagem na tela de suporte, é apresentado ao usuário a tela de contato.



The screenshot shows a mobile application interface for a contact form. At the top, there is a dark red header bar with a white back arrow and the text 'Voltar' on the left, and 'Fale Conosco' in the center. The status bar above shows 'Carrier', signal strength, Wi-Fi, the time '8:08 PM', and battery level. Below the header, the text 'Envie sua mensagem' is followed by 'Email: tulio@usemobile.xyz' and 'Telefone: (31) 98492-3952'. The form consists of a white rounded rectangle with a red border, containing a text input field labeled 'Assunto' with a dropdown arrow on the right, and a larger text area labeled 'Comentário...'. At the bottom of the form is a large, rounded, light red button with the text 'Enviar' in white.

Figura 4.26 – Fale conosco

4.1.27 Sair

Se o usuário selecionar a opção sair do menu lateral, é apresentado um alerta confirmando a intenção do mesmo de sair de sua sessão.



Figura 4.27 – Sair

4.2 Aplicativo motorista

4.2.1 Tela de carregamento

Ao iniciar a aplicação, o motorista é apresentado a uma tela de carregamento, enquanto o aplicativo carrega suas funcionalidades.

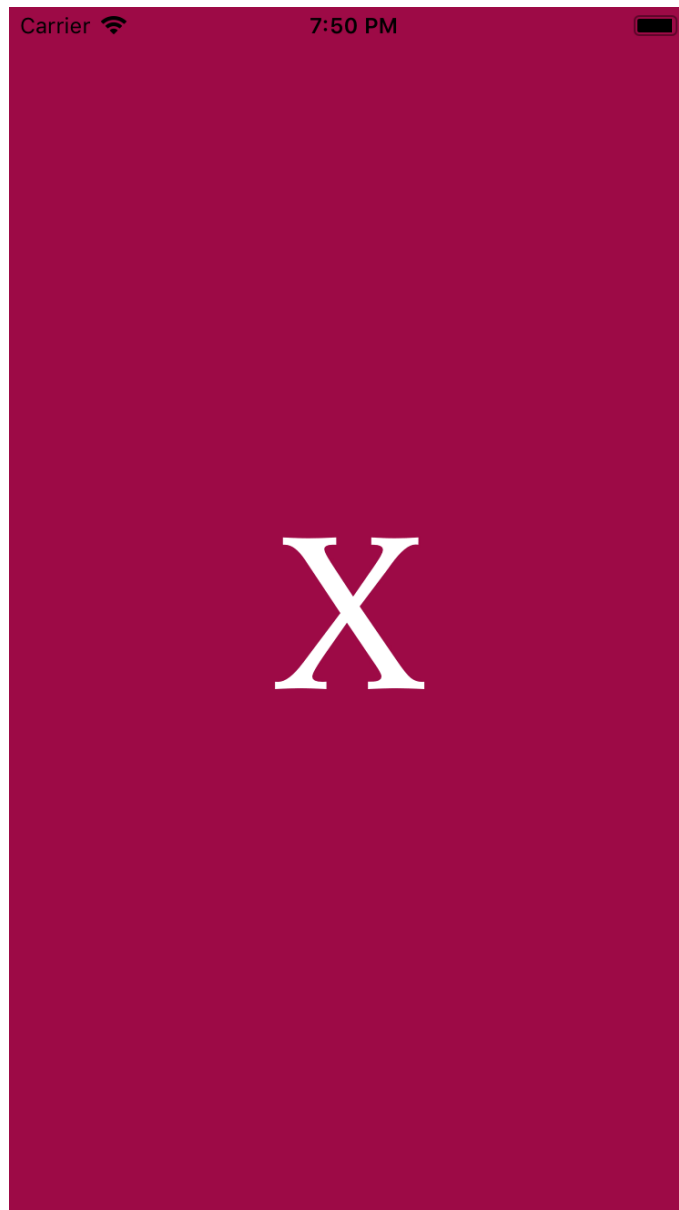


Figura 4.28 – Carregamento (motorista)

4.2.2 Tela inicial

Após o fim do carregando do mesmo, é apresentada a tela inicial do aplicativo, por onde o motorista pode escolher entre realizar o login ou se cadastrar.



Figura 4.29 – Tela inicial (motorista)

4.2.3 Tela de login

Ao selecionar a opção de login, o usuário é direcionado para a tela de login, em que o mesmo tem a opção de iniciar sua sessão informando seu e-mail e senha. O motorista pode também selecionar a opção de “Esqueci minha senha” ou optar pelo cadastro.

Carrier 7:50 PM

<

X

E-mail

Senha

Entrar

[Esqueci minha senha](#)

[Faça seu cadastro](#)

Figura 4.30 – Login (motorista)

4.2.4 Tela de recuperar senha

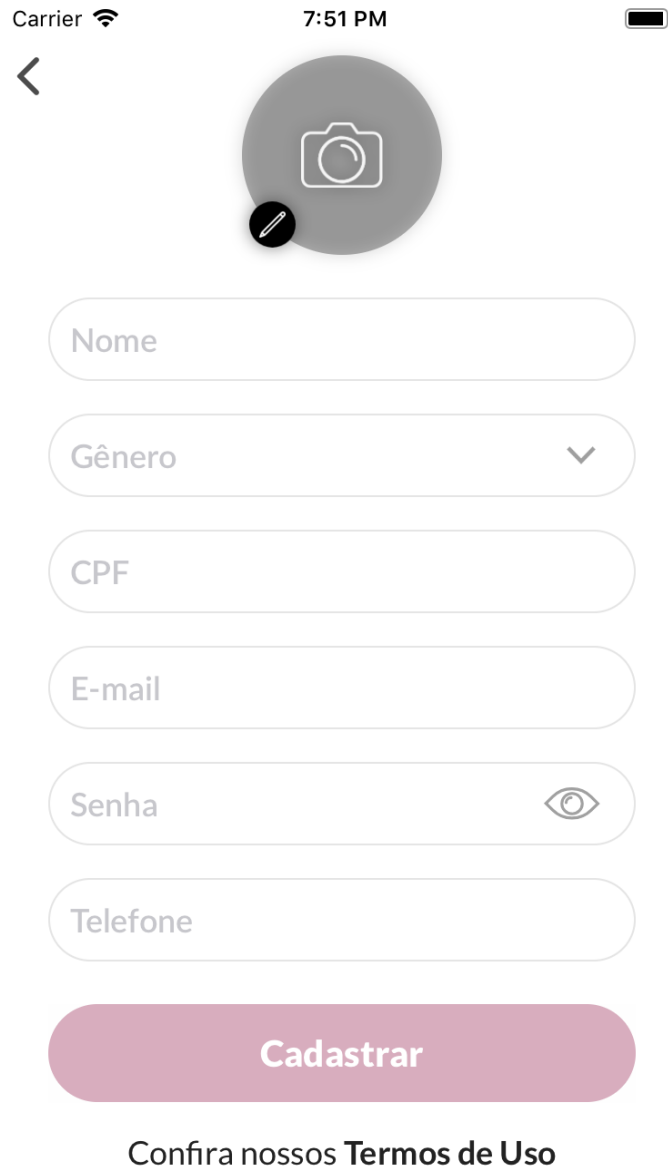
Caso o motorista opte por recuperar sua senha, ele é apresentado a uma tela em que deve informar o e-mail de sua conta e pedir a recuperação.



Figura 4.31 - Recuperar senha (motorista)

4.2.5 Tela de cadastro

Se o motorista selecionar o botão de cadastro, é carregada a tela da primeira etapa do cadastro, em que o mesmo deve entrar com algumas informais pessoais (foto, nome, e-mail, senha, telefone, CPF e gênero).



The image shows a mobile application registration screen. At the top, there is a status bar with 'Carrier', a Wi-Fi icon, the time '7:51 PM', and a battery icon. Below the status bar is a back arrow on the left and a circular profile picture placeholder in the center containing a camera icon and a small edit icon. The registration form consists of several rounded rectangular input fields: 'Nome', 'Gênero' (with a dropdown arrow), 'CPF', 'E-mail', 'Senha' (with an eye icon for visibility), and 'Telefone'. At the bottom of the form is a large, rounded, pink button labeled 'Cadastrar'. Below the button is the text 'Confira nossos Termos de Uso'.

Figura 4.32 – Cadastro (motorista)

4.2.6 Tela de consentimento legal

Após passar pela primeira etapa do cadastro, o motorista é apresentado a tela de consentimento legal, em que o mesmo deve ler os termos e condições da plataforma para ser um parceiro, aceitar e continuar.



Figura 4.33 - Consentimento legal

4.2.7 Tela dados pessoais

Tendo aceito os termos, é apresentada uma tela para que o motorista informe mais alguns dados pessoais (sobrenome, data de nascimento, cidade e código de indicação) e confirme seu nome.

Captura de tela de uma interface de usuário para a coleta de dados pessoais. O cabeçalho mostra o status do sistema (9:41 AM, 100% de bateria) e o título "Dados pessoais" com um botão "Sair". O formulário solicita o preenchimento de:

- Nome
- Sobrenome
- Data de nascimento
- Cidade de atuação
- Código de indicação (opcional)

Um botão "Avançar" está visível na base da tela.

Figura 4.34 - Dados pessoais

4.2.8 Categoria

A próxima etapa do cadastro é definir qual a categoria que o motorista deseja atuar, as opções disponíveis vêm do servidor e mesmo deve optar pela que melhor se adequar ao seu caso.

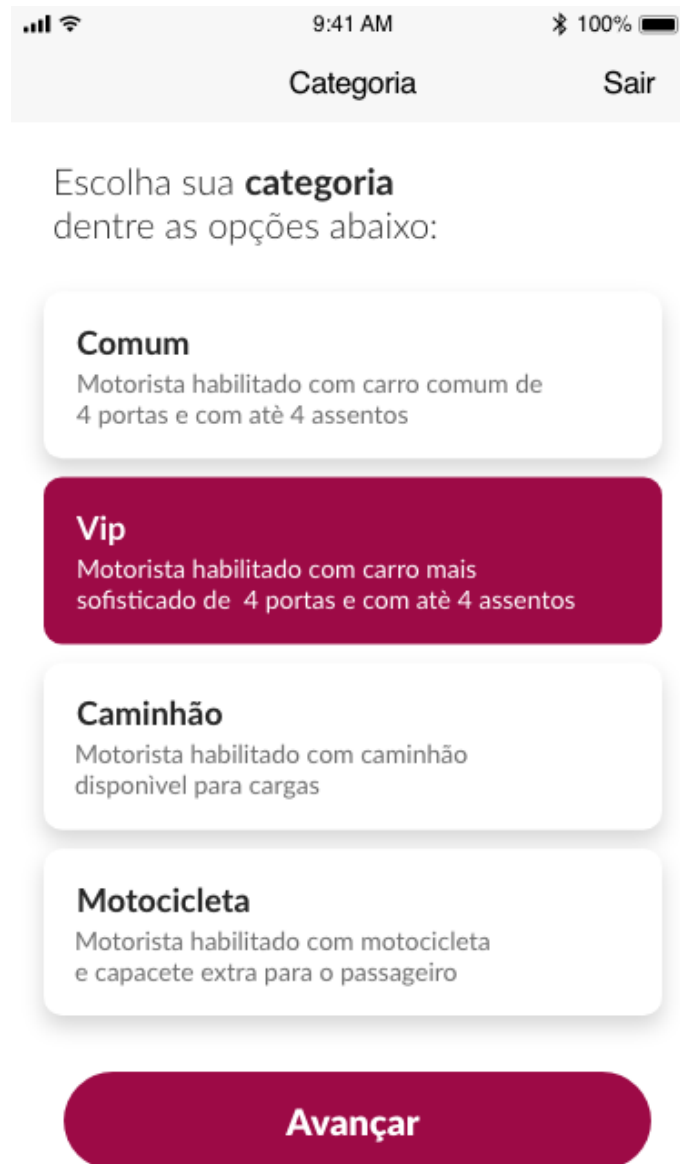


Figura 4.35 - Categoria

4.2.9 Dados do veículo

O motorista deve agora informar os dados sobre o veículo que o mesmo irá utilizar para prestar seu serviço (placa, cor, marca, modelo e ano de fabricação).



The screenshot shows a mobile application interface for entering vehicle information. At the top, there is a status bar with signal strength, Wi-Fi, time (9:41 AM), Bluetooth, and 100% battery. Below the status bar, the title "Dados do veículo" is centered, and a "Sair" button is on the right. The main content area starts with the instruction "Complete com os dados do seu veículo:" followed by "Categoria selecionada: Vip". There are five input fields: "Placa", "Cor do veículo" (with a dropdown arrow), "Marca", "Modelo", and "Ano" (with a dropdown arrow). At the bottom, there is a large, rounded, pink button labeled "Avançar".

Figura 4.36 - Dados do veículo

4.2.10 Habilitação

A última etapa do cadastro consiste em enviar fotos dos documentos do motorista para o sistema. O mesmo deve adicionar uma foto de sua CNH, CRLV e se for do seu interesse atualizar a foto do perfil.



Figura 4.37 - Habilitação

4.2.11 Mapa

Tendo finalizado o cadastro, o motorista é direcionado para a tela do mapa, em que deve aguardar pela verificação de seus documentos antes de ser possível começar a atuar.



Figura 4.38 - Mapa (motorista)

4.2.12 Tela nova corrida

Após o cadastro do motorista ser aprovado, ele pode habilitar as chamadas de corrida. Assim que um passageiro pedir um motorista é exibido uma tela com os dados do passageiro e da corrida, sendo possível aceitar ou recusar a mesma.

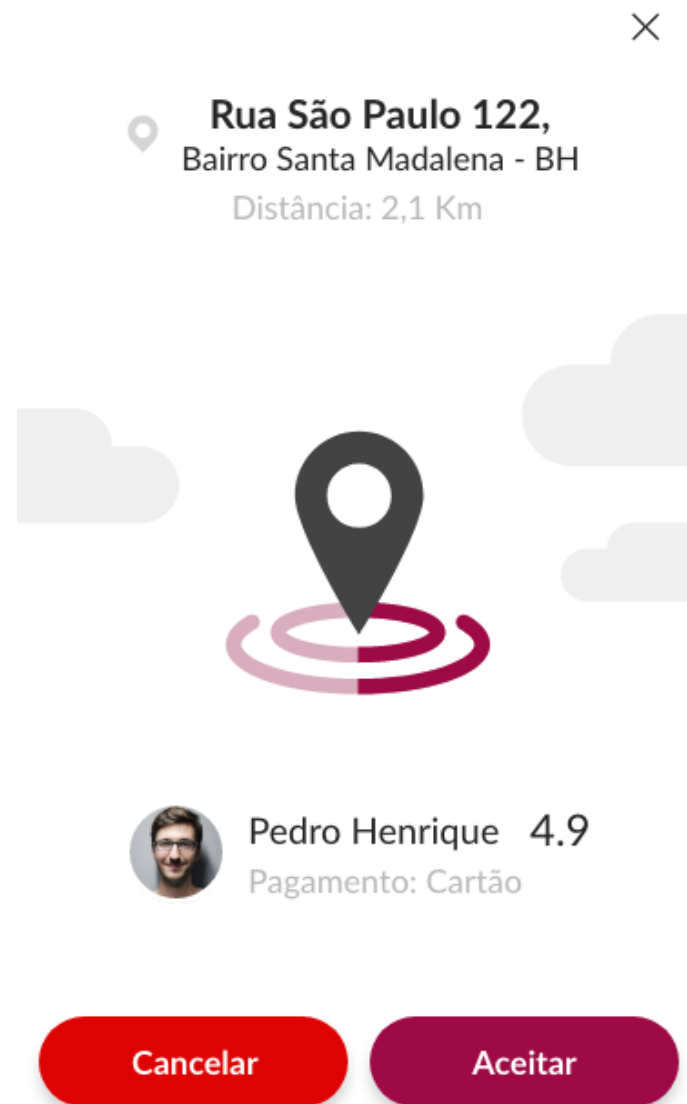


Figura 4.39 - Nova corrida

4.2.13 Corrida aceita

Ao aceitar a solicitação de corrida, a tela de nova corrida é dispensada e o mapa é exibido novamente. Ao mesmo tempo o mapa é atualizado para exibir os detalhes da corrida, que consistem no endereço de origem, dados do passageiro (nome, avaliação média e foto). Também são disponibilizadas as opções de abrir a navegação, entrar em contato com o passageiro, cancelar corrida e iniciar a corrida.

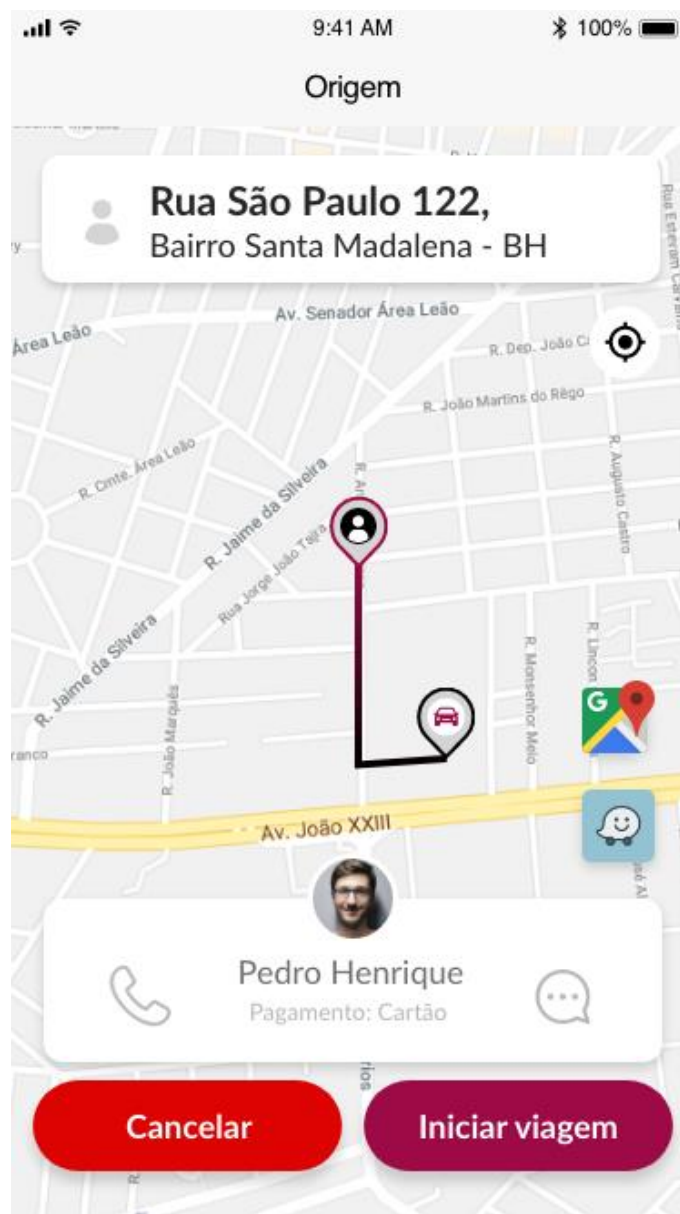


Figura 4.40 - Corrida aceita

4.2.14 Iniciar viagem

Ao iniciar a viagem, o botão de cancelar viagem é escondido, uma vez que não deve ser possível cancelar uma viagem com o passageiro presente no veículo. O botão de iniciar viagem dá lugar ao botão de finalizar viagem e o mapa é atualizado para exibir o destino da viagem.

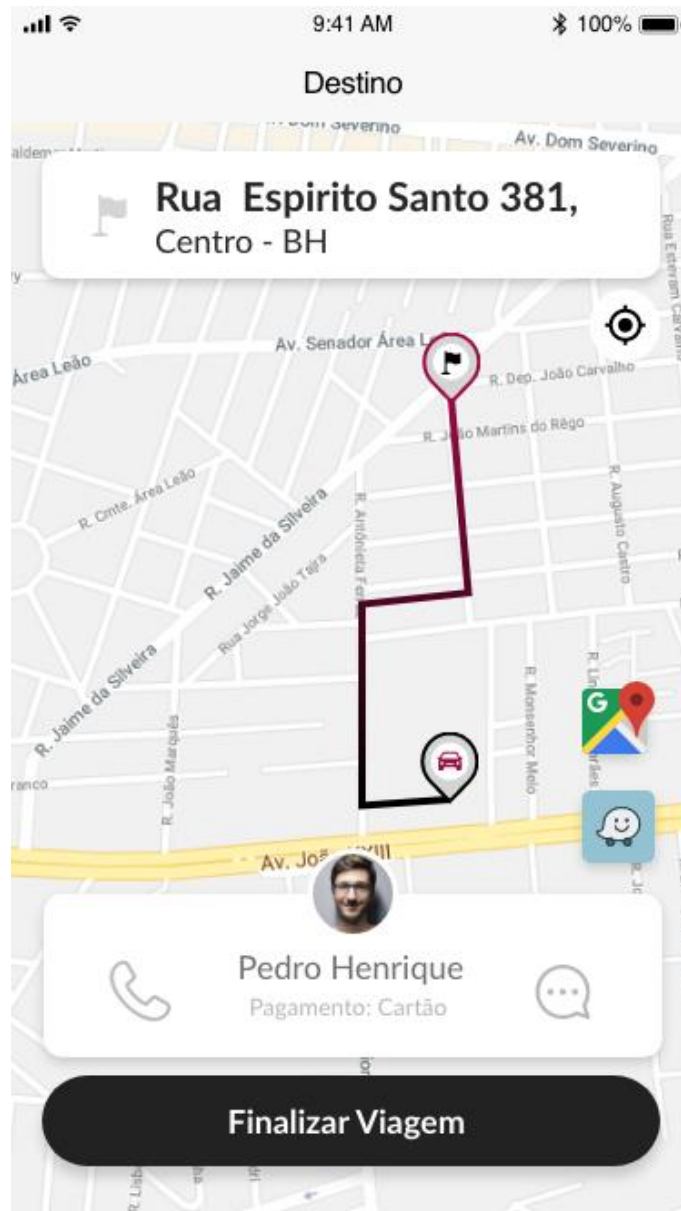


Figura 4.41 - Iniciar viagem

4.2.15 Avaliar passageiro

Quando finalizado a viagem, é exibido para o motorista a tela de avaliação do passageiro, sendo possível o motorista atribuir uma nota entre 1 e 5, bem como adicionar um comentário caso seja de seu interesse.

The screenshot shows a mobile application interface for a completed ride. At the top, the status is 'Finalizado'. Below this, the passenger's profile is shown with a circular photo of Pedro Henrique. To the right, the payment method is listed as 'PAGAMENTO Dinheiro'. The fare amount is displayed as 'VALOR R\$ 74,00'. The main section is titled 'Avalie sua viagem:' and features a 5-star rating system with four yellow stars and one grey star. Below the stars is a text input field with the placeholder 'Escreva uma mensagem (opcional)'. At the bottom, there is a large red button labeled 'Avaliar' and a smaller link labeled 'Reportar problema'.

Figura 4.42 - Avaliar passageiro

4.2.16 Menu lateral

Caso o motorista selecione o botão do menu, é exibido um menu lateral com as seguintes opções: perfil, histórico, financeiro, relatórios, suporte e ajuda, compartilhar e sair.

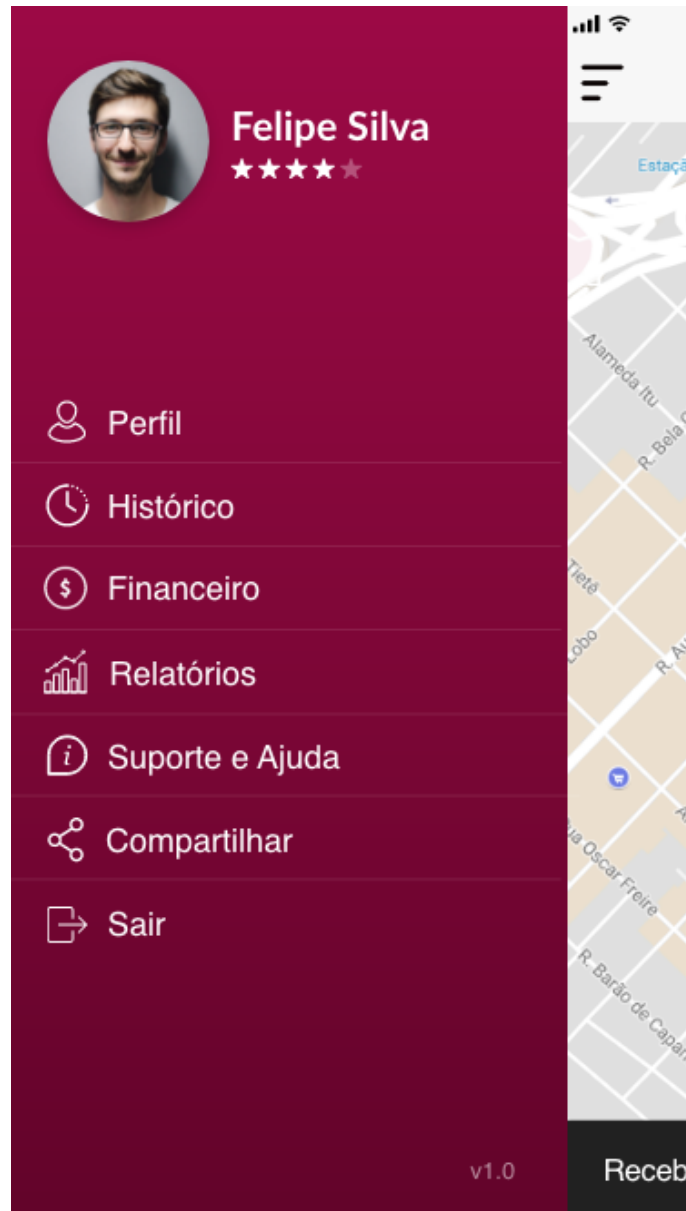


Figura 4.43 - Menu lateral (motorista)

4.2.17 Perfil

Ao selecionar a opção perfil no menu lateral, o usuário é redirecionado a tela de perfil, em que é possível o mesmo visualizar e editar algumas das informações referentes a sua conta.

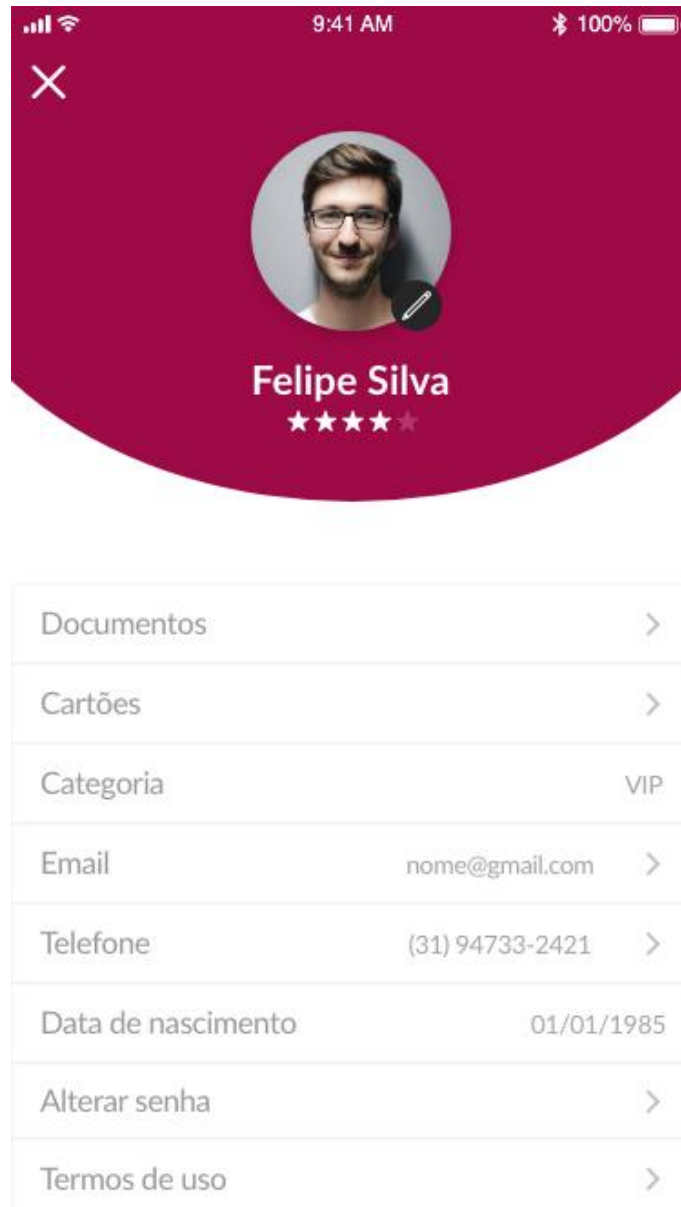


Figura 4.44 – Perfil (motorista)

4.2.18 Perfil - Cartões

Se o motorista selecionar a opção cartões, é exibido uma tela que lista os cartões cadastrados do mesmo e permite a adição de novos cartões.

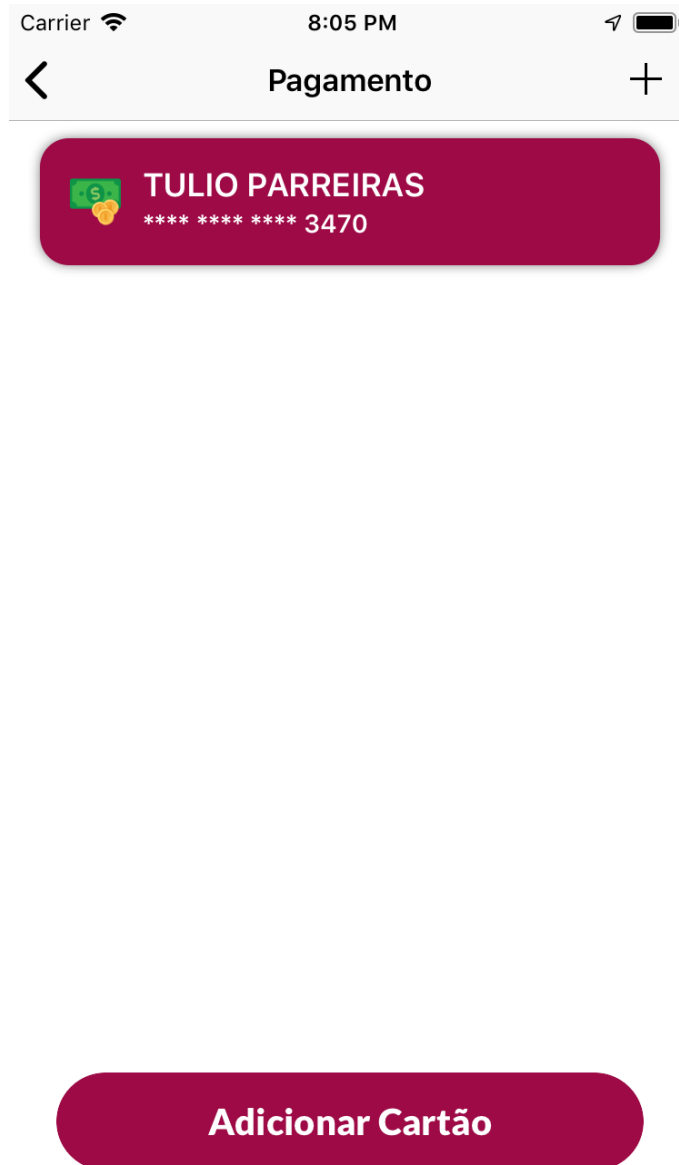


Figura 4.45 – Cartões

4.2.19 Perfil - Email

Ao selecionar a opção email, o usuário é direcionado a uma tela onde pode editar o seu email cadastrado.

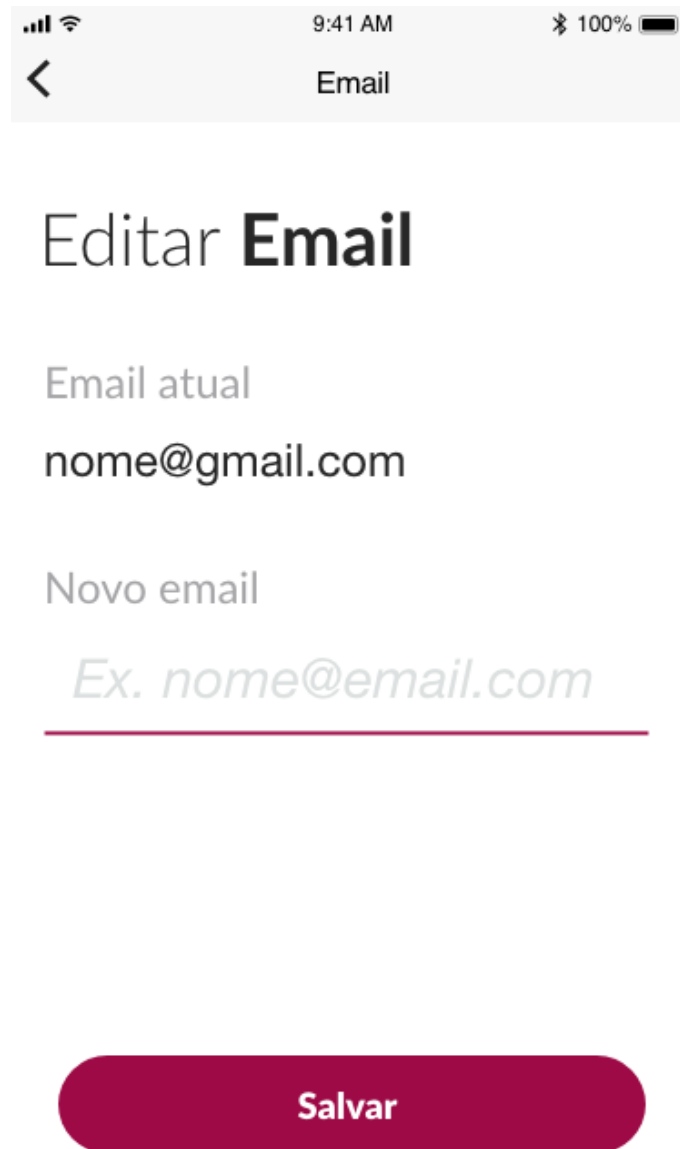


Figura 4.46 - Email

4.2.20 Perfil - Telefone

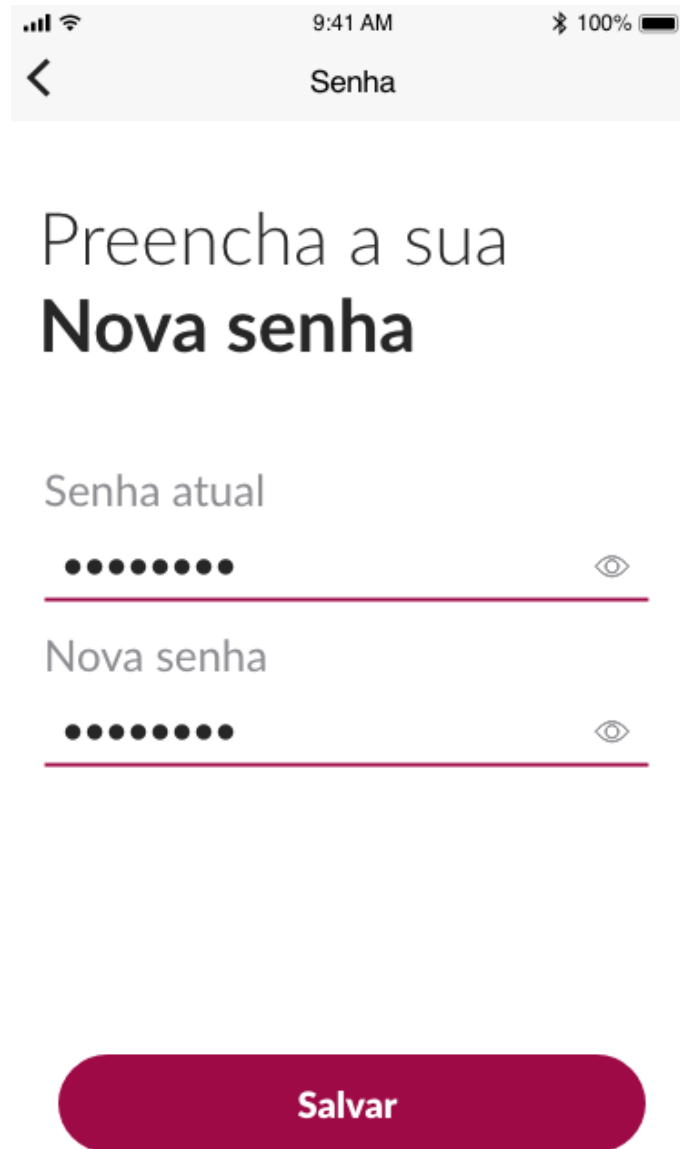
Quando selecionada a opção telefone, é exibida a tela a alteração do telefone.



Figura 4.47 - Telefone

4.2.21 Perfil - Alterar senha


Ao clicar na opção alterar senha, o motorista é apresentado a tela de alteração de senha.




Senhas

Preencha a sua
Nova senha

Senha atual

●●●●●●●● 

Nova senha

●●●●●●●● 

Salvar

Figura 48 - Alterar senha (motorista)

4.2.22 Financeiro

Ao selecionar o item financeiro do menu lateral, é apresentado ao motorista a tela do financeiro do mesmo, em que ele pode visualizar seu balanço financeiro, composto por saldo bloqueado, saldo devedor e saldo disponível. O motorista pode também visualizar os dados da conta cadastrada para saque, bem como editar e atualizar essas informações.



Figura 4.49 - Financeiro

4.2.23 Relatórios

Quando selecionado o menu relatórios, é exibido ao motorista uma tela em que ele pode acompanhar seus relatórios semanais, contendo quais foram seus ganhos diários por semana.



Figura 4.50 - Relatórios

5 CONSIDERAÇÕES FINAIS

Este trabalho apresentou alguns conceitos e princípios envolvidos no desenvolvimento de uma aplicação móvel iOS voltada para mobilidade urbana. Foi falado sobre o que consistem em mobilidade urbana e seus desafios, assim como citado o que envolve um projeto de um aplicativo iOS e por fim especificado como construir um aplicativo tipo Uber iOS.

Com o crescente acesso aos smartphones e a internet móvel, tornou-se possível um novo formato de negócios que faz uso desses dispositivos móveis para conectar prestadores de serviços e clientes, de modo que aplicações como a apresentada nesse projeto têm se tornado uma ferramenta cada vez mais presente no dia-a-dia das pessoas.

Os resultados obtidos no desenvolvimento da aplicação foram satisfatórios dito que foi possível criar dois aplicativos diferentes, passageiro e motorista, que através de uma conexão com o servidor, podem interagir entre si, resultando no processo de pedir uma corrida de forma prática e eficiente.

Também foi observado que apesar das medidas tomadas para deixar o produto mais leve e rápido, a dependência de uma internet de qualidade ainda é um obstáculo a ser superado, uma vez que esse tipo de aplicação depende diretamente de interações com o servidor e banco de dados.

De maneira análoga, com o uso da ferramenta de Realtime Database do Firebase, foi possível notar um bom rendimento em situações de conexão fraca com a internet, por conta de seu recurso de armazenamento local e web socket que mantém uma conexão persistente com o servidor, reduzindo assim a quantidade de requisições ao mesmo, acarretando uma diminuição no consumo de internet pelas aplicações.

5.1 Sugestões para trabalhos futuros

Como continuação e aprimoramento deste trabalho, é possível serem feitas melhorias ainda na parte de interação com o servidor, buscando sempre o menor consumo e dependência da internet, assim em condições de uma conexão de baixa qualidade os aplicativos possam ainda ter um bom desempenho.

É possível também notar que o projeto permite ainda a integração de novas funcionalidades e recursos, de modo a apresentar para o usuário uma experiência cada vez mais imersível. Entre as funcionalidades passíveis de integração podem ser realizadas:

- Funcionalidade apenas mulheres: adicionar um recurso que permita que passageiras mulheres optem por requisitar corridas com motoristas mulheres apenas, de modo a oferecer mais conforto e segurança para as mesmas;
- Alteração de destino: adicionar a opção de o passageiro poder alterar seu destino ao longo do curso de uma viagem já aceita por um motorista;
- Compartilhamento da corrida em tempo real: permitir ao passageiro que o mesmo compartilhe a sua corrida em tempo real com outra pessoa, melhorando assim a segurança do mesmo enquanto usa a plataforma;
- Filtro de viagens por métodos de pagamentos: permitir ao motorista que filtre suas solicitações de viagens apenas para o método de pagamento desejado, assim o mesmo pode escolher entre receber corridas apenas no dinheiro ou apenas no cartão;
- Implementação de modelo de bonificações e promoções: definir um modelo sólido para oferecer aos passageiros promoções e descontos para incentivar os mesmos a usar a plataforma. Paralelamente, oferecer aos motoristas bônus e recompensas por metas pré-definidas de corridas, para manter os mesmos mais motivados a serem parceiros da plataforma;

REFERÊNCIAS BIBLIOGRÁFICAS

99. **A 99**. [S.I.][2019?]. Disponível em <<https://99app.com/sobre-a-99/>>. Acesso em 30 jun. 2019.
- Accengage. **Configure push notifications**. [S.I.][2015?]. Disponível em <<https://docs.accengage.com/display/IOS/Configure+push+notifications>> Acesso em 14 jul. 2019.
- Apple. **App Store**. [S.I.][2019?]. Disponível em <<https://www.apple.com.br/ios/app-store/>>. Acesso em 29 jun. 2019.
- BARROS, V. **Você sabe o que é mobilidade urbana e qual o seu impacto na arquitetura?**. VivaDecoraPRO, 18 jan. 2018. Disponível em <<https://www.vivadecora.com.br/pro/arquitetura/o-que-e-mobilidade-urbana/>>. Acesso em 08 jun. 2019.
- BEZERRA, J. **Mobilidade Urbana no Brasil**. TodaMatéria, 30 jan. 2018. Disponível em <<https://www.todamateria.com.br/mobilidade-urbana/>>. Acesso em 08 jun. 2019.
- CAMPOS, V. B. G. **Uma visão da mobilidade urbana sustentável**. Revista dos Transportes Públicos, v. 2, n. 99-106, p. 4, 2006.
- COSTA, M. da S. **Um índice de mobilidade urbana sustentável**. São Carlos: Escola de Engenharia de São Carlos, Universidade de São Paulo, 2008.
- GALANTE, V. R. **O que é um aplicativo de mobilidade urbana?**. Usemobile 20 fev. 2019. Disponível em <<https://usemobile.com.br/aplicativo-mobilidade-urbana/>>. Acesso em 13 jul. 2019.
- GALANTE, V. R. **Uberização: entenda tudo sobre esse processo**. Usemobile, 13 nov. 2018. Disponível em <<https://usemobile.com.br/uberizacao-entenda-tudo-sobre-esse-processo/>>. Acesso em 30 jun. 2019.
- GLASCO, J. **Urban Mobility: Challenges & Solutions in Smart Cities**. Bee smart city 1 fev. 2019. Disponível em <<https://hub.beesmart.city/solutions/en/smart-mobility/smart-mobility-challenges-and-solutions-in-smart-cities>>. Acesso em 14 jul. 2019.

Google. **Maps SDK for iOS**. [S.I.][2019?]. Disponível em <<https://developers.google.com/maps/documentation/ios-sdk/intro?hl=pt-br>>. Acesso em 30 jun. 2019

KOTIPALLI, S.R., IMRAN, M. A. **Hacking Android**. Jul.2016

MCMAHON, J. The Four Problems Of Urban Transportation (And The Four Solutions). Forbes, 10 mar. 2018. Disponível em <<https://www.forbes.com/sites/jeffmcmahon/2018/03/19/the-four-problems-of-urban-transportation-and-the-four-separate-solutions/#410b5ea51afb>>. Acesso em 14 jul. 2019.

MELLO, C. A. **O futuro da mobilidade urbana e o caso Uber**. Revista de Direito da Cidade v.8, nº 2, p. 775-812.

SASAKI, F. **O desafio da mobilidade urbana no Brasil**. Guia do Estudante, 12 jan. 2017. Disponível em <<https://guiadoestudante.abril.com.br/blog/atualidades-vestibular/o-desafio-da-mobilidade-urbana-no-brasil/>>. Acesso em 08 jun. 2019.

SINIMBÚ, F. **Mobilidade urbana é desafio para cidades e trabalhadores**. Agência Brasil, 17 ago. 2017. Disponível em <<http://agenciabrasil.ebc.com.br/geral/noticia/2017-07/mobilidade-urbana-e-desafio-para-cidades-e-trabalhadores>>. Acesso em 08 jun. 2019.

UBER. **A história da Uber**. [S.I.][2019?]. Disponível em <<https://www.uber.com/pt-BR/newsroom/História/>>. Acesso em 08 jun. 2019.

UBER. **Informações da Empresa**. [S.I.][2019?]. Disponível em <<https://www.uber.com/pt-BR/newsroom/informções%20da%20empresa/>>. Acesso em 08 jun. 2019.

