UNIVERSIDADE REGIONAL DE BLUMENAU

CENTRO DE CIÊNCIAS EXATAS E NATURAIS

CURSO DE SISTEMAS DE INFORMAÇÃO – BACHARELADO

BEAVER: UM APLICATIVO WEB PARA LOCALIZAR PRESTADORES DE SERVIÇO POR GEOLOCALIZAÇÃO

LUCAS AFONSO LOMBARDI MOREIRA

BLUMENAU
2017

LUCAS AFONSO LOMBARDI MOREIRA

BEAVER: UM APLICATIVO WEB PARA LOCALIZAR PRESTADORES DE SERVIÇO POR GEOLOCALIZAÇÃO

Trabalho de Conclusão de Curso apresentado ao curso de graduação em Sistemas de Informação do Centro de Ciências Exatas e Naturais da Universidade Regional de Blumenau como requisito parcial para a obtenção do grau de Bacharel em Sistemas de Informação.

Profª Luciana Pereira de Araújo, Mestra - Orientadora

BLUMENAU
2017

BEAVER: UM APLICATIVO WEB PARA LOCALIZAR PRESTADORES DE SERVIÇO POR GEOLOCALIZAÇÃO

Por

LUCAS AFONSO LOMBARDI MOREIRA

Trabalho de Conclusão de Curso aprovado para obtenção dos créditos na disciplina de Trabalho de Conclusão de Curso II pela banca examinadora formada por:

Presidente:	Profa. Luciana Pereira de Araújo, Mestra – Orientadora, FURB
Membro:	Profa. Simone Erbs da Costa, Especialista – FURB
Membro:	Prof. Miguel Alexandre Wisintainer, Mestre – FURB

Blumenau, 12 de dezembro de 2017

Dedico este trabalho a minha família, meus amigos e a todos que colaboraram de alguma forma na elaboração deste trabalho.

AGRADECIMENTOS

Aos meus pais que lutaram muito para que eu chegasse até essa etapa da minha vida.

Aos meus padrinhos, que tornaram o sonho de ingressar nesse curso possível.

A minha orientadora, Prof.ª Luciana Pereira de Araújo pela paciência, motivação e ajuda na realização do trabalho.

Aos meus irmãos na amizade pelo incentivo e apoio constantes.

O primeiro passo é estabelecer que algo é possível.

Elon Musk

RESUMO

Este trabalho tem como objetivo apresentar um aplicativo web responsivo para dispositivos móveis que tem o intuito de criar uma opção para o prestador de serviço autônomo de anunciar o seu trabalho para o mercado e de ajudar o cliente a encontrar um serviço. O aplicativo, denominado Beaver foi desenvolvido pensando no consumo colaborativo, criando uma rede de serviços, permitindo que o usuário possa expor o seu trabalho, buscar por serviços de outros usuários, solicitar serviços e avaliar o serviço prestado, contando com um chat para interação entre os usuários. Para o desenvolvimento do trabalho foi utilizada a plataforma NodeJS no módulo servidor, o Quasar *framework* no aplicativo e o banco de dados MySQL. O aplicativo desenvolvido foi colocado em teste para três membros especialistas que foram convidados a realizar uma avaliação heurística a fim de aferir o grau de usabilidade da aplicação desenvolvida, encontrando problemas e discutindo suas soluções.

Palavras-chave: Prestadores de serviço. Consumo colaborativo. Aplicativo web.

ABSTRACT

This essay presents a responsive web application for mobile devices that is intended to create a way for the independent service provider to announce their work to the market and to help the customer find a service. The application, called Beaver was developed with the intention of sharing economy, creating a network of services, allowing the user to expose their work, search for services of other users, request their services, evaluate the servisse provided and talk on a chat with users. For the development of the work, was used in the server module the NodeJS platform, in the application was used the Quasar framework and for database was used MySQL. We tested the application with three experts. They were invited to carry out a heuristic evaluation in order to assess the degree of usability application.

Key-words: Service providers. Sharing economy. Web app.

LISTA DE FIGURAS

LISTA DE QUADROS

LISTA DE ABREVIATURAS E SIGLAS

API – Application Programming Interface

CLT – Consolidação das Leis de Trabalho

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

MER – Modelo Entidade Relacionamento

ORM – Object Relactional Modeling

PWA – Progressive Web App

REST – Representational State Transfer

RF – Requisitos Funcionais

RNF – Requisitos Não Funcionais

UC – Diagrama de Casos de Uso

UML – Unified Modeling Language

SUMÁRIO

1 INTRODUÇÃO

Atualmente, muitos são os trabalhadores que optam pelo trabalho autônomo e na maioria das vezes, estes autônomos optam pela prestação de serviços (SARDAGNA, 2016). De acordo com Ost (2008), a maioria dos trabalhadores brasileiros, não possuem carteira assinada, e segundo Sardagna (2016, p. 1), "[...] cerca de 23% dos trabalhadores brasileiros atua de forma autônoma [...]". Isso porque, segundo Negocio (2015, p. 1), abrindo uma empresa por conta própria, o profissional pode atuar de forma independente, tendo um horário e local flexível, além de poder colher os frutos financeiros de sua atividade. Apesar das vantagens da autonomia como a maior flexibilidade e o fato de ser seu próprio dono, existem também alguns desafios. Um dos desafios que o prestador de serviço enfrenta é que eles são o produto ofertado e por este motivo, precisam se preocupar em buscar e fidelizar mais clientes.

Asaas (2013, p. 1) explica que "A parte mais complexa do empreendedorismo individual é se estabelecer no mercado, conquistar credibilidade no segmento e captar novos clientes.", sendo que a divulgação é muito importante para o profissional autônomo que precisa estar sempre aumentando sua rede de contatos para conseguir mais clientes. Silva (2016), cita que os trabalhadores autônomos nem sempre possuem outra fonte de renda, tornando difícil investir no seu marketing pessoal. Segundo Sardagna (2016), a troca de informação verbal entre as pessoas, conhecido como "boca a boca", ainda é uma das armas mais poderosas para se promover no mercado, porém é preciso utilizar de novas tecnologias para conseguir um desempenho melhor, alcançando assim um número maior de clientes. Sardagna (2016) continua apontando a importância em utilizar os meios digitais para divulgação de seu trabalho, citando exemplos como: sites ou redes sociais onde você possa expor seu portfólio ou currículo, além de formas de contato para as pessoas interessadas.

Segundo Filantropia (2015), uma opção que está virando tendência é o consumo colaborativo, que Lima e Guilen (2012, p. 1) descrevem como "[...] um tipo de prática mercantil [...] onde pessoas buscam formas de usufruir de bens em serviços de maneira diversa da tradicional baseada na troca por moeda corrente entre comprador e vendedor.". Uma opção de consumo colaborativo, segundo Lima e Guilen (2012), são os sistemas de serviços de produtos que ofertam uma solução ao cliente, ao invés de vender um produto.

Barranger (2014) comenta que, com o crescimento destes sistemas colaborativos, em alguns lugares, as leis já estão sendo inclusive adaptadas, mostrando a grande aceitação do público. De acordo com Barranger (2014, p. 1), o consumo colaborativo traz "[...] a

possibilidade do cidadão comum poupar dinheiro por meio da divisão de gastos e ganhar dinheiro adicional investindo um pouco do tempo ou algumas propriedades.”.

Baseado nesse cenário, este trabalho apresenta o desenvolvimento de um aplicativo web responsivo para dispositivos móveis que possibilita o intermédio entre os prestadores de serviços locais e os consumidores que procuram por esses serviços por meio de geolocalização utilizando o modelo de negócios de consumo colaborativo.

1.1 OBJETIVOS

O objetivo deste trabalho é disponibilizar um aplicativo web responsivo para encontrar prestadores de serviços locais autônomos por geolocalização.

Os objetivos específicos são:

a) oferecer uma alternativa para prestadores de serviços de anunciar seu trabalho;
a) possibilitar que indivíduos encontrem prestadores de serviços autônomos por geolocalização.

1.2 ESTRUTURA

Esta monografia está dividida em quatro capítulos. No primeiro capítulo é apresentada a introdução ao assunto do trabalho e descritos os objetivos do trabalho. O segundo capítulo destina-se a fundamentação teórica explorando conceitos de trabalhador autônomo, prestação de serviços, consumo colaborativo, geolocalização e os trabalhos correlatos. No terceiro capítulo é demonstrado o desenvolvimento do aplicativo, listando os requisitos, apresentando diagramas, técnicas e ferramentas utilizadas, operacionalidade do sistema e os resultados obtidos. Por fim, o quarto capítulo descreve as conclusões do trabalho e apresenta sugestões para trabalhos futuros.

2 FUNDAMENTAÇÃO TEÓRICA

Este capítulo aborda os principais conceitos utilizados para o desenvolvimento deste trabalho, sendo dividido em cinco seções. A primeira seção aborda sobre o trabalhador autônomo, com enfoque na prestação de serviço. A segunda seção descreve o consumo colaborativo. Por fim, a terceira seção apresenta os trabalhos correlatos a este, bem como uma comparação entre eles.

2.1 TRABALHADOR AUTÔNOMO

Ost (2008, p. 1) descreve trabalhador autônomo como "[...] todo aquele que exerce sua atividade profissional sem vínculo empregatício, por conta própria e com assunção de seus próprios riscos.". O trabalhador autônomo, normalmente ganha independência econômica e financeira (FRANCINI, 2016). Poloni (2003) explica que a subordinação é a diferença entre o trabalhador autônomo e o empregado, enquanto o empregado é totalmente subordinado, o autônomo, como o nome sugere, é independente. Assim, o autônomo tem liberdade para executar o seu trabalho, da forma que lhe convir e no momento que preferir.

Segundo Neri et al. (2007) este tipo de profissional vem crescendo cada vez mais no ambiente urbano, trabalhando menos na agricultura e mais com serviços. Viana (2005) afirma que o trabalho autônomo é a única opção para um grande número de trabalhadores. R. Junior (2017) descreve muitas vantagens em trabalhar como autônomo, como a flexibilidade de horário e local de trabalho, poder escolher quais trabalhos executar e com quais clientes irá trabalhar, além de poder definir o valor do serviço. No entanto, R. Junior (2017) ressalva que é preciso ter muita disciplina para o cumprimento dos prazos e cobrança. Além disso, é necessário se preparar para as dificuldades, como calote de clientes, captação de novos clientes além da instabilidade financeira (R. JUNIOR, 2017).

> O aumento de trabalhadores autônomos é uma nova realidade social, frente ao crescimento do comercio informal, abrangendo várias áreas, onde se busca alternativas ao sistema tradicional da relação de emprego, onde obrigatoriamente os trabalhadores se submetem a normas imperativas (SANTOS, 2012, p. 10).

Segundo Ost (2008), existem dois tipos de trabalhadores autônomos: os prestadores de serviços de profissões regulamentadas e de profissões não regulamentadas. Profissões regulamentadas são aquelas que possuem, em sua legislação, obrigações, direitos e deveres para exerce-las. Já as não regulamentadas apesar de serem abrangidas pela Consolidação das Leis do Trabalho (CLT), podem ser exercidas sem a necessidade de uma formação formal (MUNDO CARREIRA, 2014). Algumas profissões regulamentadas citadas por Ost (2008)

são advogado, médico e psicólogo, já as não regulamentadas citadas estão: encanador, faxineiro e pedreiro.

2.1.1 PRESTAÇÃO DE SERVIÇOS

Segundo Sardagna (2016), muitos profissionais autônomos escolhem ser prestadores de serviços, abrangendo diversas profissões, sendo principalmente as não regulamentadas. Para Zeithaml, Bitner e Gremler (2014, p. 4), "[...] serviços são atos, processos e atuações oferecidos ou coproduzidos por uma entidade ou pessoa, para outra entidade ou pessoa.". Avila e Avila (2001) explicam que os serviços, independente da área, são produtos ofertados para satisfazer necessidades dos clientes, assim como um bem ou uma mercadoria. Oliveira (2012) descreve quatro características que mostram a diferença entre a venda de serviços e a venda de produtos:

a) intangibilidade: um serviço não é algo que possa ser tocado fisicamente;
b) inseparabilidade: independente do nível de integração do cliente com o prestador, ambos influenciam diretamente nos resultados alcançados na prestação do serviço;
c) heterogeneidade: um serviço pode sofrer influência de diversos fatores e portanto, nenhum serviço prestado é exatamente igual à outro;
d) perecibilidade: um serviço não pode ser estocado.

Gasparetto (2012) conta que antigamente a agricultura era muito importante na economia do país, oferecendo alimentos para nossa sobrevivência. Os poucos serviços que eram oferecidos à população, normalmente eram "[..] realizados ou por escravos ou por quem não podia, pelas suas condições físicas principalmente, trabalhar na dura labuta da lavoura: idosos, mulheres e deficientes." (GASPARETTO, 2012, p. 1). Por isso, prestar um serviço, nessa época era considerada uma atividade de segunda classe. Hoje, esta prática passa a ser fundamental para a economia e ser executada por profissionais qualificados (GASPARETTO, 2012).

Martins, Ramos e Ramos (2000) afirmam que a tecnologia está fortemente ligada a prestação de serviços, trazendo automatização, ganho na produtividade, na qualidade e velocidade na entrega do serviço. Gasparetto (2012) explica que, antigamente, para conseguir algum serviço, era preciso ir até o prestador de serviço em pessoa e solicitar o serviço. No passado, de acordo com Zeithaml, Bitner e Gremler (2014), toda a prestação de serviço era feita pessoalmente, havendo uma interação entre o funcionário e o cliente. No entanto, com o invento do telefone, esse cenário muda, permitindo que os clientes solicitem um serviço de suas casas (ZEITHAML; BITNER; GREMLER, 2014).

2.2 CONSUMO COLABORATIVO

Uma prática que vem sendo bastante divulgada na internet e que serve como solução para que pessoas possam divulgar seus serviços é o consumo colaborativo (LIMA; GUILEN, 2012). Segundo Filantropia (2015), o conceito de Consumo Colaborativo surgiu nos Estados Unidos durante a crise econômica de 2009. Barranger (2014, p. 1) descreve consumo colaborativo como: "[...] um sistema de troca de bens ou serviços.". O foco desse modelo é a troca de bens ou serviços entre pessoas, seja no aluguel de carros, livros, brinquedos, ou até acomodar indivíduos para jantar em sua casa. Assim, este modelo possui um grande valor social, além de diminuir o desperdício e melhorar a eficiência do acesso a bens e serviços (BARRANGER, 2014).

Lima e Guilen (2012) dividem o consumo colaborativo em três segmentos, sendo: sistemas de serviços de produtos, nos quais o produto é substituído por um serviço que o representa, como locadoras de veículos ou filmes; mercados de redistribuição, que é quando um item, que não é mais utilizado, é passado para outra pessoa, como o caso de venda de usados e feiras de trocas; e, por fim, estilos de vida colaborativos, que são pessoas que compartilham seu tempo e espaço para se conectar mais com a comunidade, como compartilhar tarefas entre os vizinhos ou ceder um espaço para turistas ficarem.

O consumo colaborativo pode gerar economia de custos, obtenção de renda, reforço do sentimento de pertencimento a uma comunidade, socialização e consciência ambiental. Isso serve de motivação para que pessoas participem deste meio (LIMA; GUILEN, 2012). Além disso, Lima e Guilen (2012), afirmam que é possível conseguir um aumento da vida útil de produtos, intensificar o uso de objetos menos utilizados, reduzir a quantidade de resíduos que vão para os aterros e diminuir a extração de recursos naturais.

Outro fator que serve de motivação para este meio é a viabilização desse tipo de comércio pela tecnologia digital (BASILIO, 2013). Segundo Basilio (2013), a tecnologia digital é a motivação para o meio, pois ela facilita o contato entre compradores e vendedores e viabiliza a automatização das operações, além de promover a confiança entre os envolvidos.

> As recentes crises econômicas de impacto global atreladas às questões ambientais que afligem a humanidade, forçam os países, as corporações industriais e as pessoas a buscarem novas formas de se relacionar com o desenvolvimento, entre si e com o meio ambiente. No contexto atual marcado pelo aquecimento global, crise financeira, consumismo, pico do petróleo e disputa comercial, a preocupação com o meio ambiente assumiu posição de destaque (LIMA; GUILEN, 2012, p. 1).

Segundo Filantropia (2015, p. 1), "o consumo colaborativo configura uma inovação na forma de consumir [...]". Assim, o consumo colaborativo surge como uma possível solução

para conseguir viabilizar negócios de maneira mais sustentável. Já que com o consumo colaborativo é preciso gastar menos energia e utilizar menos matéria-prima para conseguir um estoque enorme de produtos (BASILIO, 2013).

2.3 TRABALHOS CORRELATOS

Podem-se citar como trabalhos correlatos o Uber (2017) que é um serviço eletrônico na área do transporte privado urbano mantido pela empresa de mesmo nome, além das monografias de Campos (2015) e Silva (2016). Além disso, por fim são apresentadas as principais características entre os trabalhos correlatos, de modo a comparar os três trabalhos.

2.3.1 Uber

O Uber é um aplicativo para encontrar prestadores de serviços próximos, voltado exclusivamente para serviços de motorista particular. Além do aplicativo, Uber (2017) ainda conta com um sistema web no qual permite que o motorista gerencie sua conta e veja seu histórico de corridas, entre outras funcionalidades (UBER, 2017). Na Figura 1 é apresentada a tela inicial do sistema web Uber (UBER, 2017). Nessa tela, o usuário pode se cadastrar para virar um motorista no aplicativo, informando alguns dados pessoais nos campos que se encontram no centro da tela. A tela possui também, acima do cadastro do motorista, o botão `CADASTRE-SE` para se cadastrar como usuário solicitante.

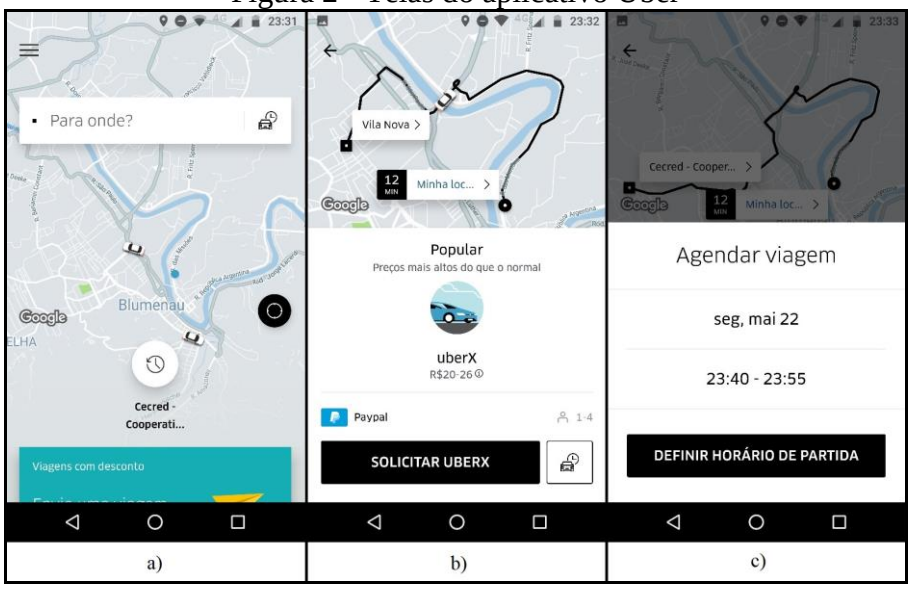
Figura 1 - Página inicial do Uber

Fonte: Uber (2017).

Para utilizar o aplicativo é preciso primeiro baixá-lo em seu dispositivo móvel e fazer um cadastro, que pode ser feito tanto na tela inicial do aplicativo quanto na página web. Após

fazer *login* no aplicativo, o usuário pode solicitar uma corrida que é enviada para os motoristas próximos ao usuário (VAZ, 2015). O usuário não escolhe o motorista, o aplicativo envia um alerta aos motoristas próximos, podendo eles aceitar ou não (VAZ, 2015). Ao aceitar a corrida, o motorista tem seus dados compartilhados com o usuário, como seu nome, uma foto, a placa do carro e sua avaliação (VAZ, 2015). Na Figura 2 (a) é possível visualizar a tela inicial do aplicativo Uber. Nesta tela, é possível solicitar uma corrida, digitando o local de destino no campo `Para onde?` presente na parte superior da tela. No mapa que aparece desta tela, ainda é possível visualizar com base na sua localidade se há algum veículo próximo.

Figura 2 - Telas do aplicativo Uber



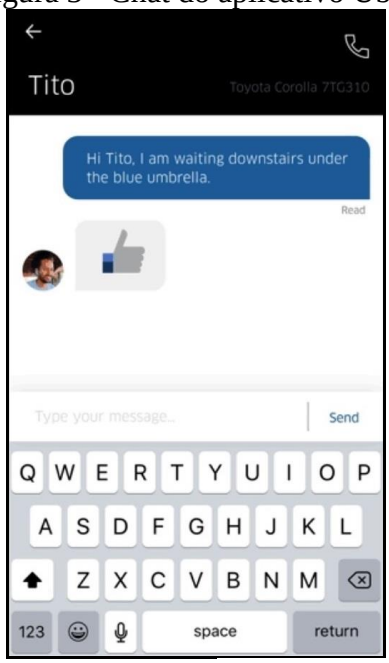
Fonte: Digitalizado de Uber (2017).

Após fazer a solicitação da corrida, aparecerá um menu, como na Figura 2 (b), que possibilita ao usuário escolher o tipo de serviço contratado e a forma de pagamento. O usuário pode pagar com dinheiro em espécie, cartão de crédito ou PayPal, sendo que para as duas últimas opções é necessário que o usuário tenha cadastrado seu cartão ou sua conta PayPal (MALEK, 2017). O exemplo mostrado na Figura 2 apresenta uma solicitação de corrida em uma região em que só existe o tipo de serviço `uberX`, porém existem regiões em que é possível utilizar outros tipos de serviços além deste (como Uber Black, Uber Select, entre outros). O botão `SOLICITAR UBERX` envia a solicitação para os motoristas próximos ao usuário. Ao lado desse botão está o botão para fazer um agendamento de viagem, conforme é mostrado na Figura 2 (c). Nessa opção, o usuário seleciona a data e o horário do agendamento utiliza o botão `DEFINIR HORÁRIO DE PARTIDA` para confirmar o agendamento.

Depois que o motorista aceita a corrida, o usuário solicitante e o motorista podem trocar mensagens através de um chat conforme apresenta a Figura 3. Nesse chat o usuário

conversa com o motorista pelo próprio aplicativo, sem ter que utilizar um aplicativo de terceiro.

Figura 3 - Chat do aplicativo Uber



Fonte: adaptado de Iclarified (2017).

Para auxiliar o motorista com o caminho, o aplicativo Uber trabalha com três ferramentas diferentes para navegação: o Google Maps, o Waze e um mapa próprio que é disponibilizado pela empresa (G1, 2017). Caso o motorista integre seu aplicativo com uma das duas primeiras opções, ao começar a corrida, o aplicativo Uber fica em segundo plano e abre o aplicativo correspondente.

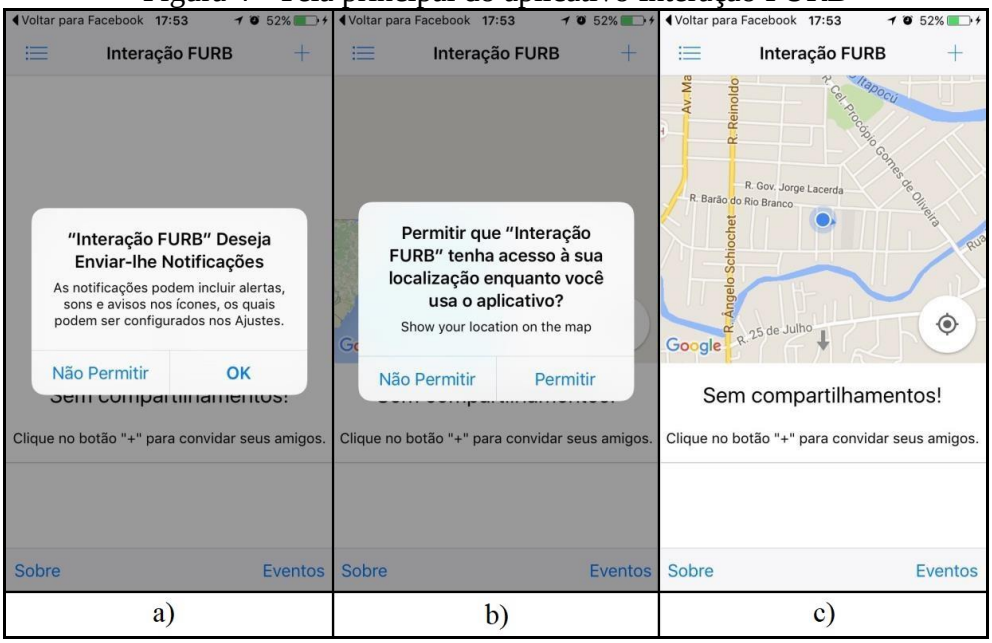
Depois da corrida, passageiro e motorista podem dar sua avaliação de sua experiência na corrida, dando de uma a cinco estrelas. Segundo Uber (2016, p. 1), “As avaliações são sempre exibidas como médias. Nem passageiros nem motoristas conseguem visualizar a avaliação individual de uma viagem específica.”. O sistema de avaliação, além de visar segurança para passageiros e motoristas, também serve de controle para que haja respeito entre ambos na corrida, pois o motorista que tiver nota baixa, para de circular e o passageiro com nota ruim é banido do aplicativo (GRAGNANI, 2016).

2.3.2 Sistema móvel na plataforma PhoneGap para compartilhamento de geolocalização integrado a rede social

O trabalho de Campos (2015) teve como objetivo o desenvolvimento de um aplicativo para facilitar a locomoção dos visitantes do evento Interação na universidade regional de Blumenau (FURB) pelo campus da universidade. O aplicativo permite o compartilhamento de

localização entre amigos. Como o aplicativo está integrado à rede social Facebook, os visitantes podem entrar nele utilizando as informações de seu perfil e enviar convites de compartilhamento de localização para suas listas de amigos. Na Figura 4 tem-se a tela principal do aplicativo de Campos (2015) em três situações diferentes.

Figura 4 - Tela principal do aplicativo Interação FURB



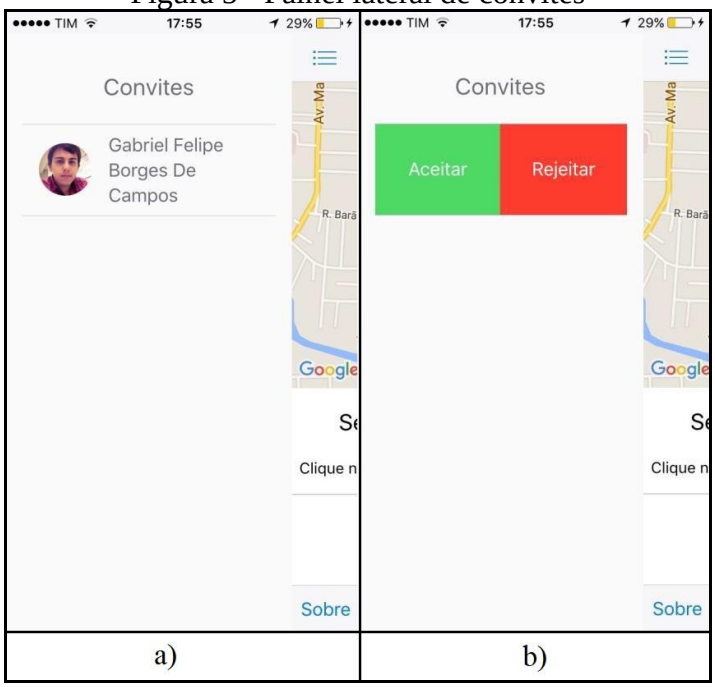
Fonte: Adaptado de Campos (2015).

Na Figura 4 (a) é apresentado um pedido de permissão para o envio de notificações sobre eventos e novas atualizações. Na Figura 4 (b) é possível ver um pedido de permissão para uso da localização do usuário para o funcionamento do aplicativo. Essas duas primeiras situações ocorrem ao abrir o aplicativo pela primeira vez, para que o usuário esteja ciente das funcionalidades do seu dispositivo que o aplicativo ocupará. A Figura 4 (c) mostra um mapa no centro da tela com a localização atual do usuário e o botão +, localizado no canto superior direito, para convidar um amigo a compartilhar sua localização. Nesta opção o usuário acessa a tela de compartilhamento, a qual é carregada uma lista de amigos que também possuem o aplicativo associado em suas contas do Facebook. O usuário então pode enviar um convite de compartilhamento de localização aos seus amigos da rede social. Nos cantos inferiores são apresentados, dois links. O link `sobre` serve para apresentar informações sobre o aplicativo enquanto o link `Eventos` abre uma lista dos eventos e informações que são registradas pela universidade.

Quando o usuário convidado entra no aplicativo, ele pode visualizar seus convites recebidos, podendo aceitar ou rejeitar a solicitação de compartilhamento. Se o convite for aceito, ambos os dispositivos passam a compartilhar suas localizações (CAMPOS, 2015). Como exibido na Figura 5 (a), é possível acessar um painel lateral através do menu localizado

no canto superior esquerdo da tela inicial do aplicativo. Esse painel mostra os convites que foram recebidos de outros usuários. No convite, são apresentados o nome e uma foto do usuário que enviou o convite. Na Figura 5 (b) tem-se a opção de aceitar ou recusar o convite, que também é acessada pelo menu lateral.

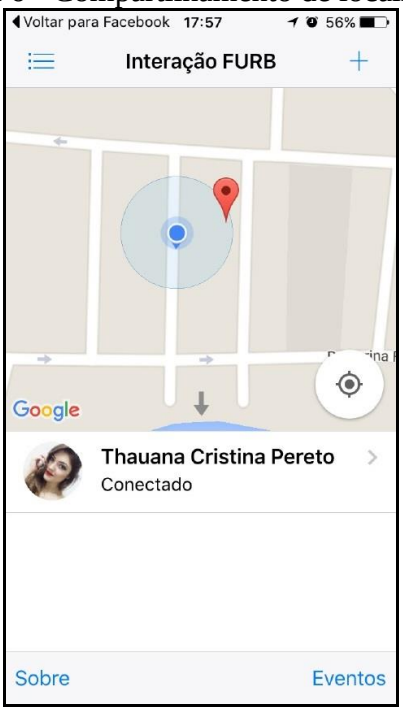
Figura 5 - Painel lateral de convites



Fonte: Adaptado de Campos (2015).

Na Figura 6 é possível visualizar o compartilhamento de localização entre os usuários, no qual é apresentado um ponto azul que representa a posição atual do usuário e um marcador vermelho identificando a posição atual de um amigo que está compartilhando sua localização. Ao clicar sobre o indicador vermelho, tem-se as informações do usuário que compartilhou a localização, como o nome e foto localizados na parte inferior da tela. Além disso, o aplicativo utiliza o sistema de navegação Google Maps para buscar as localizações dos usuários. O aplicativo foi desenvolvido utilizando a plataforma PhoneGap, para que ele possa ser utilizado em diferentes sistemas operacionais (CAMPOS, 2015).

Figura 6 - Compartilhamento de localização

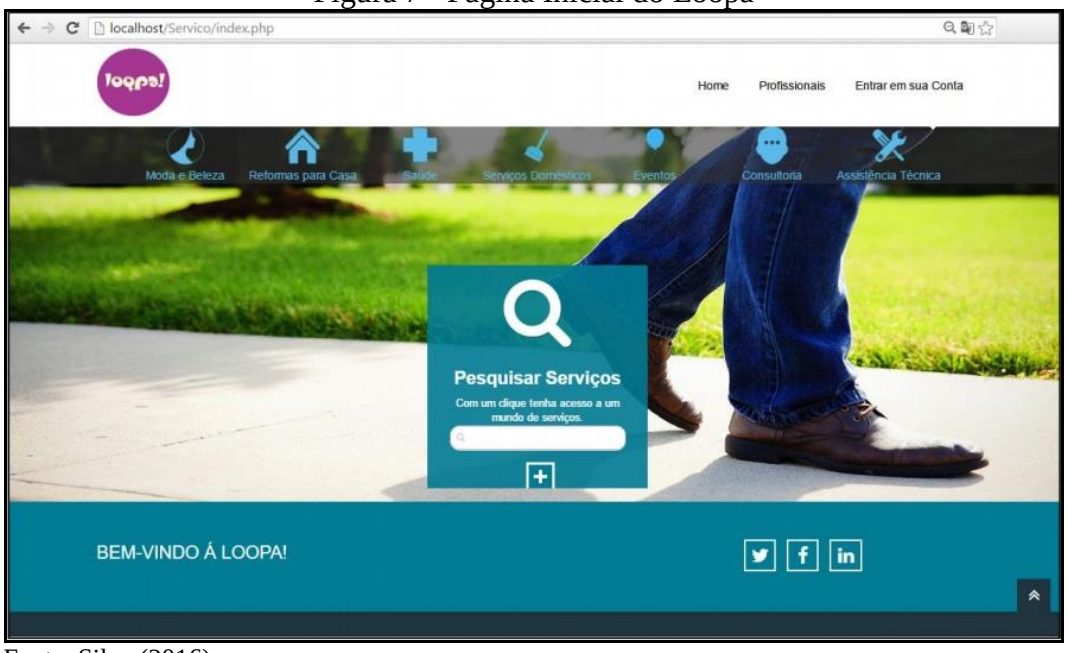


Fonte: Campos (2015).

2.3.3 Sistema de consumo colaborativo com foco em oferta de serviços locais

O trabalho de Silva (2016), teve por objetivo desenvolver um sistema web de consumo colaborativo com foco na oferta de serviços locais. O sistema é dividido em duas áreas, sendo: consumidor, que solicita serviços; e prestador de serviços, que deseja oferecer algum tipo serviço no sistema. Na Figura 7 é possível visualizar a página inicial do sistema desenvolvido por Silva (2016).

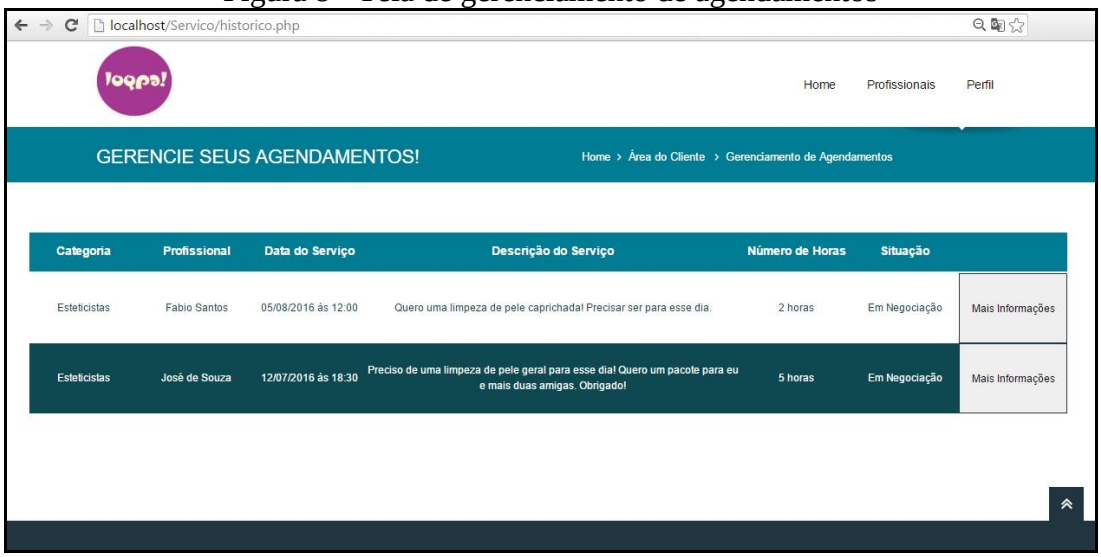
Figura 7 - Página Inicial do Loopa



Fonte: Silva (2016).

Como consumidor, o usuário pode procurar pelo serviço que precisa e consegue visualizar as informações dos prestadores de serviços para decidir se vai agendar o serviço ou não. Para isso, o usuário não precisa fazer *login* no sistema, sendo necessário somente caso o usuário queira agendar o serviço. Uma vez que o consumidor crie uma solicitação de agendamento de serviço, é enviado uma notificação para o prestador de serviço possibilitando ao mesmo confirmar o agendamento ou propor uma nova data de agendamento. Após o término do serviço, o consumidor pode avaliar o prestador. Nas Figura 8 é possível visualizar a página de gerenciamento dos agendamentos do cliente.

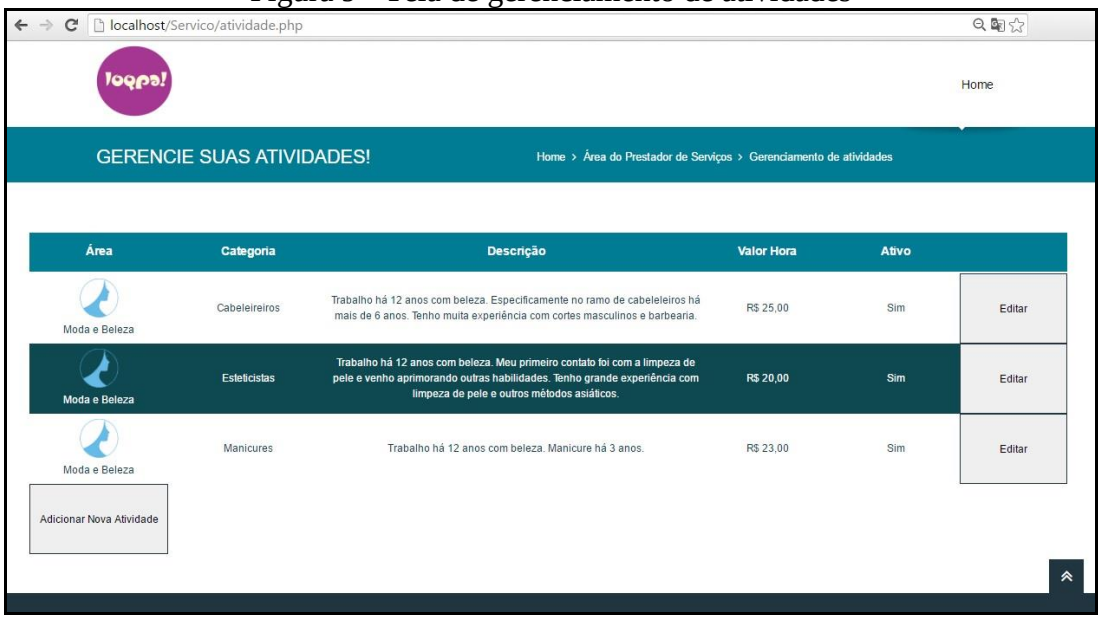
Figura 8 - Tela de gerenciamento de agendamentos



Fonte: Silva (2016).

Na página apresentada na Figura 8, são listados no centro da tela os agendamentos feito pelo usuário. Nessa lista são mostradas as principais informações sobre o agendamento, como a data agendada, seu prestador do serviço, a descrição do que foi pedido e a sua situação. Ao lado direito da lista, existe o botão `Mais informações` que direciona o usuário a página que são apresentadas as informações gerais da solicitação, como por exemplo, informações do profissional, preço que o prestador cobra por hora, descrição do tipo de serviço prestado, data do agendamento e número de horas solicitado. A Figura 9 apresenta a página de gerenciamento das atividades do prestador de serviço, na qual o prestador pode visualizar todas as atividades que ele está vinculado em uma lista.

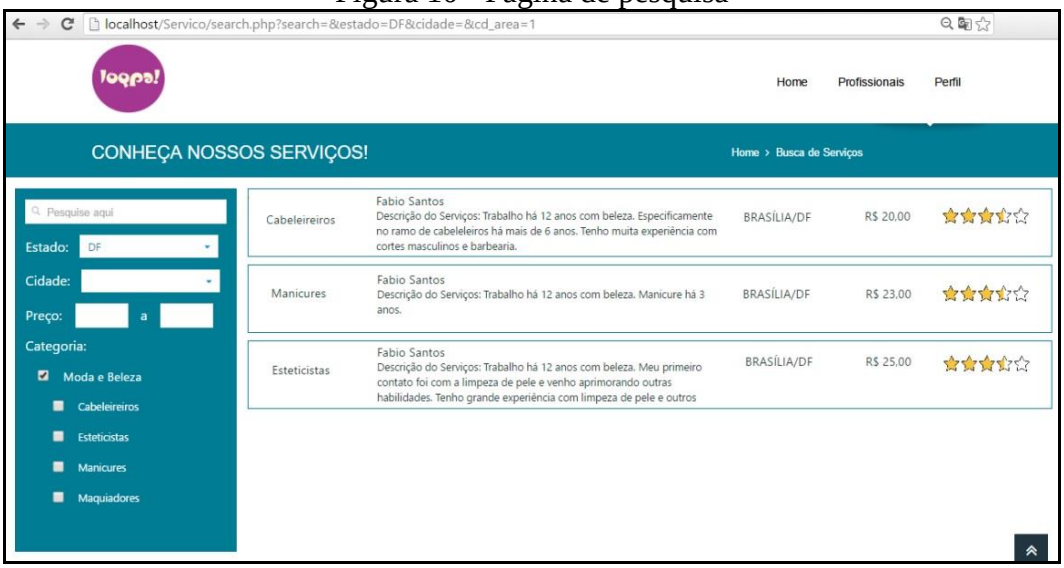
Figura 9 - Tela de gerenciamento de atividades



Fonte: Silva (2016).

Nessa página, o usuário pode, além de consultar suas atividades, incluir novas, através do botão `Adicionar Nova Atividade`, que fica no canto esquerdo inferior da página. É possível também editar as atividades já cadastradas acessando o botão `Editar` no lado direito da respectiva atividade da lista. A Figura 10 apresenta a página de pesquisa, na qual são listados os serviços cadastrados.

Figura 10 - Página de pesquisa



Fonte: Silva (2016).

Na página de pesquisa, há diversos filtros localizados no lado esquerdo da tela para encontrar o serviço adequado. É possível selecionar o estado, a cidade, escolher os preços e até selecionar a categoria do serviço. Na lista de serviços, localizada no centro da tela, são mostradas as informações do serviço oferecido e uma média de suas avaliações. Quando o

usuário selecionar um dos serviços, ele é direcionado a página de agendamento de serviço, pelo qual pode ser solicitado um agendamento.

2.3.4 Comparação entre os trabalhos relacionados

Esta seção apresenta uma comparação entre os trabalhos apresentados. No Quadro 1 é apresentado um comparativo entre os trabalhos correlatos, de modo que a primeira coluna representa as características em comum entre os trabalhos e as demais colunas representam os trabalhos relacionados.

Quadro 1 - Comparativo entre os trabalhos correlatos

Características	Uber (2017)	Campos (2015)	Silva (2016)
Plataforma	Android / IOS	IOS	Web
Modelo de consumo colaborativo	Sim	Não	Sim
Ferramenta de Geolocalização	Google Maps, Waze e Uber	Google Maps	Não possui
Busca pelo prestador de serviço	Por meio de um mapa	Não se aplica	Lista
Chat para os usuários	Sim	Não	Não
Permite avaliação dos clientes	Sim	Não	Sim
Avaliações feitas por estrelas	Sim	Não	Sim

Fonte: elaborado pelo autor.

Como tem-se no Quadro 1, tanto o Uber (2017) quanto o trabalho de Silva (2016) utilizam o modelo de consumo colaborativo, pois têm foco na divulgação de prestadores de serviço, sendo o Uber (2017) voltado para motoristas particulares e o trabalho de Silva (2016) para prestadores de serviços locais. Enquanto Uber (2017) e Campos (2015) estão voltados aos dispositivos móveis, a ferramenta de Silva (2016) é totalmente web. Um ponto a ser observado é que mesmo que a ferramenta de Campos (2015) foi desenvolvida na plataforma PhoneGap, foram feitos testes apenas na plataforma iOS.

Uber (2017) e Campos (2015) também utilizam a geolocalização para encontrar seus usuários, porém utilizam ferramentas diferentes. O Uber (2017) utiliza tanto para que o cliente e o motorista possam se encontrar, quanto para guiar ambos para sua rota. Campos (2015) permite o compartilhamento de localização entre os usuários A ferramenta de Silva (2016) não utiliza geolocalização. O Uber (2017) e o trabalho de Silva (2016) permitem avaliação dos clientes, isso porque fornece mais segurança ao cliente, podendo pedir serviços apenas de prestadores de serviços bem avaliados. Uma característica em especial do Uber (2017) é que ele possui um chat para interação de seus usuários, enquanto os demais trabalhos não possuem.

3 DESENVOLVIMENTO

Neste capítulo é apresentado o detalhamento do desenvolvimento do aplicativo desenvolvido, assim como suas especificações. A primeira seção apresenta um levantamento das informações. A segunda seção especifica o projeto desenvolvido, apresentando os Requisitos Funcionais (RF), Requisitos Não Funcionais (RNF), diagrama de Casos de Uso (UC), matriz de rastreabilidade dos requisitos funcionais, diagrama de classes, diagrama de componentes e diagrama de atividades. A terceira seção demonstra as técnicas e ferramentas utilizadas no desenvolvimento do trabalho, assim como trechos de códigos-fontes relevantes e sua operacionalidade. Por fim, a quarta seção apresenta um comparativo entre os trabalhos correlatos e os resultados obtidos com o desenvolvimento.

3.1 LEVANTAMENTO DE INFORMAÇÕES

Este trabalho apresenta a construção de um aplicativo web, responsivo e móvel, utilizando o modelo de negócios de consumo colaborativo. Este trabalho é voltado para prestadores de serviço autônomos que procuram uma alternativa para anunciar seu serviço e para consumidores que procuram por esses serviços. O conteúdo do aplicativo só pode ser acessado por usuários autenticados. O aplicativo permite criar uma rede de consumo colaborativo entre os usuários, pois qualquer usuário autenticado pode anunciar seus serviços e solicitar um serviço de outro usuário. No aplicativo é possível procurar por serviços locais por meio de geolocalização, visualizar as informações do usuário prestador de serviço e criar uma solicitação de serviço. Para o usuário que optar por prestar um serviço, é possível cadastrar um novo serviço e administrar suas solicitações de serviço. O aplicativo disponibiliza um chat para comunicação entre o usuário cliente e o usuário prestador de serviço. Quando o usuário prestador de serviço conclui a solicitação de serviço, o usuário cliente pode então avaliar o serviço.

O trabalho desenvolvido possui um módulo servidor, que é responsável pela comunicação com o banco de dados, por processar as regras de negócio e disponibilizar as informações para o aplicativo através de requisições Web Service. Optou-se por desenvolver um aplicativo web utilizando algumas técnicas de Progressive Web App (PWA) pois pode ser utilizado tanto no Android, quanto no iOS, não precisa ser instalado e pode utilizar funções prontas do navegador para a navegação no mapa do aplicativo.

3.2 ESPECIFICAÇÃO

Esta seção apresenta a especificação do sistema desenvolvido, formada pelos requisitos funcionais e não funcionais, bem como pelos diagramas desenvolvidos especificados em subseções. O Quadro 2 apresenta os Requisitos Funcionais (RF) do aplicativo desenvolvido e seu vínculo com o caso de uso associado, sendo que o Diagrama de Casos de Uso pode ser visualizado na Figura 11. Por sua vez, o Quadro 3 lista os Requisitos Não Funcionais (RNF) do aplicativo desenvolvido.

Quadro 2 - Requisitos Funcionais

Requisitos Funcionais	**Caso de Uso**
RF01: O aplicativo deverá permitir manter usuários.	UC01
RF02: O aplicativo deverá permitir manter serviços.	UC08
RF03: O aplicativo deverá permitir buscar profissionais	UC02
RF04: O aplicativo deverá mostrar os profissionais do serviço escolhido em um mapa para o cliente.	UC02
RF05: O aplicativo deverá disponibilizar informações do profissional para o cliente.	UC03
RF06: O aplicativo deverá permitir que o usuário crie solicitações de serviço.	UC06
RF07: O aplicativo deverá permitir que o usuário cancele solicitações de serviço.	UC06
RF08: O aplicativo deverá notificar ao profissional que existe um cliente precisando do seu serviço.	UC06
RF09: O aplicativo deverá disponibilizar informações do cliente para o profissional.	UC05
RF10: O aplicativo deverá permitir que o profissional aceite solicitações de serviço.	UC06
RF11: O aplicativo deverá permitir que o profissional recuse solicitações de serviço.	UC06
RF12: O aplicativo deverá permitir que o profissional conclua solicitações de serviço.	UC06
RF13: O aplicativo deverá permitir a troca de mensagens entre cliente e prestador de serviço.	UC04
RF14: O aplicativo deverá notificar o usuário sobre a mudança na situação das solicitações de serviço.	UC06
RF15: O aplicativo deverá permitir que o cliente avalie o profissional.	UC07

Fonte: elaborado pelo autor.

Quadro 3 - Requisitos não funcionais

Requisitos Funcionais
RNF01: O módulo servidor deverá ser implementado utilizando a plataforma NodeJS.
RNF02: O aplicativo deverá ser implementado utilizando o Quasar Framework.
RNF03: O módulo servidor deverá utilizar o banco de dados MySQL.
RNF04: O sistema deverá ser acessado através de um navegador que suporte HTML5.
RNF05: O aplicativo deverá controlar as funcionalidades por perfil de acesso.
RNF06: O aplicativo deverá utilizar a API do Google Maps para mostrar os usuários em um mapa.
RNF06: O aplicativo deverá utilizar o HTML5 para pegar a localização atual do usuário.
RNF07: O módulo servidor deverá enviar e-mail para os usuários como forma de notificação.

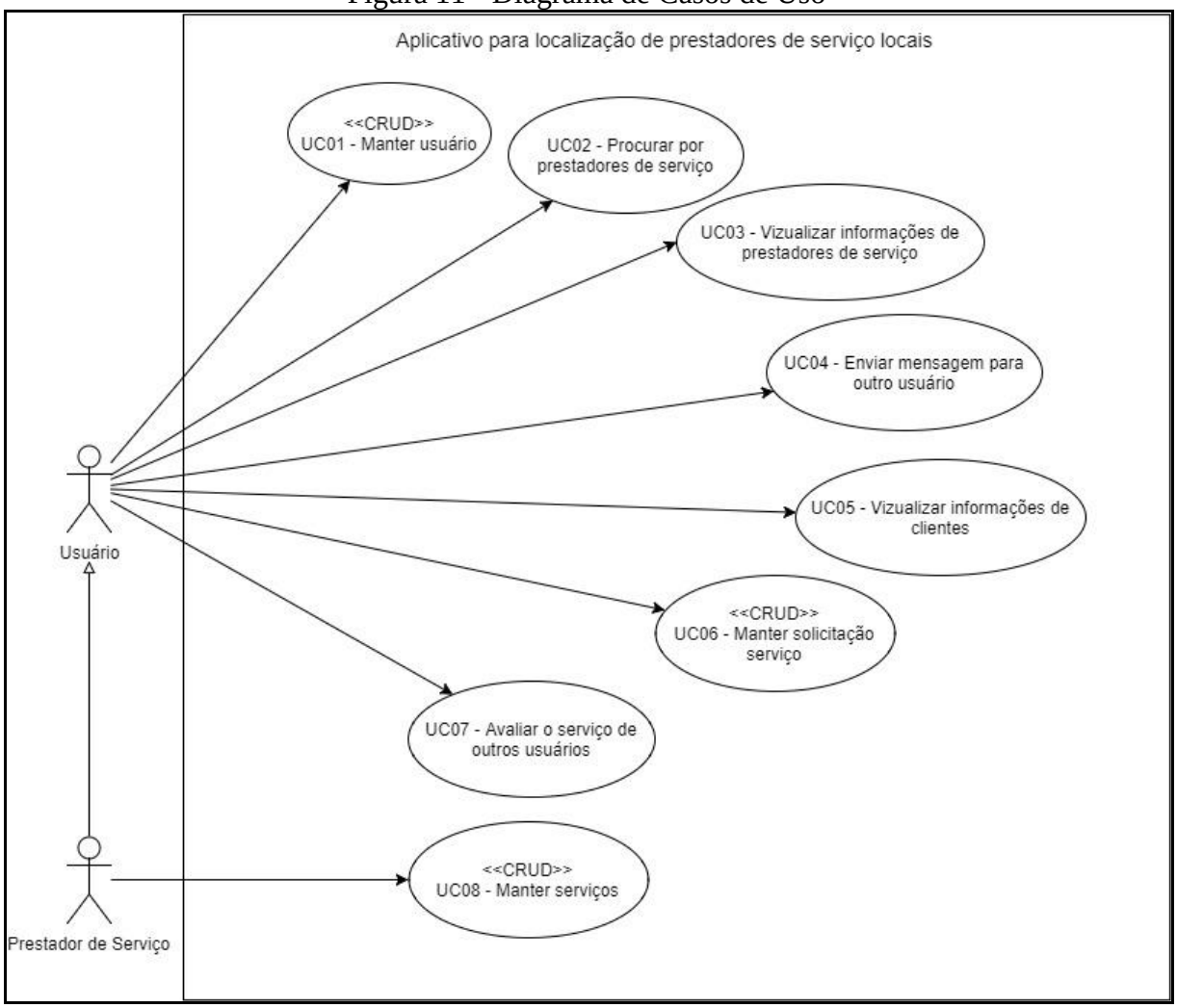
Fonte: elaborado pelo autor.

3.2.1 Diagrama de casos de uso

A Figura 11 apresenta o diagrama de casos de uso do aplicativo desenvolvido. Os atores apresentados no diagrama e presentes no sistema são: `usuário` e `prestador de`

`serviço`. O `prestador de serviço` é uma especialização do `usuário` e pode realizar as mesmas ações que ele. O `usuário` pode manter sua conta, procurar por um serviço, enviar e receber mensagens para o `prestador de serviço`, solicitar e avaliar serviços. Quando o `usuário` cria um serviço, ele se torna um `prestador de serviço`. Ele pode então manter seus serviços, aceitar, recusar e concluir solicitações de serviço. O Apêndice A apresenta o caso de uso 02 e 06 com detalhes, por meio da descrição de seus cenários.

Figura 11 - Diagrama de Casos de Uso

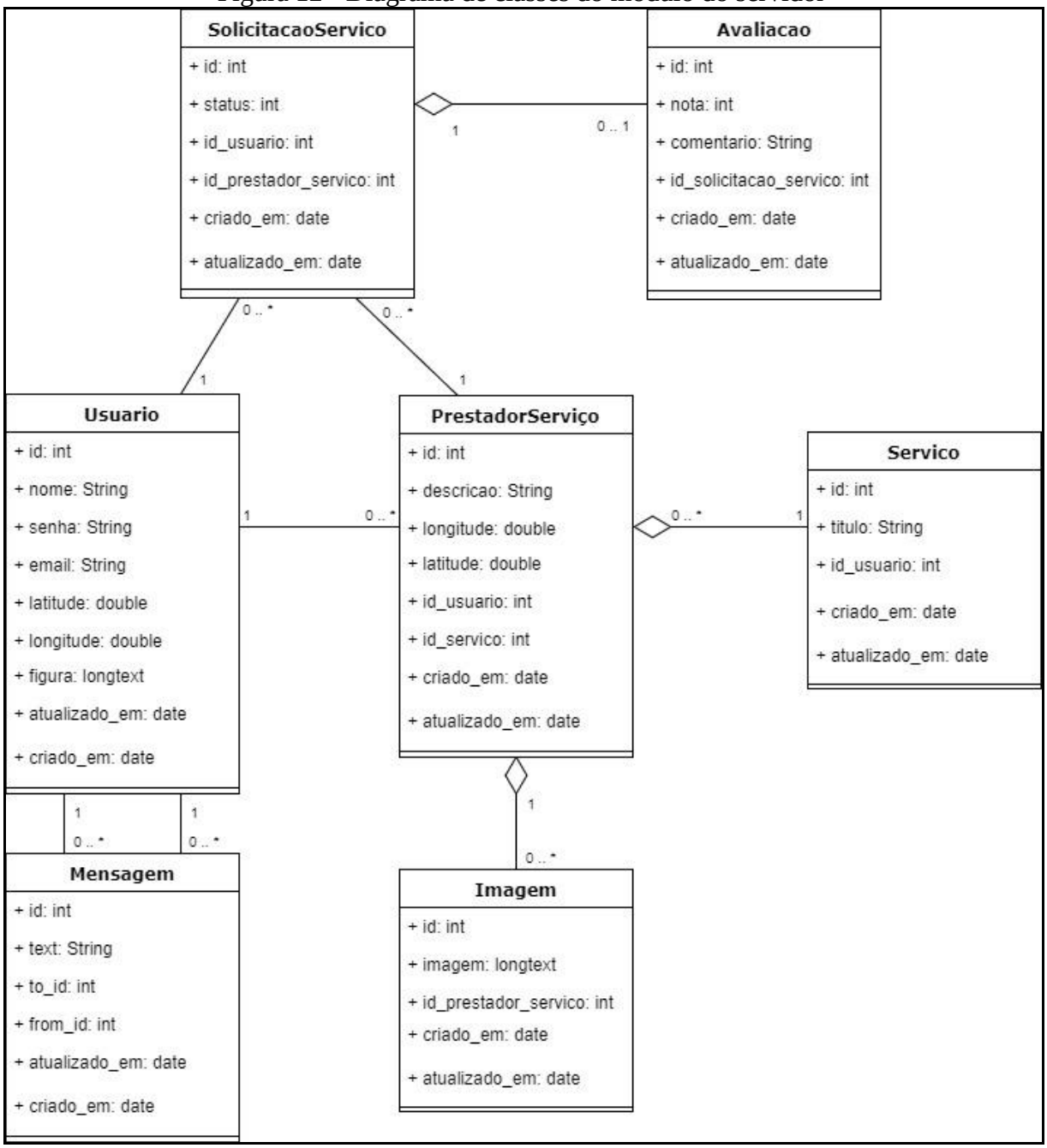


Fonte: elaborado pelo autor.

3.2.2 Diagrama de Classes

Esta seção apresenta os diagramas de classes relacionado a parte do servidor, do sistema desenvolvido. A Figura 12 ilustra o diagrama de classes dos modelos do módulo do servidor.

Figura 12 - Diagrama de classes do módulo do servidor



Fonte: elaborado pelo autor.

A seguir é apresentada uma breve explicação das classes demonstradas na Figura 12 para o trabalho desenvolvido:

a) `Usuario` possui as informações dos usuários do aplicativo e se relaciona com as classes `PrestadorServico`, `SolicitacaoServico` e `Mensagem`;

b) `PrestadorServico` possui as informações do serviço que o usuário está ofertando e se relaciona com `SolicitacaoServico` e `Servico`;

c) `Imagem` possui as imagens referentes ao serviço prestado;

d) `Servico` possui os serviços que já foram criados no sistema;

e) `SolicitacaoServico` possui as informações das solicitações de serviços criadas pelos usuários e se relaciona com `Avaliacao`;

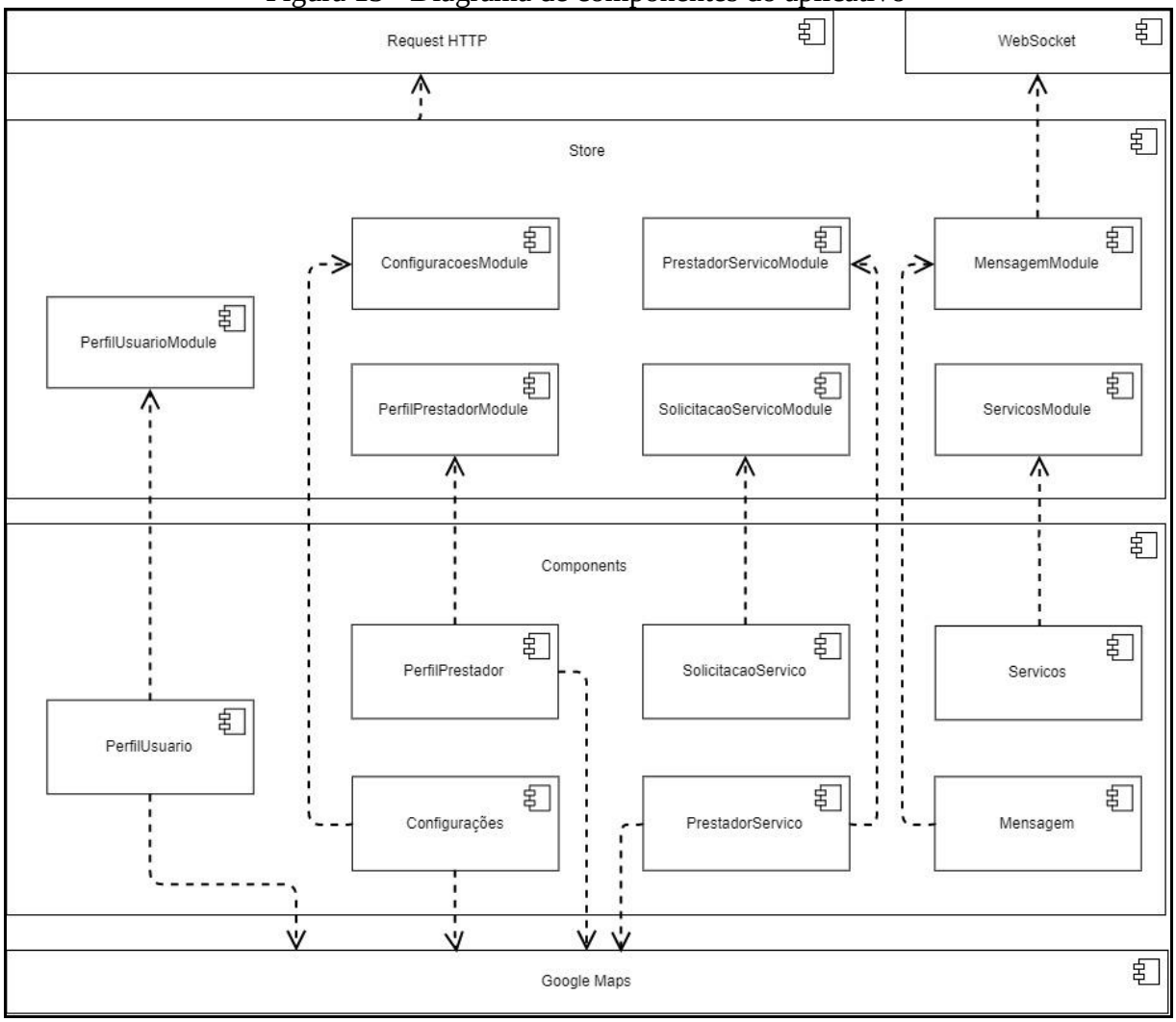
f) `Avaliacao` possui as informações das avaliações que os usuários geram aos serviços prestados;

g) `Mensagem` possui as mensagens privadas que são trocadas entre os usuários.

3.2.3 Diagrama de componentes do aplicativo

Nesta seção é apresentado o diagrama de componentes relacionado a interface gráfica desenvolvida. A Figura 13 representa o relacionamento entre as camadas envolvidas na interface gráfica, relacionando com suas principais telas.

Figura 13 - Diagrama de componentes do aplicativo

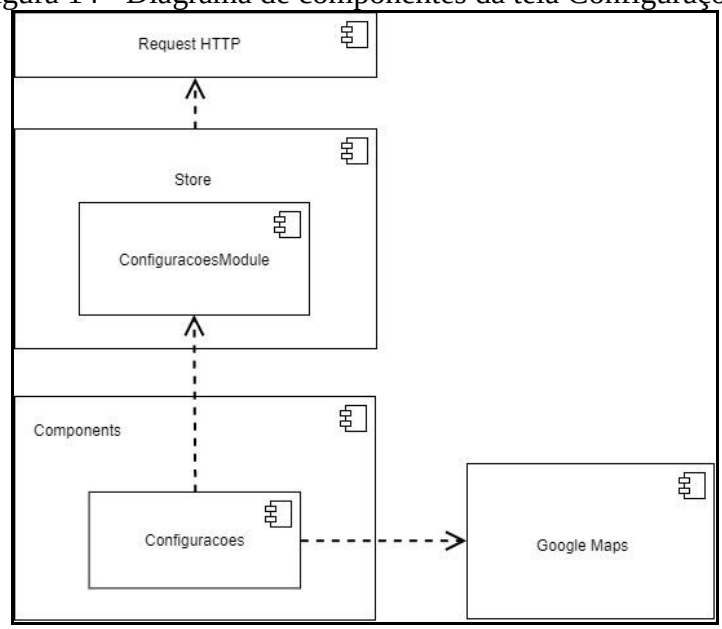


Fontes: elaborado pelo autor.

Conforme é apresentado no diagrama, ele é composto por cinco componentes macros, sendo: `Google Maps`, `Components`, `Store`, `WebSocket e Request HTTP`. O componente `Components` engloba as telas do aplicativo. Já o componente `Store` é dividido em módulos,

sendo que cada módulo contém os dados e as regras de negócio dos componentes de tela. Algumas telas utilizam requisições *WebSockets* que são feitas da camada `Store` para o componente `WebSockets` e outras fazem requisições para a API `Google Maps` a partir da própria tela. Já o componente `Request HTTP` é responsável por enviar as requisições HTTP para o módulo servidor. Para exemplificar melhor a funcionalidade das telas, a Figura 14 demonstra o diagrama de componentes da tela `Configurações`.

Figura 14 - Diagrama de componentes da tela Configurações



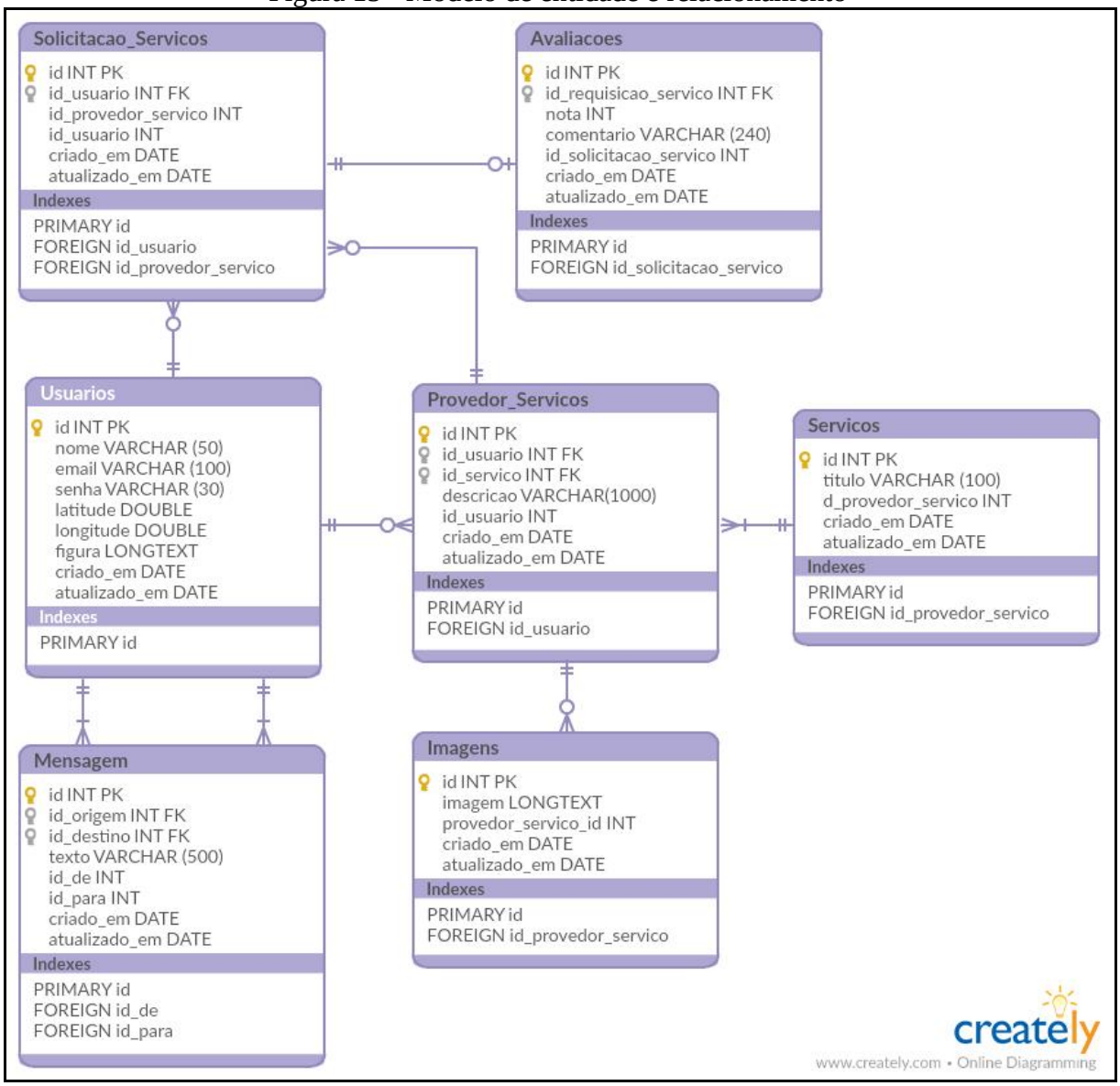
Fonte: elaborado pelo autor.

Nesse exemplo, tem-se o componente `Configuracoes`, dentro do componente `Components`, que é a tela que mostra os dados do usuário autenticado. O módulo `ConfiguracoesModule`, localizado no componente `Store`, possui os dados do usuário autenticado e as regras de negócio para carregar e salvar os dados, trocar senha e excluir a conta. Como o componente `Configuracoes` possui um mapa, ele utiliza a API Google Maps para criar o mapa na tela, que fica no componente `Google Maps` do diagrama. Este exemplo se aplica nas principais telas do sistema conforme Figura 13.

3.2.4 Modelo de entidade e relacionamento

Nesta seção é apresentado o Modelo Entidade e Relacionamento (MER), com uma breve explicação das entidades demonstradas. A Figura 15 representa as entidades e seus relacionamentos por meio da modelagem de dados.

Figura 15 - Modelo de entidade e relacionamento



Fonte: elaborado pelo autor.

A seguir é apresentada uma breve explicação das entidades demonstradas na Figura 15 para o trabalho desenvolvido:

a) a entidade `Usuarios` é responsável por armazenar os dados dos usuários registrados no aplicativo;
b) a entidade `ProvedorServicos` é responsável por armazenar os dados dos provedores de serviço;
c) a entidade `Servicos` é responsável por armazenar os títulos de serviços cadastrados no sistema;
d) a entidade `Imagens` é responsável por armazenar as imagens referentes ao serviço oferecido pelo prestador de serviços;
e) a entidade `SolicitacaoServicos` é responsável por armazenar as solicitações de

serviços criadas pelos usuários cadastrados no aplicativo;

f) a entidade `Avaliacoes` é responsável por armazenar as informações referentes as avaliações dos usuários perante os serviços oferecidos;
g) a entidade `Mensagem` é responsável por armazenar a troca de informações dos usuários pelo chat.

3.3 IMPLEMENTAÇÃO

Nesta seção são apresentadas as técnicas e ferramentas utilizadas para o desenvolvimento do aplicativo, assim como a sua operacionalidade. Também é descrito o desenvolvimento do aplicativo, apresentando trechos de códigos-fontes das principais rotinas desenvolvidas.

3.3.1 Técnicas e ferramentas utilizadas

Esta seção apresenta as técnicas e ferramentas utilizadas para a construção do aplicativo. Para o desenvolvimento desde trabalho foram utilizadas as ferramentas:

a) Atom Editor de texto;
b) MySQL como banco de dados;
c) MySQL Workbench como ferramenta para manipular o banco de dados;
d) Draw.io como ferramenta para criação dos diagramas
e) Creately como ferramenta para criação do MER;
f) Node.js como plataforma de desenvolvimento;
g) Quasar Framework como *framework* para desenvolvimento do aplicativo web;
h) VueJS como biblioteca para manipulação de telas;
i) SequelizeJS como *framework* para o Mapeamento Objeto-Relacional (ORM);
j) ExpressJS como *framework* para tratar requisições HTTP;
k) Postman como ferramenta para testes das requisições HTTP no módulo servidor;
l) SocketIO como *framework* para tratar eventos WebSocket;
m) Google Maps API para uso de recursos de geolocalização;
n) vue2-google-maps para integrar os recursos da API do Google Maps com o VueJS.

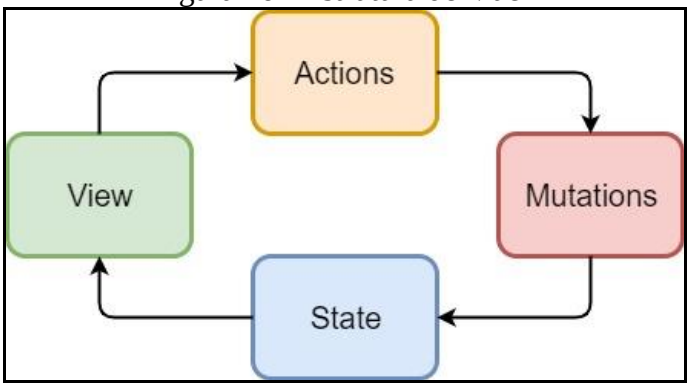
O módulo servidor foi desenvolvido com a plataforma Node.js que utiliza a linguagem de programação Javascript e o motor Javascript V8 do Google Chrome. O Node.js é escalável e utilizado na criação de servidores web, aplicações que lidam com recursos do sistema operacional e até ferramentas para auxiliar no desenvolvimento de *softwares*, como editores de texto. A plataforma utiliza o modelo de programação orientada a eventos com um modelo

de entrada/saída direcionada a eventos não bloqueantes. O *front-end* do aplicativo foi desenvolvido de forma responsiva para ser executado como web *app* em um navegador de dispositivo móvel. Assim, ele pode ser utilizado em qualquer plataforma móvel, como Android, iOS e Windows Phone, sem precisar ser instalado.

Para o desenvolvimento do aplicativo, foi utilizado o Quasar Framework que contém uma série de componentes prontos para a criação das telas. Este *framework* funciona com VueJS, uma biblioteca Javascript para o desenvolvimento de componentes reativos para interfaces web modernas. Juntamente com o VueJS, foram utilizadas as ferramentas VueRouter, para roteamento das páginas e Vuex, para o controle compartilhado dos dados através dos componentes de tela.

O Vuex utiliza a prática "Fonte Única da Verdade" (Single Source of Truth), que estrutura os dados de forma a ficarem centralizados no aplicativo. Assim, todos os dados podem ser salvos em um único local e ficam disponíveis para todos os componentes. A Figura 16 apresenta a estrutura do Vuex (VUE, 2017).

Figura 16 - Estrutura do Vuex



Fonte: elaborado pelo autor.

Como a Figura 16 ilustra, o Vuex possui um fluxo de dados unidirecional composto por quatro camadas. Os dados do aplicativo ficam na camada `State` para serem utilizados pela `View`, como por exemplo as informações do usuário autenticado. As `Actions` são os eventos disparados pela `View` e é nessa camada que as requisições HTTP para o módulo servidor são feitas. Para que os dados possam então ser persistidos na camada `State`, as `Actions` precisam disparar as `Mutations` que por sua vez efetuarão as alterações dos dados.

Para o uso da geolocalização, foi utilizado o *framework* vue2-google-maps que integra os recursos da API do Google Maps com o VueJS. Este *framework* contém os componentes prontos para a criação do mapa na tela do aplicativo e utilização de seus recursos. Para buscar a localização atual do usuário foi utilizada a API de geolocalização do HTML5. O Quadro 4

apresenta a relação dos navegadores com suas respectivas versões que suportam a API, sendo o Google Chrome, Internet Explorer, Mozilla Firefox, Safari e Opera.

Quadro 4 - Navegadores que suportam a API de geolocalização do HTML5

Navegadores	**Versões**
Google Chrome	5.0, 4.9 (http), 50.0 (https)
Internet Explorer	9.0
Mozilla Firefox	3.5
Safari	5.0
Opera	16.0

Fonte: W3schools (2017).

Uma observação a se fazer é que a partir da versão 50.0 do navegador Google Chrome, a API só funciona em sites com protocolo HTTPS. Para utilização de *WebSockets* foi utilizada a biblioteca `SocketIO` que é baseada em eventos e possibilita uma comunicação bidirecional entre o aplicativo e o servidor. Esta biblioteca é escrita em Javascript e por isso pode rodar no servidor em conjunto com o Node.js. No aplicativo foi utilizado o protocolo `WebSockets` para o envio dos eventos para o servidor.

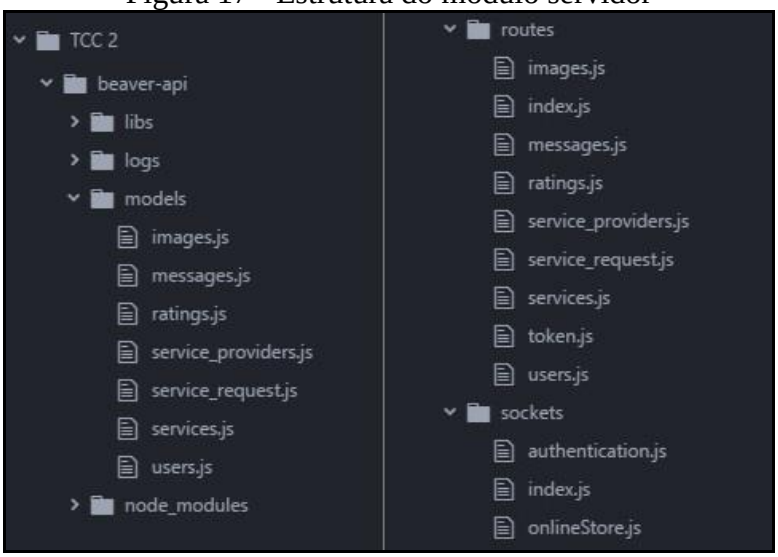
3.3.2 Desenvolvimento

Nesta seção é apresentado o desenvolvimento do trabalho, expondo os principais trechos de códigos-fontes do módulo servidor e do aplicativo, utilizando exemplos do trabalho desenvolvido.

3.3.2.1 Módulo servidor

A estrutura principal do módulo servidor é separada em três partes. A Figura 17 mostra a estrutura utilizada no módulo servidor.

Figura 17 - Estrutura do módulo servidor



Fonte: elaborado pelo autor.

As partes dessa estrutura estão divididas pelas pastas `models`, `routes` e `sockets`. A pasta `models` contém os modelos de classes que representam as tabelas do banco de dados. Os modelos são definidos com o *framework* `Sequelize`. O Quadro 5 apresenta a constante `Services`, que referencia a tabela `Servicos` do banco de dados.

Quadro 5 - Modelo Serviços

```
1  module.exports = (sequelize, DataType) => {
2    const Services = sequelize.define("Services", {
3      id: {
4        type: DataType.INTEGER,
5        primaryKey: true,
6        autoIncrement: true
7      },
8      title: {
9        type: DataType.STRING(100),
10       unique: true,
11       allowNull: false,
12       validate: {
13         notEmpty: true
14       }
15     }
16   }, {
17     classMethods: {
18       associate: (models) => {
19         Services.hasMany(models.ServiceProviders);
20       }
21     }
22   });
23   return Services;
24 };
```

Fonte: elaborado pelo autor.

A função `sequelize.define` da linha 2 cria a tabela de serviços no banco de dados caso ela não exista. O nome da tabela e suas colunas são passadas por parâmetro, assim como suas associações. O Quadro 5 mostra as colunas da tabela Services (linha 2): `id` (linha 3) e `title` (linha 8) e seus atributos, como por exemplo: `type` (linhas 3 e 9), `primaryKey` (linha 5), `unique` (linha 10) e `allowNull` (linha 11). As associações são definidas a partir do atributo `classMethods` (linha 17). A função `associate` (linha 18) define os relacionamentos da tabela de serviços com outras tabelas. A cardinalidade é definida pela função `hasMany` (linha 19) que define que o modelo `Services` é referenciado no modelo `ServiceProviders` que representa a tabela de prestadores de serviços. Outro exemplo de cardinalidade é a função `belongsTo` que é demonstrada na linha 15 da Quadro 6 apresentada a seguir.

Quadro 6 - Cardinalidades do modelo Messages

```
12    }, {
13      classMethods: {
14        associate: (models) => {
15          Messages.belongsTo(models.Users,  {
16            foreignKey: 'to_id'
17          });
18          Messages.belongsTo(models.Users, {
19            foreignKey: 'from_id'
20          });
21        }
22      }
23    });
24    return Messages;
25  };
```

Fonte: elaborado pelo autor.

A função `belongsTo` define que o modelo `Messages` referencia o modelo `Users` duas vezes (linhas 15 e 18). Os nomes das colunas são definidos como `to_id` (linha 16) e `from_id` (linha 19) através do atributo `foreignKey`, passado por parâmetro. Quando o atributo `foreignKey` não é utilizado, o nome da coluna é definido com um padrão próprio do *framework* Sequelize. Por exemplo, no Quadro 5, a coluna criada para referenciar o modelo `Services` na tabela `ServiceProviders` foi `service_id`.

A pasta `routes` do módulo servidor é responsável por lidar com as requisições HTTP. O modulo servidor recebe essas requisições por meio de rotas que utilizam o design de arquitetura Representational State Transfer (REST). O Quadro 7 apresenta as rotas para buscar, alterar e deletar um prestador de serviço.

Quadro 7 - Rotas para buscar, alterar e deletar prestadores de serviço

```
 97  app.route("/serviceProvider/:id")
 98    .all(app.auth.authenticate())
 99    .get((req, res) => {
100 ›    ServiceProviders.findOne({=
111      .then(result => {
112        if (result) {
113          res.json(result);
114        } else {
115          res.sendStatus(404);
116        }
117      })
118      .catch(error => {
119        res.status(412).json({msg: error.message});
120      });
121
122    })
123    .put((req, res) => {
124 ›    ServiceProviders.update(req.body, { where: {=
127      .then(result => res.sendStatus(204))
128      .catch(error => {
129        res.status(412).json({msg: error.message});
130      });
131    })
132    .delete((req, res) => {
133 ›    ServiceProviders.destroy({ where: {=
136      .then(result => res.sendStatus(204))
137      .catch(error => {
138        res.status(412).json({msg: error.message});
139      });
140    });
```

Fontes: elaborado pelo autor.

No Quadro 7 é apresentado o código-fonte com as rotas do prestador de serviço com a URL de entrada `/serviceProvider/:id`, na linha 97. Esta URL realiza autenticação do usuário e espera uma requisição HTTP com um dos seguintes métodos: `GET` (linha 99), `PUT` (linha 123) e `DELETE` (linha 132). A requisição com método `GET` é utilizado para a buscar os registros da tabela `ProvedorServicos` através da função `findOne` (linha 100). A requisição com método `PUT` é utilizado para a atualizar um registro na tabela `ProvedorServicos` através da função `update` (linha 124). Por fim, a requisição com método `DELETE` é utilizado para a deletar um registro da tabela `ProvedorServicos` pela função `destroy` (linha 133).

As funções `findOne`, `update` e `destroy` pertencem ao *framework* Sequelize. Além da função `findOne` que retorna o primeiro registro da busca, existem outras opções para buscar registros do banco de dados, como por exemplo: `findById` para buscar o registro pelo `id` e `findAll` que busca uma lista de registros. O Quadro 8 e o Quadro 10 apresentam exemplos das funções `findById` e `findAll` respectivamente.

Quadro 8 - Rota para buscar dados do usuário

```
48  app.route("/user/:user_id")
49    .all(app.auth.authenticate())
50    .get((req, res) => {
51      Users.findById(req.params.user_id, {
52        attributes: ["id", "name", "email", "latitude", "longitude", "picture", "updated_at" ],
53        include: [{
54          model: app.db.models.ServiceProviders,
55          attributes: ["id"],
56          include: [{
57            model: app.db.models.Services,
58            attributes: ["title"]
59          }]
60        }]
61      })
62      .then(result => res.json(result))
63      .catch(error => {
64        res.status(412).json({msg: 'Usuário não encontrado.'});
65      });
66    });
```

Fonte: elaborado pelo autor.

A função `findById` recebe como parâmetro o `id` do registro e um JSON contendo os dados para a busca no banco de dados. No Quadro 8, é passado por parâmetro o `id` do usuário (linha 51), os campos da tabela que serão retornados, através do atributo `attributes` (linha 52), além do atributo `include` (linha 53). O atributo `include` é responsável pelos `joins` com outros modelos. No caso do Quadro 8, é feito `join` com o modelo `ServiceProviders` (linha 54), buscando somente o campo `id` (linha 55). O modelo `ServiceProviders` por sua vez, faz `join` com o modelo `Services` (linha 57), buscando seu título (linha 58). O resultado dessa busca é apresentado no Quadro 9.

Quadro 9 - Resultado da busca de usuário

```
{
    "id": 7,
    "name": "Pedro Junior da Silva",
    "email": "pedro.junior.silva@gmail.com",
    "latitude": -26.9078152307243,
    "longitude": -49.06665453068848,
    "picture": null,
    "updated_at": "2017-11-07T02:11:37.000Z",
    "ServiceProviders": [
        {
            "id": 4,
            "Service": {
                "title": "Pedreiro"
            }
        },
        {
            "id": 5,
            "Service": {
                "title": "Pintor"
            }
        }
    ]
}
```

Fonte: elaborado pelo autor.

Quadro 10 - Rota para buscar prestadores de serviço

```
app.route("/serviceProviders/:service_id")
  .all(app.auth.authenticate())
  .get((req, res) => {
    ServiceProviders.findAll({
      attributes:  ["id", "description", "latitude", "longitude", "updated_at",
                   [Sequelize.literal('(SELECT ROUND(avg(Ratings.rate), 2)' +
                                      '  FROM Ratings,' +
                                      '       ServiceRequests' +
                                      ' WHERE ServiceRequests.id = Ratings.service_request_id' +
                                      '   AND ServiceRequests.service_provider_id = ServiceProviders.id' +
                                      ' GROUP BY ServiceRequests.service_provider_id)'),'rating']],
      where: {
        service_id: req.params.service_id,
        user_id: {
          $ne: req.user.id
        }
      },
      include: [{
        model: Users,
        attributes: ["name", "picture"]
      }]
    })
    .then(result => res.json(result))
    .catch(error => {
      res.status(412).json({msg: error.message});
    });
  });
```

Fonte: elaborado pelo autor.

O Quadro 10 apresenta uma consulta a prestadores de serviços utilizando a função `findAll`. Esta função retorna uma lista de registros e recebe como parâmetro um JSON contendo os dados para a busca no banco de dados. Para esta consulta foi utilizado o atributo `where` (linha 87) para estabelecer as regras para a busca no banco de dados. Para buscar a avaliação geral do prestador de serviço foi utilizada função `literal` (linha 74) do Sequelize.

Essa função permite fazer uma busca a partir de uma *String* com o comando escrito em MySql.

Por fim, a pasta `sockets` é responsável pelos eventos *WebSockets* enviados pelo cliente e por manter uma estrutura de dados dos usuários *online*. O arquivo `index.js` é responsável por inicializar o servidor. Ao fazer isso, todos os pacotes do módulo servidor são carregados e os modelos são sincronizados com o banco de dados.

3.3.2.2 Aplicativo Web

O aplicativo web foi desenvolvido utilizando o *framework* Quasar. Este *framework* utiliza a biblioteca VueJS, que permite ao desenvolvedor escrever códigos para componentes reativos para interface web. Os componentes do VueJS contêm o HyperText Markup Language (HTML), Cascading Style Sheets (CSS) e Javascript em sua estrutura, representados pelas *tags* `template`, `style` e `script` respectivamente. O Quadro 11 apresenta o modelo de um componente que a biblioteca disponibiliza como exemplo.

Quadro 11 - Modelo de um componente VueJS

```
1  <template>
2    <div></div>
3  </template>
4
5  <script>
6  export default {
7    data () {
8      return {}
9    }
10 }
11 </script>
12
13 <style>
14 </style>
```

Fonte: elaborado pelo autor.

A *tag* `template` (linha 1) possui a marcação HTML do componente. A *tag* `script` (linha 5) contém os códigos Javascript responsáveis pelo comportamento do componente. Por fim, na *tag* `style` (linha 13) estão os estilos do componente, sendo os códigos CSS. Uma característica dos componentes VueJS é que podem ser utilizados como `tags` customizadas em outros componentes. O Quadro 12 mostra a *tag* customizada `SideMenu`.

Quadro 12 – Tag SideMenu

a)

```
1  <template lang="html">
2    <q-drawer ref="leftDrawer" swipe-only>
3      <div class="toolbar light">
4        <q-toolbar-title :padding="1">
5            Opções
6        </q-toolbar-title>
7      </div>
8      <div class="list no-border platform-delimiter">
38   </q-drawer>
39 </template>
```

b)

```
11
12     <sideMenu></sideMenu>
13
```

Fonte: elaborado pelo autor.

O Quadro 12 (a) apresenta parte do código-fonte do componente `SideMenu.vue`, sendo apresentado entre a linha 1 e 39. Este componente é o menu lateral do aplicativo e está presente em todas as telas do aplicativo, que está descrito na Figura 20 da seção 3.3.3. O Quadro 12 (b) mostra a *tag* `sideMenu` (linha 12) sendo utilizada em outro componente. O *framework* Quasar possui *tags* customizadas próprias para poder gerar um leiaute padrão em todas as telas. Um exemplo é a *tag* `q-drawer` (linha 2) presente no Quadro 12 (a) que gera o menu lateral do aplicativo. Além das *tags*, o *framework* também contém estilos próprios que são utilizados através das classes das *tags*. No Quadro 12 (a) há uma `div` com as classes `list`, `no-border` e `platform-delimiter` para formatar o menu, conforme pode-se observar na linha 8 respectivamente. O Quadro 13 mostra uma parte do código-fonte do componente `Setup.vue`.

Quadro 13 - Componente Setup.vue

```
36  <template>
37    <q-layout>
38      <div slot="header" class="toolbar">
39        <button class="hide-on-drawer-visible" @click="$refs.sideMenu.open()">
40          <i>menu</i>
41        </button>
42        <q-toolbar-title :padding="1">
43          Configurações
44        </q-toolbar-title>
45      </div>
46
47      <sideMenu ref="sideMenu"></sideMenu>
48      <topTab slot="navigation"></topTab>
49
50      <div id="setupForm" class="layout-view">
103     <q-modal ref="layoutModal">
104       <q-layout>
105         <div slot="header" class="toolbar">
116
117         <div class="layout-view">
142       </q-layout>
143     </q-modal>
144     <q-modal ref="minimizedModal" class="minimized" :content-css="{padding: '15px'}">
161     </q-modal>
162     <q-modal ref="confirmModal" class="minimized" :content-css="{padding: '15px'}">
174     </q-modal>
175   </q-layout>
176 </template>
```

Fonte: elaborado pelo autor.

No Quadro 13 é apresentada a *tag* `q-layout` (linha 37) que define o leiaute da tela do aplicativo. A *tag* `q-toolbar-title` contém a formatação padrão para mostrar o título da tela. A *tag q-modal* (linhas 103, 144 e 162) cria um modal já formatado na tela do aplicativo. Através das classes dessa *tag* é possível definir seu tamanho, de qual lado o modal irá aparecer e até sua animação. Por exemplo, no Quadro 13, o `q-modal` da linha 144 aparece em tamanho pequeno no centro da tela, pois possui a classe `minimized`. Quando não é passada uma classe para essa *tag* como na linha 103, por padrão o modal aparece em tela-cheia. O Quadro 14 apresenta o trecho do código-fonte com os campos do componente `ServiceProvider.vue`.

Quadro 14 – Campos do componente ServiceProvider.vue

```
61          <div class="floating-label">
62            <textarea class="full-width" style="height: 100px;" :value="serviceProviderState.description" @change="updateDescription"></textarea>
63            <label>Descrição</label>
64          </div>
65          <div class="stacked-label">
66            <textarea class="full-width" readonly v-model="address" @click="openMap"></textarea>
67            <label>Localização</label>
68          </div>
69        </div>
70        <center>
71          <div class="full-width" style=" padding-top: 10px;">
72            <button type="submit" class="primary small outline" @click="saveServiceProvider">Alterar Dados</button>
73          </div>
74        </center>
75      </div>
76    </form>
77  </div>
```

Fonte: elaborado pelo autor.

O Quadro 14 apresenta os campos `Descrição` (linhas 62 e 63) e `Localização` (linhas 66 e 67), seguidos do botão `Alterar Dados` (linha 72). O campo `Descrição` possui o mapeamento do seu valor na camada `State` do VueX. Seu atributo `value` recebe o valor que está armazenado enquanto a ação `change` ativa a função `updateDescrition` que ativa uma `action` para mudar o valor desse campo na camada `State`. O campo `Localização` possui a descrição da localização do provedor de serviço. Este campo possui na ação `click` a função `openMap` que abre um modal com o mapa para o usuário escolher a localização. O botão `Alterar Dados` é responsável por ativar a `action` para enviar os dados que estão na camada `State` para o módulo servidor. O Quadro 15 mostra uma parte do `script` do componente de configurações do prestador de serviço.

O *script* do componente possui o objeto `computed` (linha 213) que utiliza o método `mapState` para mapear os valores da camada `State`. Assim eles podem ser utilizados no `template` do componente. O objeto `methods` (linha 241) possui as funções utilizadas na tela e mapeia as `actions` para a camada `State`. O objeto `components` (linha 208) contém os componentes que foram importados para serem usados na tela.

Quadro 15 - Trecho do script do componente de configurações do prestador de serviço

```
206  export default {
207    components: {
208      'googleMap': VueGoogleMaps.Map,
209      'MapMarker': VueGoogleMaps.Marker,
210      sideMenu
211    },
212    computed: mapState({
213      serviceProviderState: state => state.serviceProviderModule
214    }),
215 ›  data () {…
230 ›  created () {…
240    methods: {
241      ...mapActions({
242        updateId: UPDATE_ID,
243        updateDescription: UPDATE_DESCRIPTION,
244        updateLocation: UPDATE_LOCATION,
245        performLoadServiceProvider: PERFORM_LOAD_SERVICE_PROVIDER,
246        performLoadServiceRequests: PERFORM_LOAD_SERVICE_REQUESTS,
247        performLoadMyProviderRating: PERFORM_LOAD_PROVIDER_RATING,
248        performSaveServiceProvider: PERFORM_SAVE_SERVICE_PROVIDER,
249        performRemoveServiceProvider: PERFORM_REMOVE_SERVICE_PROVIDER,
250        performLoadImages: PERFORM_LOAD_IMAGES,
251        performUploadImage: PERFORM_UPLOAD_IMAGE,
252        performRemoveImage: PERFORM_REMOVE_IMAGE,
253        clearState: CLEAR_STATE
254      }),
```

Fonte: elaborado pelo autor.

Na linha 20 está o componente `googleMap` do *framework* `vue2-google-maps` que contém o mapa da API do Google Maps. O componente `MapMarker` (linha 2010) representa os marcadores com as localizações do mapa. O Quadro 16 apresenta a utilização dos componentes `googleMap` e `MapMarker`.

Quadro 16 – Utilização do componente do Google Maps

```
110    <google-map
111      :center="center"
112      :zoom="14"
113      :options="{
114        disableDefaultUI: true,
115        panControl: false,
116        mapTypeControl: false,
117        streetViewControl: false,
118        overviewMapControl: false,
119        resetBoundsOnResize:true
120      }"
121      style=" height: 100%;">
122      <map-marker
123        :position="m.position"
124        :opacity="m.opacity"
125        :draggable="m.draggable"
126        @position_changed="updMarker(m, $event)"
127        @dragend="updateMarker(m)"
128        v-for="(m, index) in markers"
129        :key="index"
130        >
131      </map-marker>
132    </google-map>
```

Fonte: elaborado pelo autor.

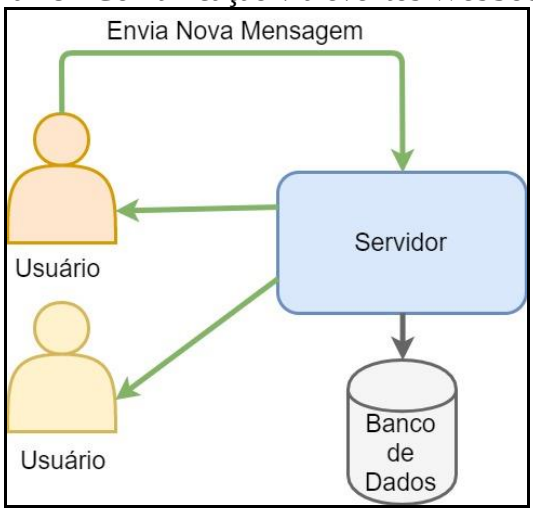
Na linha 110 do Quadro 16 é apresentada a *tag* customizada que representa o componente `googleMap`. Essa *tag* contém alguns atributos para compor o mapa como o atributo `center` que recebe a coordenada onde o mapa deve estar centralizado, o `zoom`, que

determina a proximidade em que o mapa será mostrado e o atributo `options` que é composto por vários outros atributos. Para esta instância do mapa, foram retiradas algumas opções, como por exemplo, alternar entre os tipos de mapa (linha 116) e ativar o *Street View* (linha 117).

Dentro do componente `googleMap` está o componente `mapMarker` (linha 122). Este componente representa o marcador do provedor de serviço no mapa. Esta *tag* também possui atributos para sua composição na tela. Alguns exemplos são: o atributo `position` (linha 123), que recebe a coordenada do provedor de serviço, o atributo `opacity` (linha 124), que representa a opacidade da marcação no mapa e o atributo `draggable` (linha 125) que define se a marcação poderá ser movida no mapa. A ação `dragend` (linha 127) é ativada quando um marcador é movido e executa a função `updateMarker`, que atualiza as coordenadas do provedor de serviço.

O chat do aplicativo utiliza *WebSockets* para fazer a comunicação entre os usuários e o módulo servidor. Para lidar com esses eventos foi utilizado o *framework* SocketIO. Este *framework* possibilita que o chat funcione em tempo real, sem a necessidade do usuário atualizar a tela para receber a nova mensagem. A comunicação entre os usuários e o servidor utilizando *WebSockets* é demonstrada pela Figura 18.

Figura 18 - Comunicação via eventos WebSockets



Fonte: elaborado pelo autor.

A Figura 18 exemplifica a comunicação do chat entre dois usuários. Quando um usuário envia uma nova mensagem, ela é enviada para o módulo servidor que grava no banco de dados a mensagem. Após a inserção no banco, o módulo servidor encontra os usuários que estão *online* nessa conversa e envia um evento para atualizar as mensagens da tela de cada um. O Quadro 17 apresenta a função `sendMessage` que faz esse envio e atualização.

Quadro 17 - função para enviar mensagem para o servidor via WebSocket

```
128       sendMessage (message) {
129         this.message = ''
130         this.$socket.emit('direct-message', {
131           to: this.messageState.other_user.id,
132           content: message
133         })
134       },
```

Fonte: elaborado pelo autor.

A função `sendMessage` apresentada no Quadro 17 recebe por parâmetro a variável `message` (linha 128) e é responsável por emitir o evento *WebSocket* `direct-message` (linha 130) para o módulo servidor. Este evento contém o `id` do usuário de destino (linha 131) e o conteúdo da mensagem (linha 132). O Quadro 18 apresenta a função que lida com o evento `direct-message` no módulo servidor.

Quadro 18 – Trecho do código-fonte que lida com o evento de direct-message

```
16      socket.on("direct-message", app.sockets.authentication(io, socket, (error, user, message) => {
17          if (error) return;
18          app.db.models.Messages.create({
19            from_id: user.id,
20            to_id: message.to,
21            text: message.content
22          })
23          .then(directMessage => {
24              emitToUsers('direct_message', [user.id, message.to], directMessage)
25          })
26      }))
27  })()
28
29  const emitToUsers = function(eventName, userIds, payload) {
30      app.sockets.onlineStore.getSocketsFromUsers(userIds).forEach(socketId => {
31          console.log(socketId + ': ' + payload.text);
32          io.sockets.connected[socketId].emit(eventName, payload)
33      })
34  }
```

Fonte: elaborado pelo autor.

A função apresentada na linha 16 do Quadro 18 é responsável por gravar a mensagem no banco de dados e dispersar a mensagem para os usuários envolvidos que estão *online*. Na linha 18 é utilizada a função `create` do Sequelize para criar um novo registro da tabela `Mensagem` passando por parâmetro as colunas que devem ser gravadas: `from_id` (linha 19), `to_id` (linha 20) e `text` (linha 21). Após a inserção no banco é utilizada a função `emitToUsers` (linha 24) para verificar quem está *online* nessa conversa e então repassar o evento `direct-message` para estes usuários. O Quadro 19 mostra a função que lida com o evento `direct-message` no aplicativo.

Quadro 19 – Função para lidar com o evento direct-message

```
16    socket_directMessage (store, message) {
17      let to = Number(message.to_id),
18        from = Number(message.from_id),
19        contactId = Number(store.rootState.route.params.userId),
20        loggedUserId = Number(store.rootState.setupModule.id)
21      if ((to === contactId && from === loggedUserId) ||
22        (to === loggedUserId && from === contactId)) {
23        store.dispatch(messageActions.PERFORM_LOAD_MESSAGE)
24      }
25    },
```

Fonte: elaborado pelo autor.

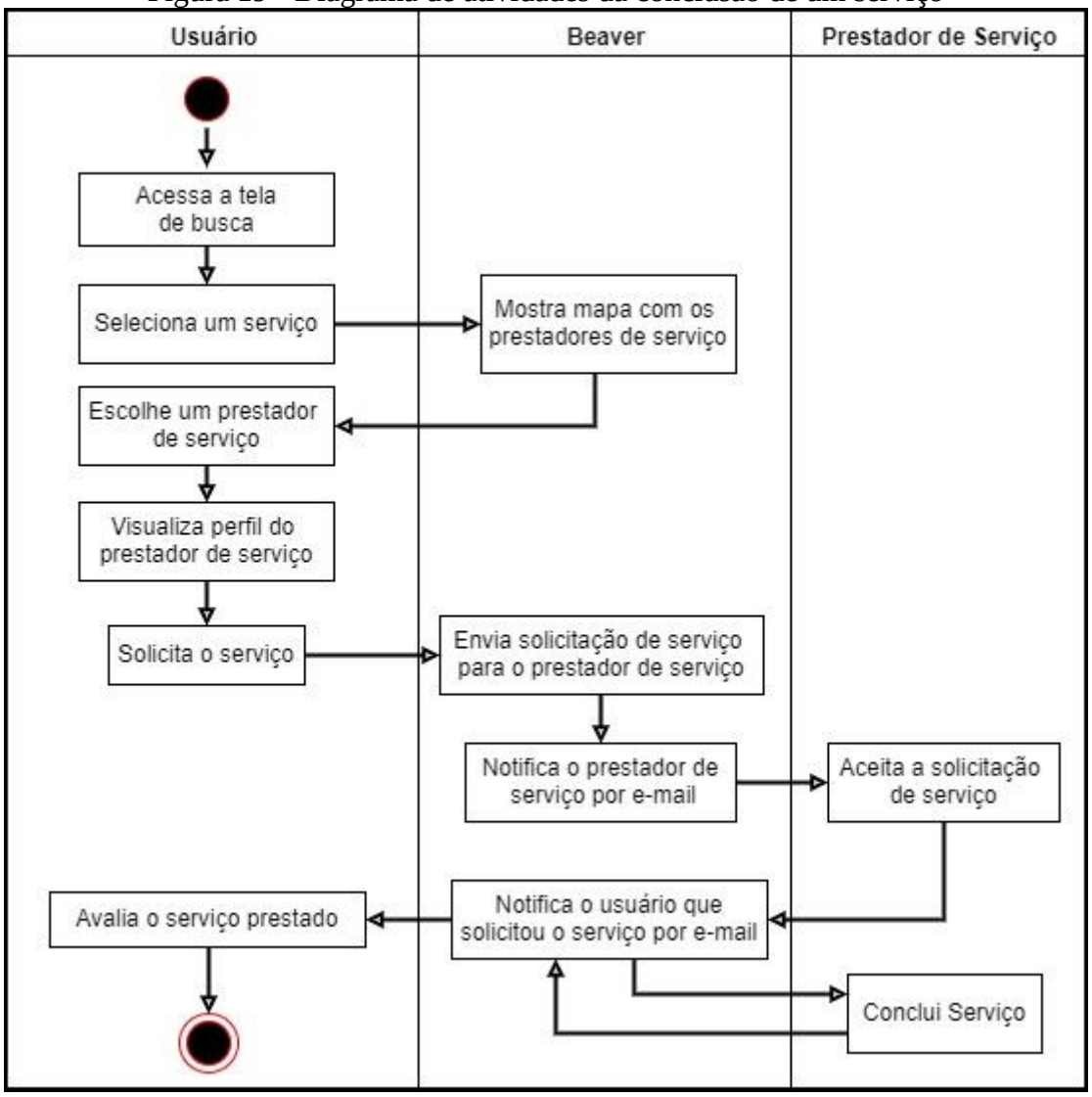
A função `socket_directMessage` (linha 16) é chamada quando o módulo servidor emite o evento `direct-message`. Ela é responsável por atualizar a tela com a nova mensagem, através da `action` chamada `PERFORM_LOAD_MESSAGE` que faz uma nova busca das mensagens do banco e atualiza a lista de mensagens na camada `State` do VueX.

3.3.3 Operacionalidade da implementação

Nesta seção é apresentada a operacionalidade da implementação do aplicativo Beaver, demonstrando o fluxo apresentado no diagrama de atividades. A Figura 19 ilustra o diagrama de atividades referente à solicitação de um serviço, contendo os dois principais atores do sistema: `Usuário` e `Prestador de Serviço`.

No início do diagrama apresentado o `Usuário` acessa a tela de busca no aplicativo. Em seguida ele seleciona um serviço que ele queira contratar. O aplicativo `Beaver` mostra um mapa com os prestadores de serviço que estão na região e oferecem esse serviço. O `Usuário` escolhe um prestador de serviço no mapa e visualiza seu perfil. O `Usuário` então, solicita o serviço. O `Beaver` enviar a solicitação para o prestador de serviço e o notifica por e-mail. O `prestador de serviço` aceita a solicitação e o `Beaver` notifica o usuário que solicitou o serviço por e-mail. Após o `Prestador de serviço` concluir a solicitação de serviço, o `Beaver` notifica o usuário por e-mail. O `Usuário` então avalia o serviço prestado.

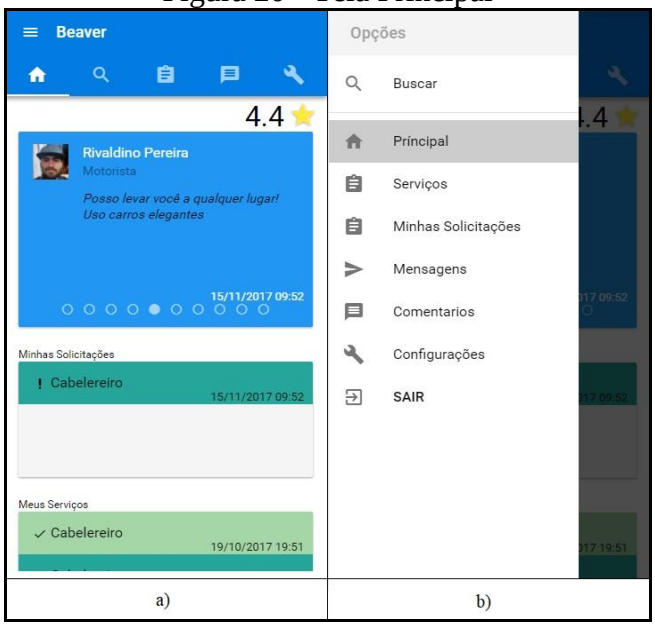
Figura 19 - Diagrama de atividades da conclusão de um serviço



Fonte: elaborado pelo autor

Para melhor compreensão desse fluxo, a tela principal do aplicativo é ilustrada na Figura 20, na qual a maioria das telas são acessadas. Na tela principal (Figura 20 a) são mostrados os últimos serviços cadastrados até o momento na parte superior. Nessa tela também são listados os serviços solicitados pelo usuário, que estão em aberto ou aceitos pelo prestador de serviço, na parte inferior. Caso o usuário também seja um prestador de serviço, são listados nessa tela os serviços que o usuário presta que foram solicitados. Prestadores de serviço tem sua avaliação apresentada no canto superior direito da tela.

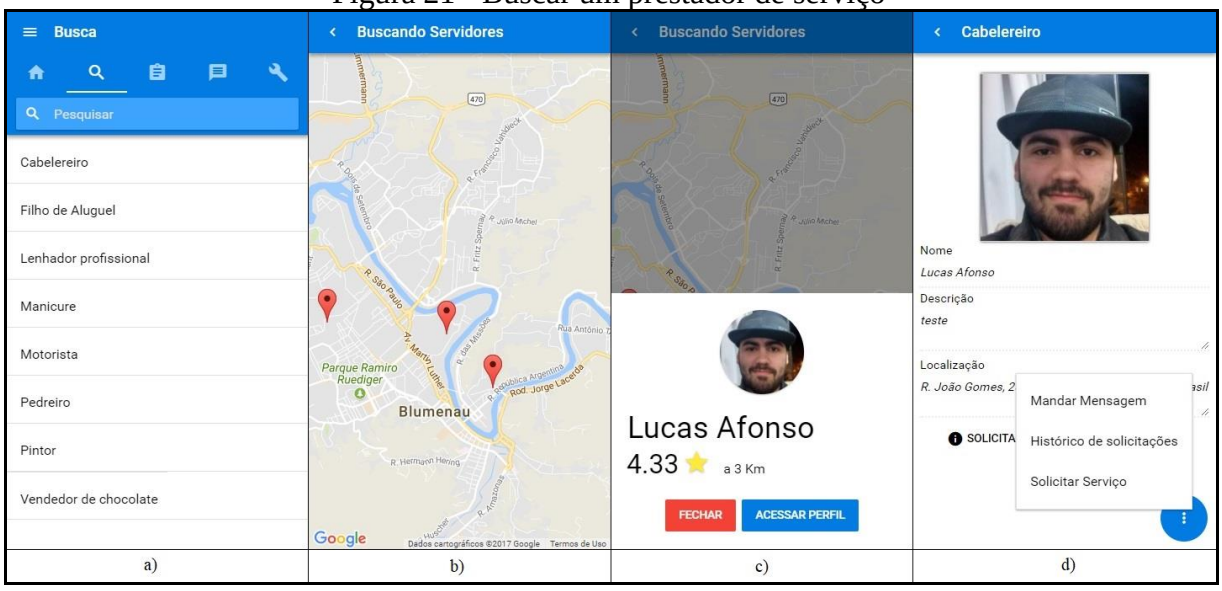
Figura 20 - Tela Principal



Fonte: elaborado pelo autor.

A parte superior da tela possui cinco abas para facilitar o acesso aos recursos do aplicativo. Além das abas, também é possível acessar, de qualquer tela, um menu lateral (Figura 20 b) contendo o acesso para todos os recursos do aplicativo. Para a solicitação de serviço apresentada no diagrama de atividades é preciso primeiro acessar a tela de busca de prestadores de serviço do aplicativo. A Figura 21 demonstra os passos para buscar um prestador de serviço.

Figura 21 - Buscar um prestador de serviço



Fonte: elaborado pelo autor.

Acessando a tela `Busca` (Figura 21 a) é apresentada uma lista de serviços cadastrados no aplicativo. O usuário pode filtrar a lista utilizando o campo de pesquisa na parte superior da tela e selecionar um tipo de serviço da lista. Ao selecionar o tipo de serviço, é apresentado

um mapa (Figura 21 b) com os prestadores deste serviço próximos a sua geolocalização. O mapa é centralizado na localização do usuário. Ao selecionar um prestador de serviço do mapa são apresentadas algumas de suas informações (Figura 21 c), como a foto, nota, distância e nome, além de uma opção para acessar seu perfil.

Na tela de perfil do prestador de serviço (Figura 21 d) são mostrados seus dados e pelo menu, que fica no canto inferior direito da tela, é possível enviar mensagens para o prestador de serviço através de um chat, além de visualizar seu histórico de solicitações para esse serviço e solicitar um serviço. Essas funcionalidades são apresentadas na Figura 24 ao fim dessa seção. Para aceitar um serviço solicitado, o prestador de serviço deve acessar a tela de solicitação de serviço conforme mostra a Figura 22.

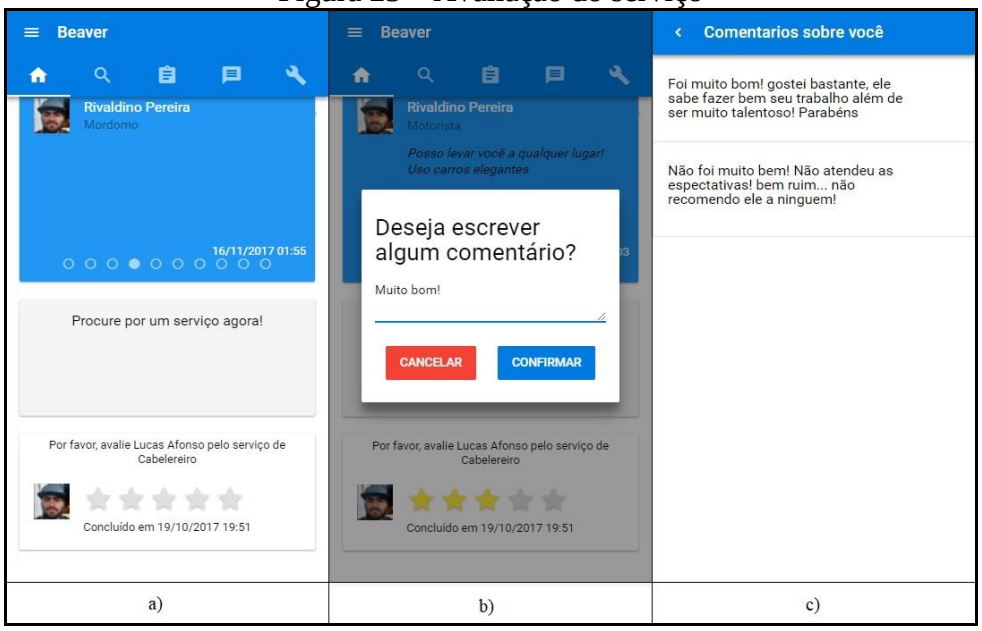
Figura 22 – Situações de uma solicitação de serviço

Fonte: elaborado pelo autor.

Na Figura 22 (a) é apresentada a solicitação de serviço que contém as informações sobre as solicitações realizadas. Essa tela é visualizada somente pelo prestador de serviços. A tela também apresenta os botões para aceitar ou recursar a solicitação, visualizar o perfil do solicitante e o botão `ENVIAR MENSAGEM` na qual direciona o usuário para um chat com o solicitante. Ao aceitar a solicitação, surge um botão para concluir a solicitação (Figura 22 b).

Para o usuário que solicitou o serviço, além das informações da solicitação, são apresentados nessa tela (Figura 22 c) os botões para cancelar a solicitação, visualizar perfil do prestador de serviço e o botão `ENVIAR MENSAGEM` na qual direciona o usuário para um chat com o prestador de serviço. Toda vez que uma solicitação é criada ou sua situação é alterada é enviado um e-mail de aviso para o responsável (Figura 22 d). Quando a solicitação é concluída, o usuário que solicitou o serviço pode avaliá-lo. A Figura 23 apresenta o processo de avaliação do serviço.

Figura 23 – Avaliação do serviço



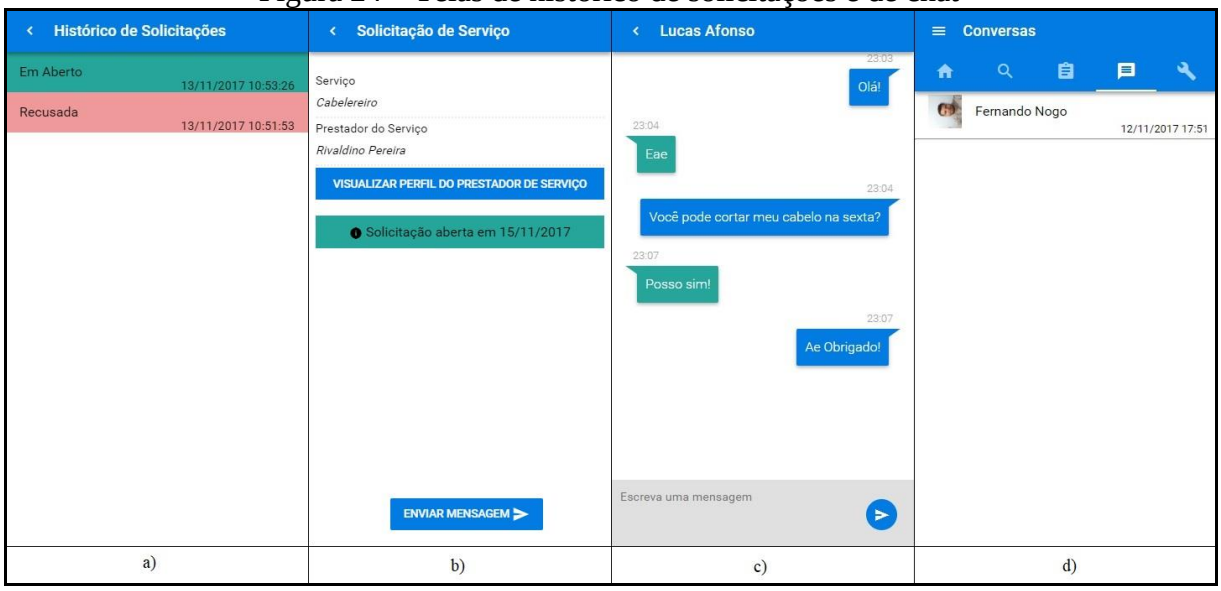
Fonte: elaborado pelo autor.

Quando a solicitação de serviço é concluída, a tela principal do aplicativo (Figura 23 a) apresenta um lembrete ao usuário solicitante para avaliar o serviço prestado, localizado na parte inferior da tela. O lembrete apresenta cinco estrelas que representam a nota da avaliação. Após dar uma nota ao serviço, selecionando uma quantia de estrelas, o aplicativo pede para que o usuário escreva um comentário (Figura 23 b). Este campo não é obrigatório, sendo possível confirmar sem preenche-lo.

A média das notas do prestador de serviço é mostrada na tela principal conforme já mostrado na Figura 20 (a). Para essa versão do aplicativo optou-se para que somente o prestador de serviço possa ver os comentários sobre ele. A tela `Comentários sobre você` é acessada no menu mostrado na Figura 20 (b). Duas funcionalidades que são importantes para os usuários do Beaver são as telas `Histórico de Solicitações` e o `chat` que possibilita conversar com outros usuários. Essas telas são apresentadas na Figura 24.

A Figura 24 (a) apresenta a tela `Histórico de Solicitações`, que lista todas as solicitações de serviços que o usuário já fez ao prestador de serviço. Essa tela pode ser acessada da tela de perfil do prestador de serviço (Figura 21 d). Ao selecionar um item da lista, o usuário acessa a tela `Solicitação de Serviço` (Figura 24 b), que mostra as informações sobre a solicitação e contém um botão para acessar o perfil do prestador de serviço e o botão `ENVIAR MENSAGEM` para abrir o chat com ele.

Figura 24 – Telas de histórico de solicitações e de chat



Fonte: elaborado pelo autor.

A Figura 24 (c) apresenta o `chat` entre os usuários. Tanto o prestador de serviço quanto o usuário que solicitou o serviço tem acesso a tela `Conversas` (Figura 24 d) que possui uma lista das conversas feitas até o momento.

3.4 RESULTADOS E DISCUSSÕES

Os trabalhos correlatos foram fundamentais para o desenvolvimento do trabalho, pois serviram de base para definir as funcionalidades do aplicativo Beaver. O aplicativo Uber (2017) e o trabalho de Silva (2016) serviram de base para construção das regras de negócio, sendo estes mais voltados para o consumo colaborativo, como a solicitação de serviço, chat e o sistema de avaliação por estrelas. O trabalho de Campos (2015) ajudou na percepção da utilização da API do Google Maps. O Quadro 20 apresenta a relação de características presentes nos trabalhos correlatos e no aplicativo desenvolvido.

Quadro 20 - Relação de características presentes nos trabalhos correlatos e no aplicativo desenvolvido

Características	Uber (2017)	Campos (2015)	Silva (2016)	Beaver
Plataforma	Android / IOS	IOS	Web	Híbrido
Modelo de consumo colaborativo	Sim	Não	Sim	Sim
Ferramenta de Geolocalização	Google Maps, Waze e Uber	Google Maps	Não possui	Google Maps
Busca pelo prestador de serviço	Por meio de um mapa	Não se aplica	Lista	Através de um mapa
Chat para os usuários	Sim	Não	Não	Sim
Permite avaliação dos clientes	Sim	Não	Sim	Sim
Avaliações feitas por estrelas	Sim	Não	Sim	Sim

Fonte: elaborado pelo autor.

Percebe-se pelo Quadro 20 que enquanto o Uber (2017) e o trabalho de Campos (2015) são desenvolvidos para plataformas móveis, o trabalho de Silva (2016) possui sua interface apenas para navegadores web. O aplicativo desenvolvido utilizou de algumas ideias retiradas dessas ferramentas para a criação da interface, sendo voltada para aplicações móveis.

Silva (2016) utiliza uma listagem na busca dos prestadores de serviço. O Beaver emprega um mapa para mostrá-los, assim como Uber (2017), utilizando a API Google Maps como Campos (2015). Uber (2017) e Silva (2016) permitem uma avaliação dos clientes sobre o serviço através de uma nota que é representada por um número de estrelas. O aplicativo desenvolvido também adotou este método de avaliação. Além disso, o Beaver utilizou um chat para os usuários se comunicarem, como Uber (2017).

Para testar a usabilidade e operacionalidade do aplicativo foi realizada uma avaliação heurística que é descrita na seção 3.4.1.

3.4.1 Avaliação Heurística

A avaliação heurística é uma técnica de inspeção de usabilidade utilizando um conjunto de heurísticas em que especialistas avaliam se os elementos da interface com o usuário estão de acordo com os princípios especificados. Essa técnica busca por problemas de usabilidade e propõe soluções para elas, com o objetivo de antever potenciais problemas (NIELSEN; MACK, 1994). Para avaliar o aplicativo através deste método, ele foi publicado em uma rede local na casa do autor deste trabalho. Três membros especialistas foram convidados a realizar uma avaliação heurística adaptada do trabalho de Mantau et al. (2013), afim de aferir o grau de usabilidade da aplicação desenvolvida.

Os especialistas selecionados possuem diferentes perfis. Dois deles tem uma segunda renda como autônomo, sendo um como barbeiro e outro como marido de aluguel, sendo estes especialistas em prestação de serviços de forma autônoma. O terceiro avaliador trabalha com desenvolvimentos aplicações móveis, sendo este um especialista voltado a usabilidade para dispositivos móveis, bem como usuário de um serviço de prestação de serviços. Para aplicar o processo de avaliação foi empregado um formulário elaborado pelo autor utilizando a ferramenta de formulários do Google, que pode ser visualizado na íntegra no Apêndice B. A seguir são descritas as heurísticas utilizadas para o processo de avaliação, adaptadas de Nielsen e Mack (1994) por Mantau et al. (2013), que tem por objetivo avaliar sistemas móveis com funcionalidades colaborativas. O sistema em questão é classificado como um sistema colaborativo, pois possui os três eixos da colaboração sendo: coordenação, cooperação e comunicação. A comunicação é dada através do chat, a cooperação através da

disponibilização de serviços para a comunidade, bem como pela contratação destes e a coordenação através do prestador de serviço.

a) H1 - Visibilidade do estado do sistema: o aplicativo deve fornecer feedback adequado aos usuários dentro de um tempo razoável;
b) H2 - Compatibilidade do sistema com o mundo real: o sistema deve usar termos familiares ao usuário ao invés de termos orientados ao software;
c) H3 - Consistência e mapeamento: o modelo conceitual de interação do usuário com o aplicativo deve estar de acordo com o contexto;
d) H4 - Reconhecimento ao invés de memorização: o aplicativo deve conter instruções visíveis ou de fácil recuperação quando necessárias;
e) H5 - Flexibilidade e eficiência de uso: o aplicativo deve prover meios para usuários experientes acelerarem a interação e de apoiar usuários novatos;
f) H6 – Design estético e minimalista: o aplicativo deve exibir apenas as informações que sejam importantes e necessárias;
g) H7 - Gerenciamento de erros: o aplicativo deve emitir mensagens de erro com linguagem clara indicando o problema e construtivamente sugerindo uma solução;
h) H8 - Facilidade de entrada, visualização, e leitura da tela: os dados apresentados na tela devem ser fáceis de serem lidos e navegados;
i) H9 - Convenções estéticas, sociais e privativas: Levar em consideração aspectos estéticos e emocionais a serem apresentados nos dispositivos móveis. Deixar claro que as informações do usuário estão seguras;
j) H10 - Fornecer comunicação de artefatos compartilhados: o aplicativo deve disponibilizar informações sobre as ações dos outros usuários;
k) H11 - Fornecer proteção: o aplicativo deve conter mecanismos para evitar conflitos ao acesso simultâneo a um mesmo conjunto de artefatos;
l) H12 - Gerenciamento de colaboração de baixo e alto acoplamento: o acoplamento é o grau que as pessoas trabalham em conjunto, e representa a quantidade de trabalho que uma pessoa pode realizar antes de precisar discutir, consultar ou pedir uma informação para outra pessoa;
m) H13 - Permitir que as pessoas coordenem suas ações: o aplicativo deve permitir que as tarefas sejam realizadas na ordem certa, no momento certo, sem ignorar as restrições impostas.

O processo de avaliação foi composto por três etapas. Na primeira etapa cada especialista passou por pelo menos mínimo 30 minutos inspecionando o aplicativo. Na segunda etapa os avaliadores responderam as 13 heurísticas, descritas anteriormente, informando se elas se aplicam ao aplicativo Beaver e caso não se apliquem, informaram os problemas encontrados e seu grau de relevância. O grau de relevância do erro varia de zero a cinco, sendo zero nulo, um, menor relevância para ser corrigido e cinco, alto grau de relevância. Durante as respostas, os avaliadores estavam livres para voltar a usar o aplicativo, afim esclarecer alguma dúvida que possa ter ficado. Na última etapa, os avaliadores se reuniram a fim de discutir o que descobriram e priorizar os problemas que encontraram, bem como sugerir soluções.

3.4.1.1 Resultados

A avaliação heurística encontrou seis problemas de usabilidade ao todo, sendo que cada um dos avaliadores encontrou um conjunto diferente de problemas. O Quadro 21 apresenta o resumo das heurísticas, problemas e gravidade encontrados na avaliação.

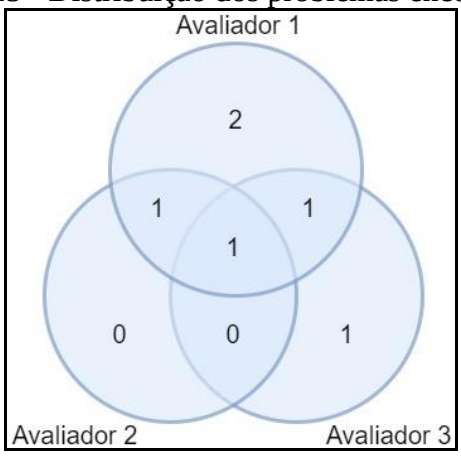
Quadro 21- Resumo das heurísticas, problemas e gravidade

Heurística	Problemas	Gravidade média
H1	1	2
H2	2 e 3	3
H3	--	--
H4	2	2
H5	--	--
H6	--	--
H7	4	2
H8	2	2
H9	5 e 6	1
H10	--	--
H11	--	--
H12	--	--
H13	--	--

Fonte: elaborado pelo autor.

Conforme indica o Quadro 21, não foi encontrado nenhum problema com gravidade acima de 3, no qual é considerado uma catástrofe de usabilidade (NIELSEN; MACK, 1994). Outro aspecto que pode ser observado, é que para as heurísticas H3, H5, H6, H10, H11, H12 e H13, não foi encontrado nenhum problema de usabilidade, o que demonstra que o aplicativo Beaver está de acordo com o defendido por estas heurísticas. O número dos problemas corresponde a descrição dos problemas apresentados no decorrer desses resultados. A Figura 25 apresenta a distribuição dos problemas encontrados para cada um dos avaliadores.

Figura 25 - Distribuição dos problemas encontrados



Fonte: elaborado pelo autor.

Embora o Quadro 21 indique que os principais problemas do Beaver estão relacionados aos aspectos da compatibilidade do sistema com o mundo real (H2) e convenções estéticas, sociais e privativas (H9), também deve-se levar em consideração o Problema 1 (Quadro 22) que, conforme mostra a Figura 25, foi um problema levantado pelos três avaliadores. Os principais problemas encontrados estão listados no Quadro 22, bem como uma possível solução para o mesmo, sendo que a solução também foi indicada pelos especialistas.

Quadro 22 – Principais problemas encontrados

Problema 1	Falta de notificações. O envio de e-mail não é suficiente. Este problema viola a H1 e é classificado como uma gravidade de grau 2. Solução: Incluir notificações sobre as ações de outros usuários e símbolos informando quando houver novas mensagens.
Problema 2	Nome de algumas seções são pouco explicativos. A tela "Serviços", gerou dificuldades de distinguir se os serviços eram do usuário ou se era uma busca de serviço. Este problema viola a H2 e H4 e H8 e é classificado como uma gravidade de grau 2. Solução: alterar o nome de algumas telas.
Problema 3	Falta de uma lista de nomes de prestador de serviços como alternativa além do mapa para a busca. Este problema viola a H2 e é classificado como uma gravidade de grau 3. Solução: Adicionar a alternativa de buscar os prestadores de serviços através de uma lista de nomes.
Problema 4	Mensagens com erros não relacionadas com a usabilidade do sistema foram identificadas, pois ocasionalmente ocorriam erros no banco de dados. Este problema viola a H7 e é classificado como uma gravidade de grau 2. Solução: Tratar os erros do sistema para que sejam mostrados erros que o usuário possa entender.
Problema 5	Uma foto tirada do aparelho aparece normal para o usuário, porém aparece virada quando vista por os outros usuários (A foto foi tirada de um IPad Mini). Este problema viola a H9 e é classificado como uma gravidade de grau 1. Solução: Investigar o problema que está virando a foto.
Problema 6	Falta de uma confirmação de senha no momento do cadastro. Este problema viola a H9 e é classificado como uma gravidade de grau 1. Solução: incluir confirmação de senha.

Fonte: elaborado pelo autor.

4 CONCLUSÕES

Neste trabalho é apresentado um aplicativo web responsivo para encontrar prestadores de serviços locais autônomos por geolocalização. Conforme os objetivos do trabalho, pode-se afirmar que os objetivos específicos foram alcançados, pois o aplicativo oferece uma alternativa para prestadores de serviços de anunciar seu trabalho e possibilita que indivíduos encontrem prestadores de serviços autônomos através de geolocalização. Sua usabilidade pode ser comprovada com a avaliação heurística por meio de três especialistas, que apontaram problemas que afetam apenas seis heurísticas, sendo que nenhum problema atingiu gravidade maior que três.

Pela avaliação heurística, percebe-se que o aplicativo tem alguns pontos a serem melhorados em sua usabilidade, além da falta de algumas funcionalidades. A falta de notificações em segundo plano foi um problema encontrado pelos três avaliadores. Além disso, foi citado pelos especialistas a falta de uma opção para encontrar prestadores de serviço pelo nome além do mapa. Algumas telas tem o nome pouco explicativo, dificultando o entendimento dos usuários, como por exemplo a tela `Serviços`, que deixou ambíguo se apresentava os serviços contratados ou os serviços nos quais o usuário era o prestador.

Uma grande dificuldade para o desenvolvimento deste trabalho foi a falta de conhecimento do autor sobre as tecnologias empregadas, gerando uma grande curva de aprendizado e tomando um tempo maior no desenvolvimento do aplicativo. No entanto, essas tecnologias se provaram ser úteis e ágeis para criação do aplicativo. Os *frameworks* VueJS e Quasar possuem uma comunidade bastante ativa, o que auxiliou no desenvolvimento do Beaver.

É possível concluir que este trabalho possibilitou uma alternativa para os prestadores de serviço de mostrarem seu trabalho no mercado, estimulando o consumo colaborativo. Além disso, este trabalho possui contribuição tecnológica para a automatização do consumo colaborativo, motivando a economia social. A possibilidade de indivíduos encontrarem prestadores de serviços autônomos através de geolocalização, acarreta um foco maior para os prestadores da região, fortalecendo a comunidade local dos trabalhadores autônomos. A avaliação do serviço, gera um aumento progressivo na qualidade dos serviços, uma vez que a busca pelas avaliações mais positivas, ocasionem um aumento da credibilidade e confiabilidade da comunidade local.

4.1 EXTENSÕES

Como sugestões para possíveis extensões ao trabalho desenvolvido citam-se:

a) permitir que o conteúdo do aplicativo seja visualizado de forma pública, sem a necessidade de autenticação do usuário;
b) permitir que os usuários reportem outros usuários por má conduta;
c) refinar padrão de design das telas;
d) suporte para pagamento dos serviços;
e) solicitar grupo de serviços de um mesmo prestador de serviço;
f) permitir que o usuário veja a localização do prestador de serviço em tempo real;
g) enviar notificação para os usuários quando recebem mensagens;
h) enviar notificação para os usuários quando um serviço é solicitado ou aceito;
i) apresentar fotos do serviço feito
j) apresentar cartão de visita do prestador de serviço
k) melhorar as regras de usabilidade identificadas na avaliação heurística.

REFERÊNCIAS

ASAAS (Ed.). **Vida de profissional autônomo**: dicas para captação dos primeiros clientes. 2013. Disponível em: <https://www.asaas.com/blog/vida-de-profissional-autonomo-dicas-para-captacao-dos-primeiros-clientes/>. Acesso em: 20 mar. 2017.

AVILA, Mário L.; AVILA, Silvia R. S. A. **Satisfação de usuários**: Uma análise dos serviços prestados por uma cooperativa médica. Rubiataba, 2001. Disponível em: <http://www.fecap.br/adm_online/art24/silvia.htm>. Acesso em: 19 mar. 2017.

BARRANGER, Daniel. **Entenda a economia colaborativa de serviços como Uber, Airbnb e outros**. 2014. Disponível em: <https://artigos.softonic.com.br/uber-airbnb-economia-compartilhar>. Acesso em: 25 mar. 2017.

BASILIO, Andressa. Empreendedores descobrem o consumo colaborativo. **Época Negócios**, [s. L.], 28 jan. 2013. Disponível em: <http://epocanegocios.globo.com/Inspiracao/noticia/2012/10/empreendedores-descobrem-o-consumo-colaborativo.html>. Acesso em: 21 maio 2017.

CAMPOS, Gabriel Felipe Borges de. **Sistema móvel na plataforma phonegap para compartilhamento de geolocalização integrado a rede social**. 2015. 58 f. Trabalho de Conclusão de Curso (Bacharelado em Ciências da Computação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

FILANTROPIA, Instituto. **Consumo Colaborativo**: A Tendência Compartilhada. 2015. Disponível em: <http://www.filantropia.ong/informacao/8271-consumo-colaborativo-a-tendência-compartilhada>. Acesso em: 19 mar. 2017.

FRANCINI, Caroline. **Trabalhando com profissionais autônomos**: tudo o que você precisa saber. 2016. Disponível em: <https://blog.contaazul.com/como-trabalhar-com-profissionais-autonomos/>. Acesso em: 08 jul. 2016.

G1. **Waze, Google Maps ou mapa da Uber**: o GPS que seu motorista escolhe faz diferença na corrida? 2017. Disponível em: <https://g1.globo.com/tecnologia/noticia/waze-google-maps-ou-mapa-da-uber-o-gps-que-seu-motorista-escolhe-faz-diferenca-na-corrida.ghtml>. Acesso em: 19 nov. 2017.

GASPARETTO, Luiz Eduardo. **A Imagem da Prestação de Serviços**. 2012. Disponível em: <http://www.blogdogasparetto.com.br/a-imagem-da-prestacao-de-servicos/>. Acesso em: 18 maio 2017.

GRAGNANI, Juliana. Passageiros do Uber 'se comportam' para receber boas notas. **Folha de São Paulo**, São Paulo, 13 nov. 2016. Disponível em: <http://www1.folha.uol.com.br/cotidiano/2016/11/1831943-passageiros-do-uber-se-comportam-para-receber-boas-notas.shtml>. Acesso em: 29 maio 2017.

ICLARIFIED. **Uber adds in-app chat between riders and drivers**. 2017. Disponível em: <http://www.iclarified.com:8081/61924/uber-adds-inapp-chat-between-riders-and-drivers>. Acesso em: 19 nov. 2017.

LIMA, Dan; GUILEN, Carol. **Você já conhece o Consumo Colaborativo?** 2012. Disponível em: <http://www.coletivoverde.com.br/consumo-colaborativo/>. Acesso em: 23 mar. 2017.

MALEK, Karen. **Como pagar em dinheiro uma corrida no Uber**. 2017. Disponível em: <http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2016/09/como-pagar-em-dinheiro-uma-corrida-no-uber.html>. Acesso em: 22 maio 2017.

MANTAU, Márcio J. et al. **Avaliação heurística para groupwares móveis**: um estudo de caso utilizando um audience response system. Joinville, 2013.

MARTINS, Filipe Bruno de Araújo Ramos; RAMOS, Anatália Saraiva Martins; RAMOS, Rubens Eugênio Barreto. Uso da tecnologia da informação na prestação de serviços: possibilidades e perspectivas. In: ENCONTRO NACIONAL DE ENGENHARIA DE PRODUÇÃO, 20, 2000, São Paulo. **Anais**... São Paulo: Usp, 2000. p. 1 - 8.

MUNDO CARREIRA. **Saiba quais são as profissões regulamentadas e não regulamentadas no Brasil**. 2014. Disponível em: <http://www.mundocarreira.com.br/guia-de-carreiras/saiba-quais-sao-profissoes-regulamentadas-e-nao-regulamentadas-brasil/>. Acesso em: 29 maio 2017.

NEGOCIO, Destino (Ed.). **Confira quando vale a pena abrir uma empresa como autônomo**. 2015. Disponível em: <http://destinonegocio.com/br/empreendedorismo/confira-quando-vale-a-pena-abrir-uma-empresa-como-autonomo/>. Acesso em: 24 mar. 2017.

NERI, Marcelo et al. Em busca de incentivos para atrair o trabalhador autônomo para a Previdência Social. **Nova Economia**, Belo Horizonte, v. 17, p.363-394, dez. 2007. Disponível em: <http://www.scielo.br/scielo.php?script=sci_arttext&pid=S0103-63512007000300001&lng=en&nrm=iso>. Acesso em: 13 mar. 2017.

NIELSEN, Jakob; MACK, Robert L. **Usability Inspection Methods**. New York: John Wiley & Sons, 1994. 188 p.

OLIVEIRA, Rodrigo. **Marketing de Serviços**: As 4 principais características dos serviços e como gerenciá-las. 2012. Disponível em: <https://www.portalgsti.com.br/2012/10/marketing-de-servicos-as-4-principais.html>. Acesso em: 22 out. 2017.

OST, Stelamaris. Trabalho autônomo. **Âmbito Jurídico**, Rio Grande, v. 51, mar 2008. Disponível em: <http://www.ambito-juridico.com.br/site/index.php?n_link=revista_artigos_leitura&artigo_id=4755>. Acesso em mar 2017.

POLONI, Antônio Sebastião. **Trabalhador Autônomo**. 2003. Disponivel em: < http://uj.novaprolink.com.br/doutrina/1368/trabalhador_autonomo >. Acesso em: 17 de maio de 2017.

R. JUNIOR, José Carlos. **Trabalhar como freelancer ou PJ? Veja vantagens e desvantagens!** 2017. Disponível em: <https://conube.com.br/blog/freelancer-ou-pj/>. Acesso em: 23 out. 2017.

SANTOS, Lázaro Matter dos. **A caracterização do trabalhador autônomo no ordenamento jurídico brasileiro**. 2012. 44 f. Trabalho de Conclusão de Curso (Graduação em Direito) - Departamento de Ciências Jurídicas e Sociais, Universidade Regional do Noroeste do Estado do Rio Grande do Sul, Ijui.

SARDAGNA, José Carlos. **Como fazer marketing sendo profissional autônomo?** 2016. Disponível em: <https://blog.contaazul.com/como-fazer-o-marketing-profissional-autonomo/>. Acesso em: 22 mar. 2017.

SILVA, Marcos Willian da. **Sistema de consumo colaborativo com foco em oferta de serviços locais**. 2016. 56 f. Trabalho de Conclusão de Curso (Bacharelado em Sistemas de Informação) - Centro de Ciências Exatas e Naturais, Universidade Regional de Blumenau, Blumenau.

UBER. **Homepage**. [S.l.], 2017. Disponível em: < https://www.uber.com/pt-BR/our-story/>. Acesso em: 19 mar. 2017.

UBER. **Como as avaliações funcionam?** 2016. Disponível em: <https://help.uber.com/pt_BR/h/e9302f73-8625-427f-abf7-dbe7ab25af7d>. Acesso em: 23 maio 2017.

VAZ, Marcela. **Uber**: como funciona o app de motorista particular. 2015. Disponível em: <http://www.techtudo.com.br/dicas-e-tutoriais/noticia/2014/07/como-usar-o-aplicativo-uber-para-pedir-taxi-e-carona.html>. Acesso em: 22 maio 2017.

VIANA, Márcio Túlio. As relações de trabalho sem vínculo de emprego e as novas regras de competência. **Revista da Faculdade de Direito da Ufmg**, Minas Gerais, v. 46, p.217-241, 2005. Semestral.

VUE. **The Progressive Javascript Framework**. 2017. Disponível em: < https://vuejs.org/ >. Acesso em: 18 dez. 2017.

W3SCHOOLS. **HTML5 Geolocation.** 2017. Disponível em: <https://www.w3schools.com/html/html5_geolocation.asp>. Acesso em: 31 out. 2017.

ZEITHAML, Valarie A.; BITNER, Mary Jo; GREMLER, Dwayne D. **Marketing de Serviços**: A Empresa com Foco no Cliente. 6. ed. Porto Alegre: Amgh Editor Ltda, 2014. Tradução de: Felix Nonnenmacher.

APÊNDICE A – Descrição dos Casos de Uso

Este Apêndice apresenta a descrição dos casos de uso UC02 e UC06. O primeiro caso de uso a ser descrito é o UC02 `Procurar por prestadores de serviço` que se refere a busca por um prestador de serviço. O usuário pode realizar essa ação acessando a tela de busca, através da tela Principal ou do menu lateral. O Quadro 23 explica mais detalhadamente o caso de uso.

Quadro 23 - Descrição do caso de uso Procurar por prestadores de serviço (UC02)

Caso de Uso	**Procurar por prestadores de serviço.**
Descrição	O aplicativo deverá permitir o usuário buscar prestadores de serviço
Ator	Usuário
Pré-condição	O usuário deverá estar autenticado
Fluxo principal	1. O usuário seleciona um serviço; 2. O aplicativo envia a informação ao módulo servidor; 3. O módulo servidor consulta no banco de dados os profissionais que prestam o serviço selecionado e retorna-os em uma lista; 4. O aplicativo mostra um mapa com os prestadores de serviço em suas respectivas localizações.
Fluxo alternativo	3. O módulo servidor consulta no banco de dados os profissionais que prestam o serviço selecionado e retorna uma lista vazia; 4. O aplicativo mostra o mapa sem prestadores de serviço.
Pós-condição	O usuário realiza a busca por um prestador de serviço.

Fonte: elaborado pelo autor.

Como descrito no Quadro 23, o usuário deve estar autenticado para realizar o fluxo principal deste cenário. Estando autenticado, o usuário acessa a tela de busca e seleciona um serviço que esteja precisando. O aplicativo então envia a informação ao módulo servidor. O módulo servidor por sua vez realiza uma consulta no banco de dados pelos profissionais que prestam o serviço selecionado e os retorna em uma lista para o aplicativo. O aplicativo então mostra um mapa com os prestadores de serviço em suas respectivas localizações. O usuário pode então acessar as informações dos prestadores de serviço.

O segundo caso de uso a ser descrito é o UC06 `Manter solicitação de serviço` que se refere a contratação de um serviço e alteração de sua situação (aprovado, reprovado, concluído e cancelado). O usuário pode criar uma solicitação de serviço através da tela de perfil do prestador de serviço. As demais opções são acessadas na tela de solicitação de serviço. O Quadro 24 explica mais detalhadamente o caso de uso.

Quadro 24 - Descrição do caso de uso Manter solicitação de serviço (UC06)

Caso de Uso	**Manter solicitação de serviço.**
Descrição	O aplicativo deverá permitir que o usuário crie, aceite, recuse, conclua e cancele solicitações de serviço.
Ator	Usuário
Pré-condição	O usuário deverá estar autenticado.
Fluxo principal	1. O usuário acessa perfil do prestador de serviço; 2. O usuário solicita serviço do prestador de serviço; 3. O aplicativo envia a informação ao módulo servidor; 4. O módulo servidor cria uma nova solicitação de serviço; 5. O módulo servidor envia e-mail para o prestador de serviço notificando que uma nova solicitação foi feita; 6. O aplicativo informa o usuário sobre a criação.
Pós-condição	Uma nova solicitação de serviço é criada.
Pré-condição para o fluxo alternativo 1	O usuário deverá estar autenticado; A situação da solicitação de serviço deverá estar como “aberta”; O usuário deverá ser o prestador de serviço da solicitação.
Fluxo alternativo 1	1. O usuário acessa a solicitação de serviço; 2. O usuário escolhe a opção: 2.1 Aceitar solicitação; 2.2 Recusar solicitação. Passos para 2.1 Aceitar solicitação 1. O usuário aceita a solicitação de serviço; 2. O aplicativo envia a informação ao módulo servidor; 3. O módulo servidor atualiza a situação da solicitação de serviço para “aceita”; 4. O módulo servidor envia e-mail para o solicitante informando que sua solicitação foi aceita; 5. O aplicativo informa o usuário sobre a alteração. Passos para 2.2 Recusar solicitação 1. O usuário aceita a solicitação de serviço; 2. O aplicativo envia a informação ao módulo servidor; 3. O módulo servidor atualiza a situação da solicitação de serviço para “recusada”; 4. O módulo servidor envia e-mail para o solicitante informando que sua solicitação foi recusara; 5. O aplicativo informa o usuário sobre a alteração.
Pré-condição para o fluxo alternativo 2	O usuário deverá estar autenticado; A situação da solicitação de serviço deverá estar como “aceita”; O usuário deverá ser o solicitante da solicitação de serviço.
Fluxo alternativo 2	1. O usuário acessa a solicitação de serviço; 2. O usuário cancela a solicitação de serviço; 3. O aplicativo envia a informação ao módulo servidor; 4. O módulo servidor atualiza a situação da solicitação de serviço para “cancelada”; 5. O módulo servidor envia e-mail para o prestador de serviço informando que a solicitação foi cancelada; 6. O aplicativo informa o usuário sobre a alteração.
Pré-condição para o fluxo alternativo 3	O usuário deverá estar autenticado; A situação da solicitação de serviço deverá estar como “aceita”; O usuário deverá ser o prestador de serviço da solicitação.
Fluxo alternativo 3	1. O usuário acessa a solicitação de serviço;

	2. O usuário conclui a solicitação de serviço; 3. O aplicativo envia a informação ao módulo servidor; 4. O módulo servidor atualiza a situação da solicitação de serviço para “concluida”; 5. O módulo servidor envia e-mail para o solicitante informando que a solicitação foi concluída; 6. O aplicativo informa o usuário sobre a alteração.
Pós-condição	A situação da solicitação de serviço é alterada.

Fonte: elaborado pelo autor.

O Quadro 24 descreve o fluxo principal e três fluxos alternativos. Para todos os fluxos, o usuário deve estar autenticado. O fluxo principal representa a criação da solicitação de serviço. Nesse fluxo o usuário acessa o perfil do prestador de serviço e solicita o serviço. O aplicativo envia a informação ao módulo servidor, que cria uma nova solicitação de serviço e envia um e-mail para o prestador de serviço notificando sobre a nova solicitação. O aplicativo então informa o usuário sobre a criação.

No fluxo alternativo 1, ao acessar a tela de solicitação de serviço, são apresenta duas opções. Na primeira opção, aceitar solicitação, o usuário acessa o perfil do prestador de serviço e aceita a solicitação. O aplicativo envia a informação ao módulo servidor, que atualiza a situação da solicitação de serviço para `aceita`. Após a atualização, o módulo servidor envia um e-mail para o solicitante informando que a solicitação foi aceita. O aplicativo então informa o usuário sobre a atualização.

Na segunda opção, recusar solicitação, o usuário acessa o perfil do prestador de serviço e recusa a solicitação. O aplicativo envia a informação ao módulo servidor, que atualiza a situação da solicitação de serviço para `recusada`. Após a atualização, o módulo servidor envia um e-mail para o solicitante informando que a solicitação foi recusada. O aplicativo então informa o usuário sobre a atualização.

O fluxo alternativo 2 representa o cancelamento da solicitação de serviço. Nesse fluxo o usuário acessa a tela de solicitação de serviço e a cancela. O aplicativo envia a informação ao módulo servidor, que atualiza a situação da solicitação para `cancelada`. Após a atualização, o módulo servidor envia um e-mail para o prestador de serviço notificando que a solicitação foi cancelada. O aplicativo então informa o usuário sobre o cancelamento.

O fluxo alternativo 3 representa a conclusão da solicitação de serviço. Nesse fluxo o usuário acessa a tela de solicitação de serviço e a conclui. O aplicativo envia a informação ao módulo servidor, que atualiza a situação da solicitação para `concluída`. Após a atualização, o módulo servidor envia um e-mail para o solicitante notificando que a solicitação foi concluída. O aplicativo então informa o usuário sobre a conclusão.

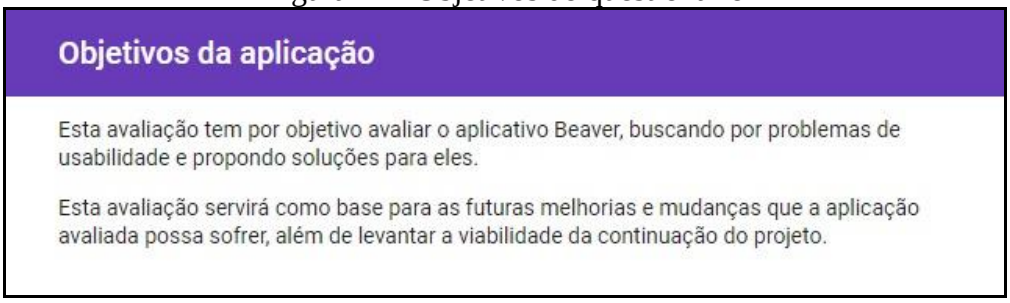
APÊNDICE B – Formulário de avaliação

Este apêndice contém o formulário de avaliação heurística com os especialistas. Este conteúdo foi gerado com o auxílio da ferramenta Google Forms.

Figura 26 - Introdução ao questionário

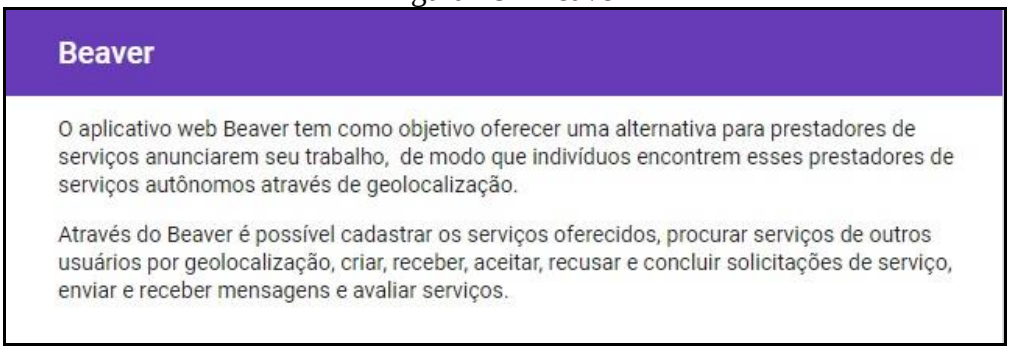
Fonte: elaborado pelo autor.

Figura 27 - Objetivos do questionário



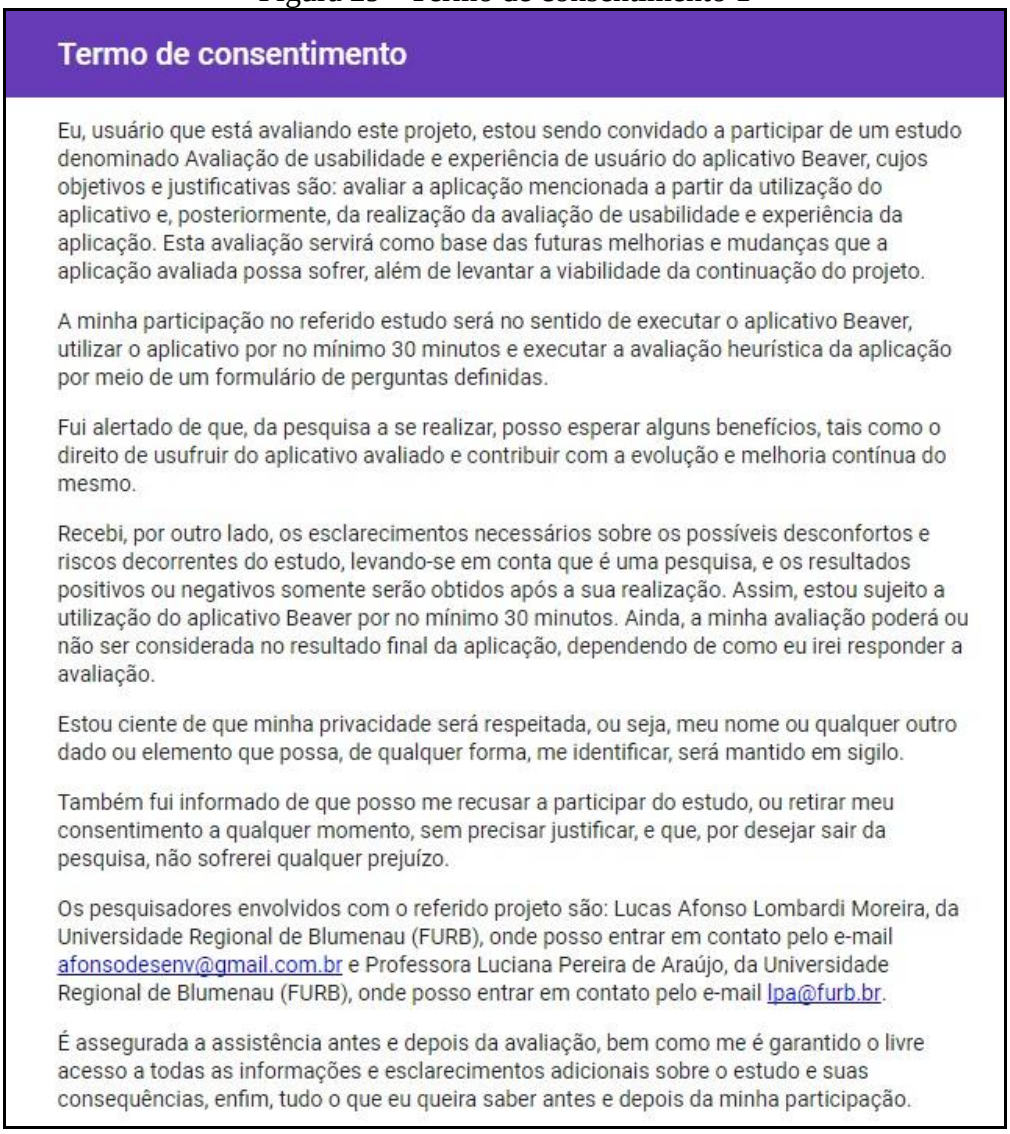
Fonte: elaborado pelo autor.

Figura 28 - Beaver



Fonte: elaborado pelo autor.

Figura 29 - Termo de consentimento 1



Termo de consentimento

Eu, usuário que está avaliando este projeto, estou sendo convidado a participar de um estudo denominado Avaliação de usabilidade e experiência de usuário do aplicativo Beaver, cujos objetivos e justificativas são: avaliar a aplicação mencionada a partir da utilização do aplicativo e, posteriormente, da realização da avaliação de usabilidade e experiência da aplicação. Esta avaliação servirá como base das futuras melhorias e mudanças que a aplicação avaliada possa sofrer, além de levantar a viabilidade da continuação do projeto.

A minha participação no referido estudo será no sentido de executar o aplicativo Beaver, utilizar o aplicativo por no mínimo 30 minutos e executar a avaliação heurística da aplicação por meio de um formulário de perguntas definidas.

Fui alertado de que, da pesquisa a se realizar, posso esperar alguns benefícios, tais como o direito de usufruir do aplicativo avaliado e contribuir com a evolução e melhoria contínua do mesmo.

Recebi, por outro lado, os esclarecimentos necessários sobre os possíveis desconfortos e riscos decorrentes do estudo, levando-se em conta que é uma pesquisa, e os resultados positivos ou negativos somente serão obtidos após a sua realização. Assim, estou sujeito a utilização do aplicativo Beaver por no mínimo 30 minutos. Ainda, a minha avaliação poderá ou não ser considerada no resultado final da aplicação, dependendo de como eu irei responder a avaliação.

Estou ciente de que minha privacidade será respeitada, ou seja, meu nome ou qualquer outro dado ou elemento que possa, de qualquer forma, me identificar, será mantido em sigilo.

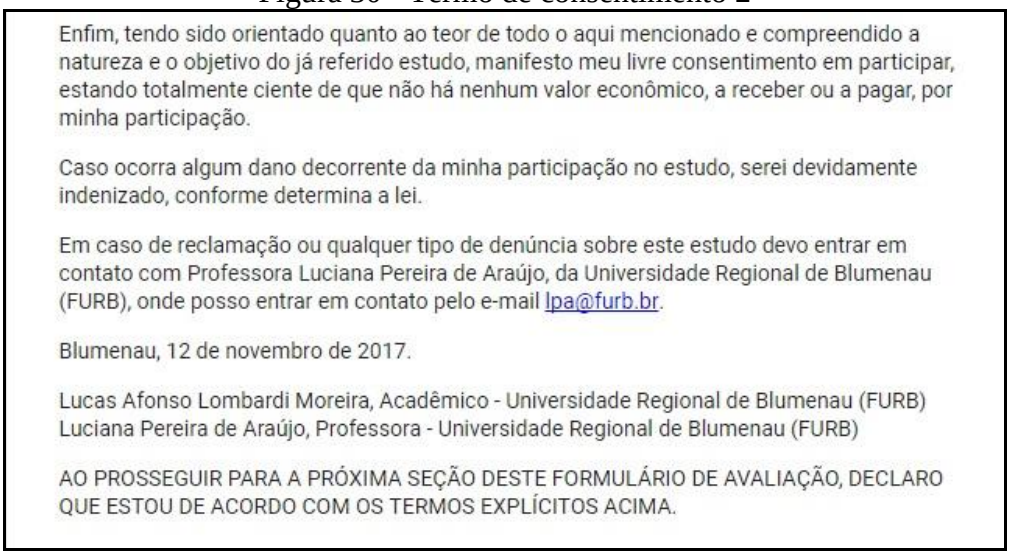
Também fui informado de que posso me recusar a participar do estudo, ou retirar meu consentimento a qualquer momento, sem precisar justificar, e que, por desejar sair da pesquisa, não sofrerei qualquer prejuízo.

Os pesquisadores envolvidos com o referido projeto são: Lucas Afonso Lombardi Moreira, da Universidade Regional de Blumenau (FURB), onde posso entrar em contato pelo e-mail afonsodesenv@gmail.com.br e Professora Luciana Pereira de Araújo, da Universidade Regional de Blumenau (FURB), onde posso entrar em contato pelo e-mail lpa@furb.br.

É assegurada a assistência antes e depois da avaliação, bem como me é garantido o livre acesso a todas as informações e esclarecimentos adicionais sobre o estudo e suas consequências, enfim, tudo o que eu queira saber antes e depois da minha participação.

Fonte: elaborado pelo autor.

Figura 30 - Termo de consentimento 2



Enfim, tendo sido orientado quanto ao teor de todo o aqui mencionado e compreendido a natureza e o objetivo do já referido estudo, manifesto meu livre consentimento em participar, estando totalmente ciente de que não há nenhum valor econômico, a receber ou a pagar, por minha participação.

Caso ocorra algum dano decorrente da minha participação no estudo, serei devidamente indenizado, conforme determina a lei.

Em caso de reclamação ou qualquer tipo de denúncia sobre este estudo devo entrar em contato com Professora Luciana Pereira de Araújo, da Universidade Regional de Blumenau (FURB), onde posso entrar em contato pelo e-mail lpa@furb.br.

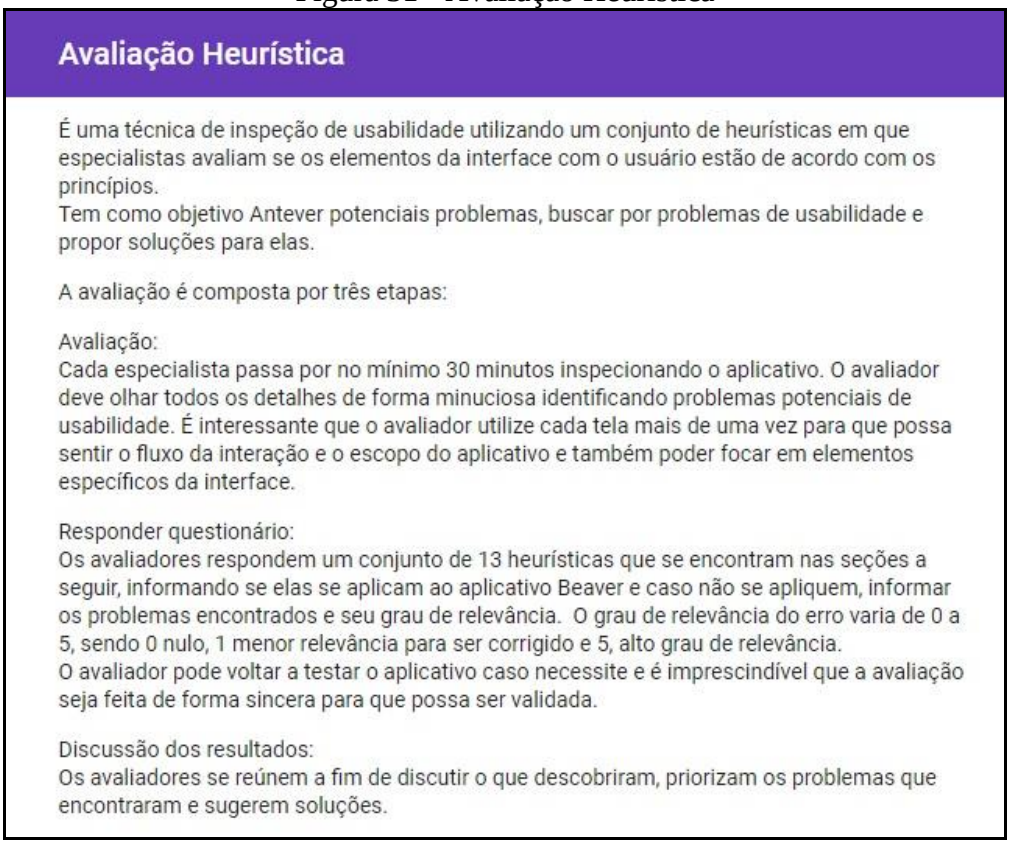
Blumenau, 12 de novembro de 2017.

Lucas Afonso Lombardi Moreira, Acadêmico - Universidade Regional de Blumenau (FURB)
Luciana Pereira de Araújo, Professora - Universidade Regional de Blumenau (FURB)

AO PROSSEGUIR PARA A PRÓXIMA SEÇÃO DESTE FORMULÁRIO DE AVALIAÇÃO, DECLARO QUE ESTOU DE ACORDO COM OS TERMOS EXPLÍCITOS ACIMA.

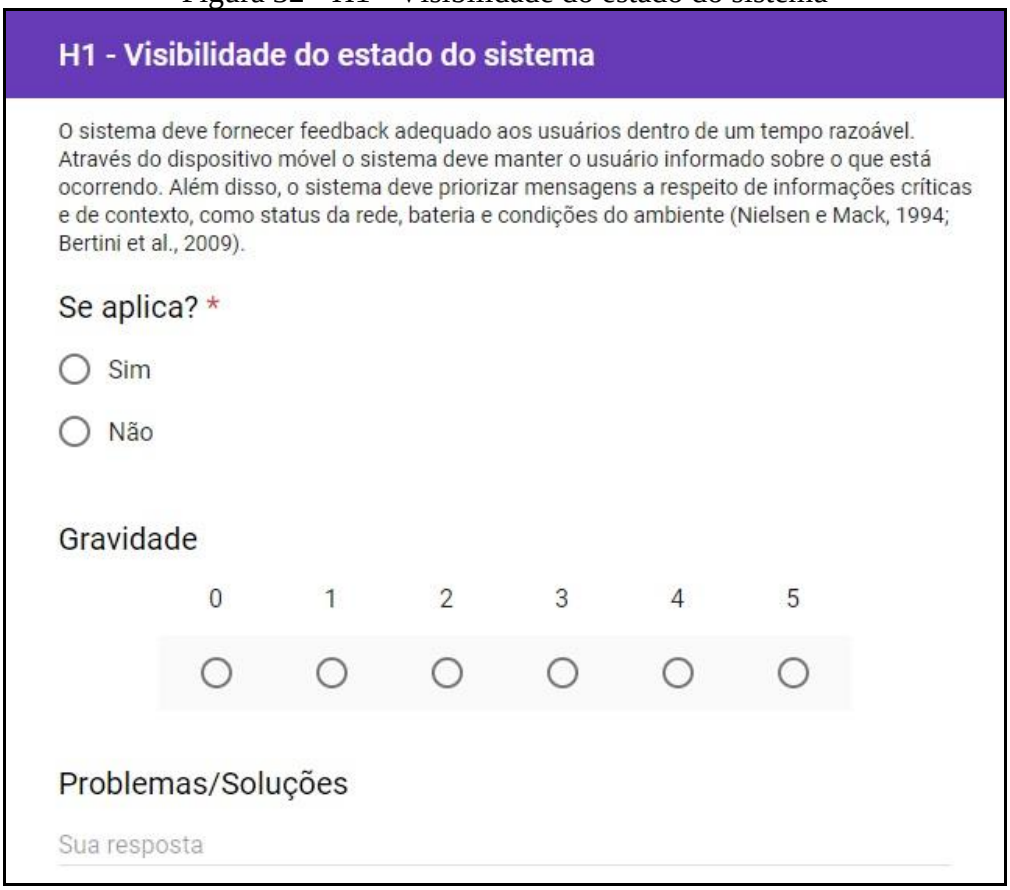
Fonte: elaborado pelo autor.

Figura 31 - Avaliação Heurística



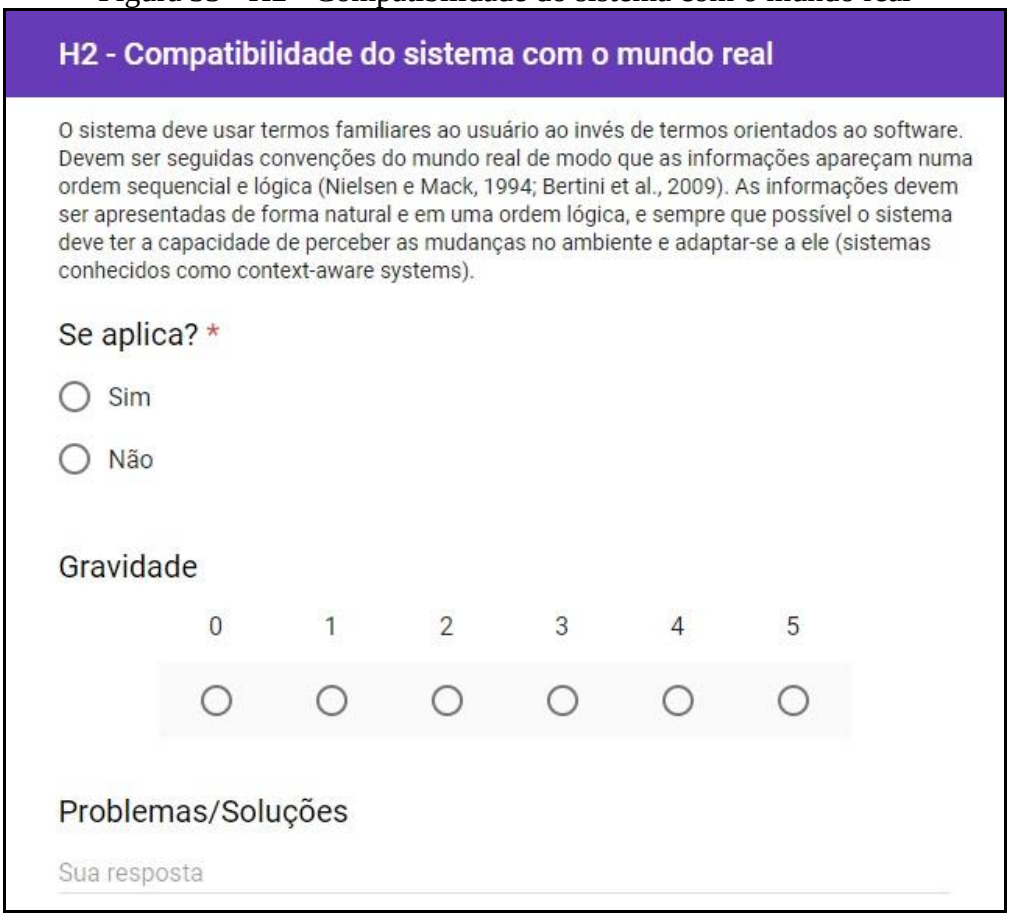
Fonte: elaborado pelo autor.

Figura 32 - H1 - Visibilidade do estado do sistema



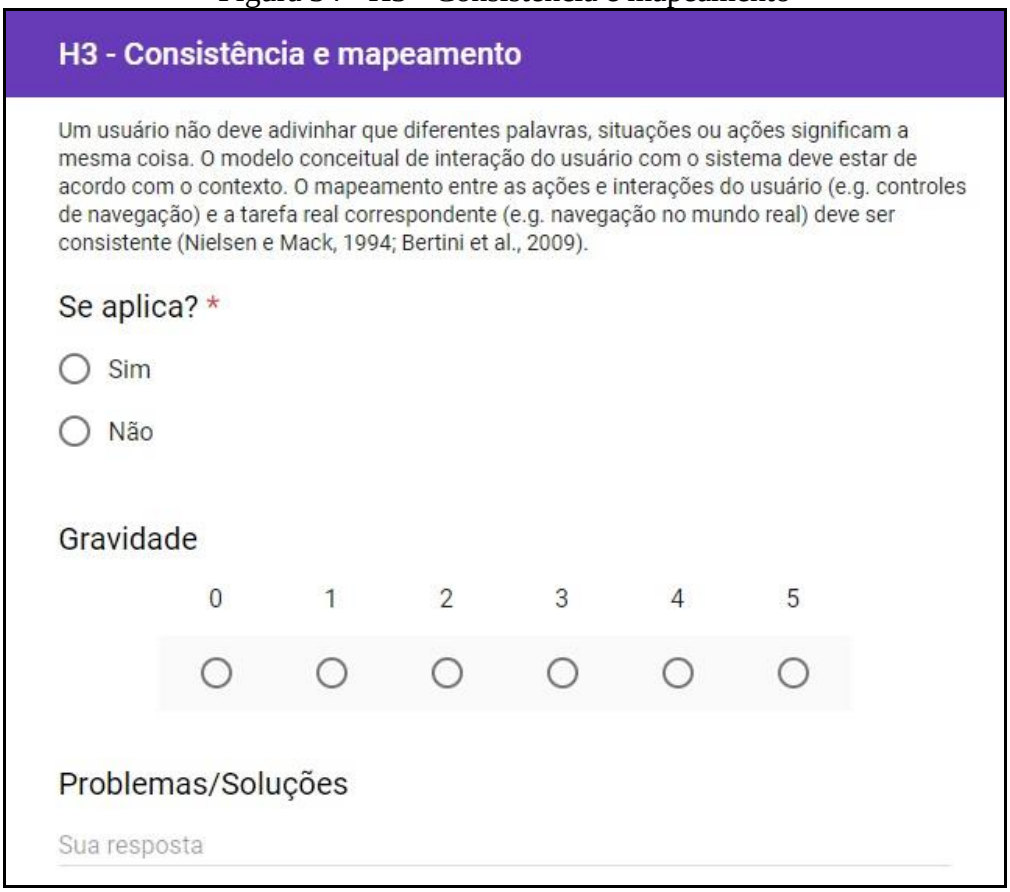
Fonte: elaborado pelo autor.

Figura 33 - H2 - Compatibilidade do sistema com o mundo real



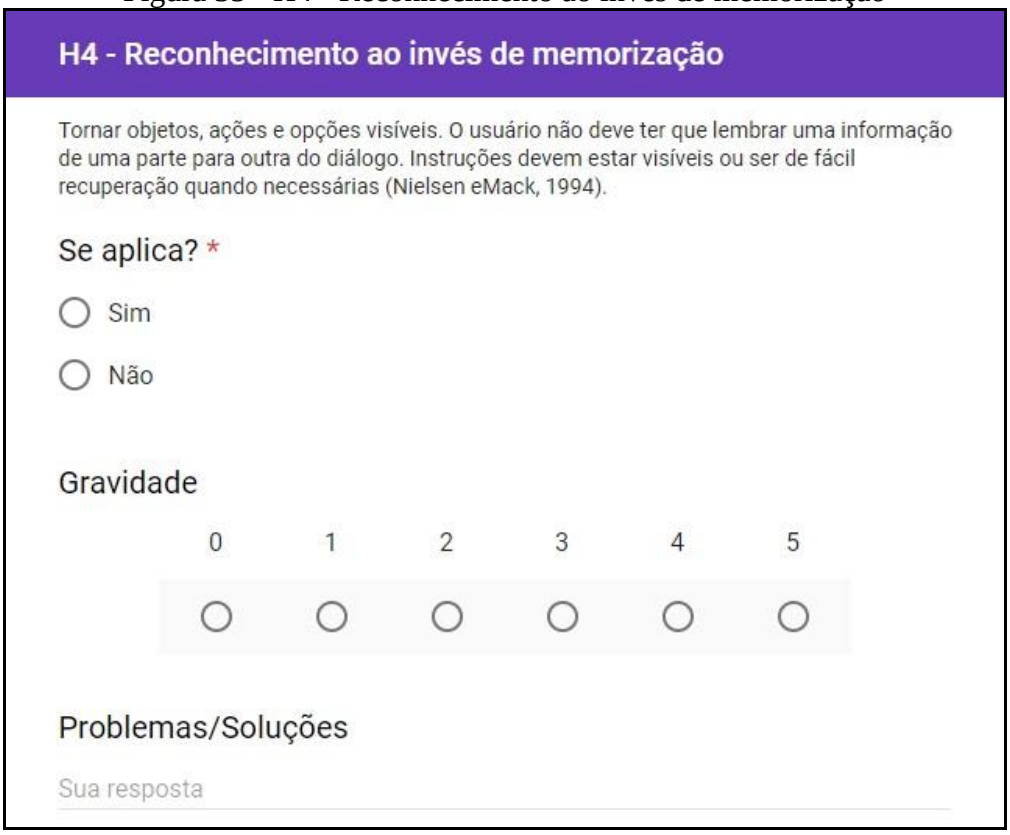
Fonte: elaborado pelo autor.

Figura 34 - H3 - Consistência e mapeamento



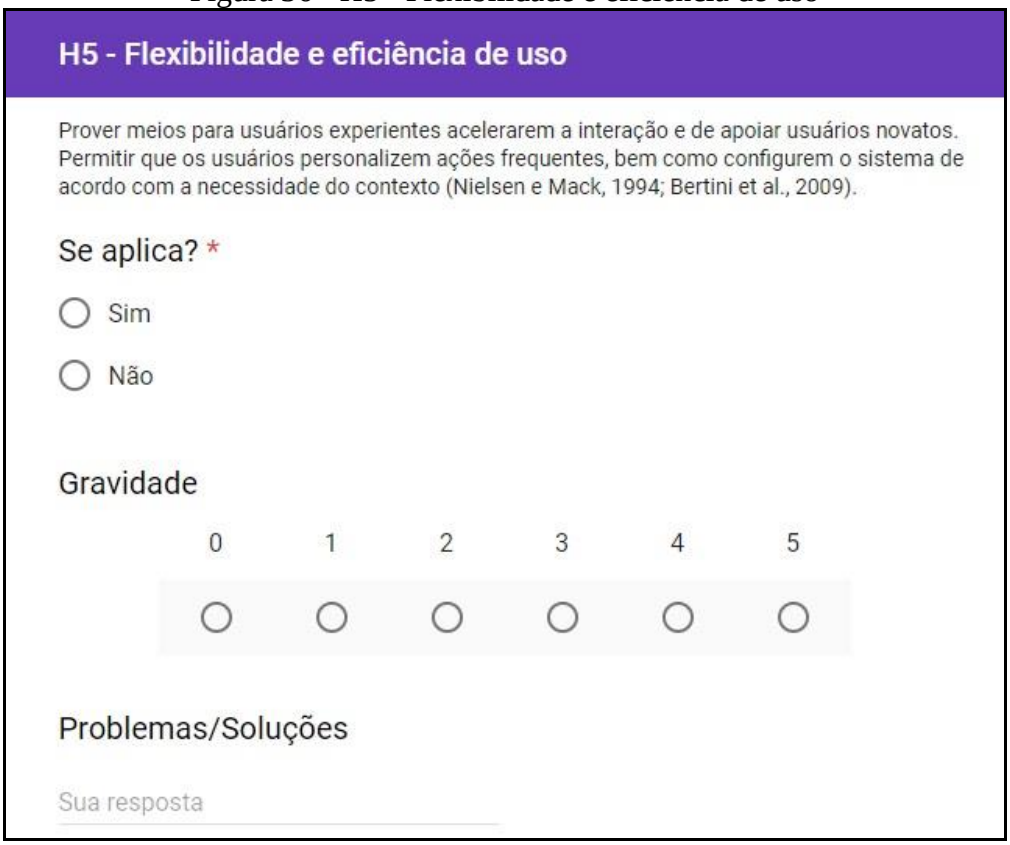
Fonte: elaborado pelo autor.

Figura 35 - H4 - Reconhecimento ao invés de memorização



Fonte: elaborado pelo autor.

Figura 36 - H5 - Flexibilidade e eficiência de uso

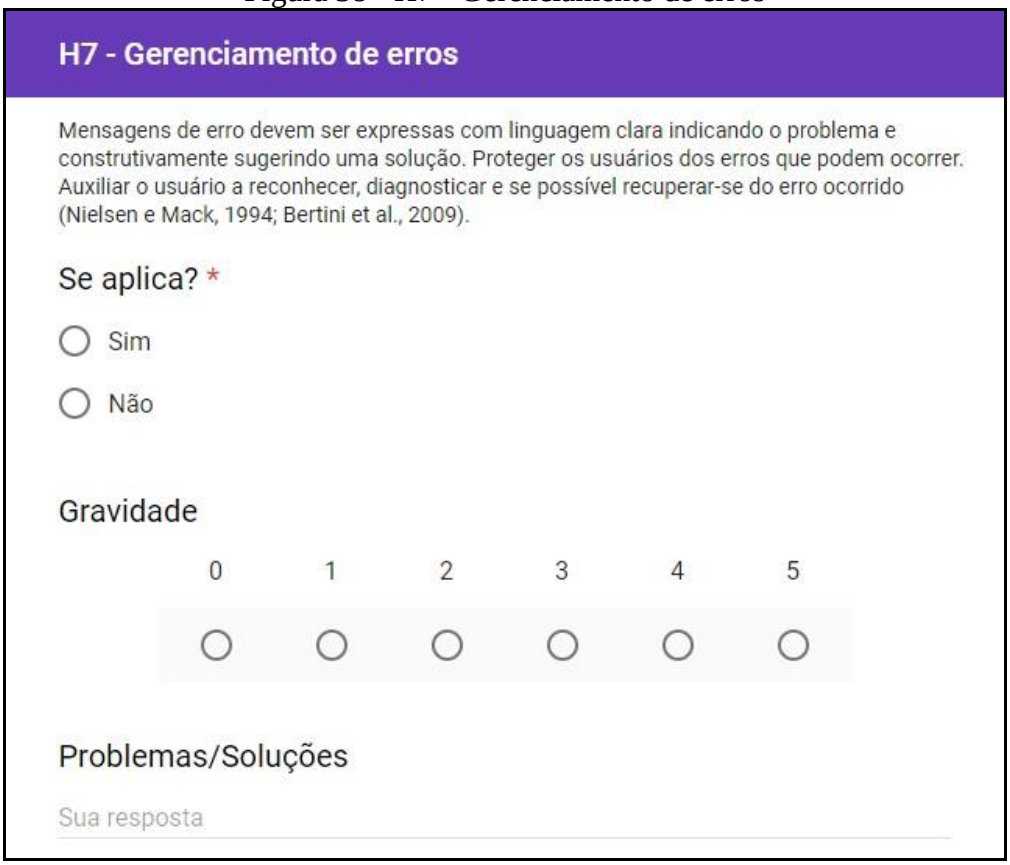


Fonte: elaborado pelo autor.

Figura 37 - H6 – Design estético e minimalista

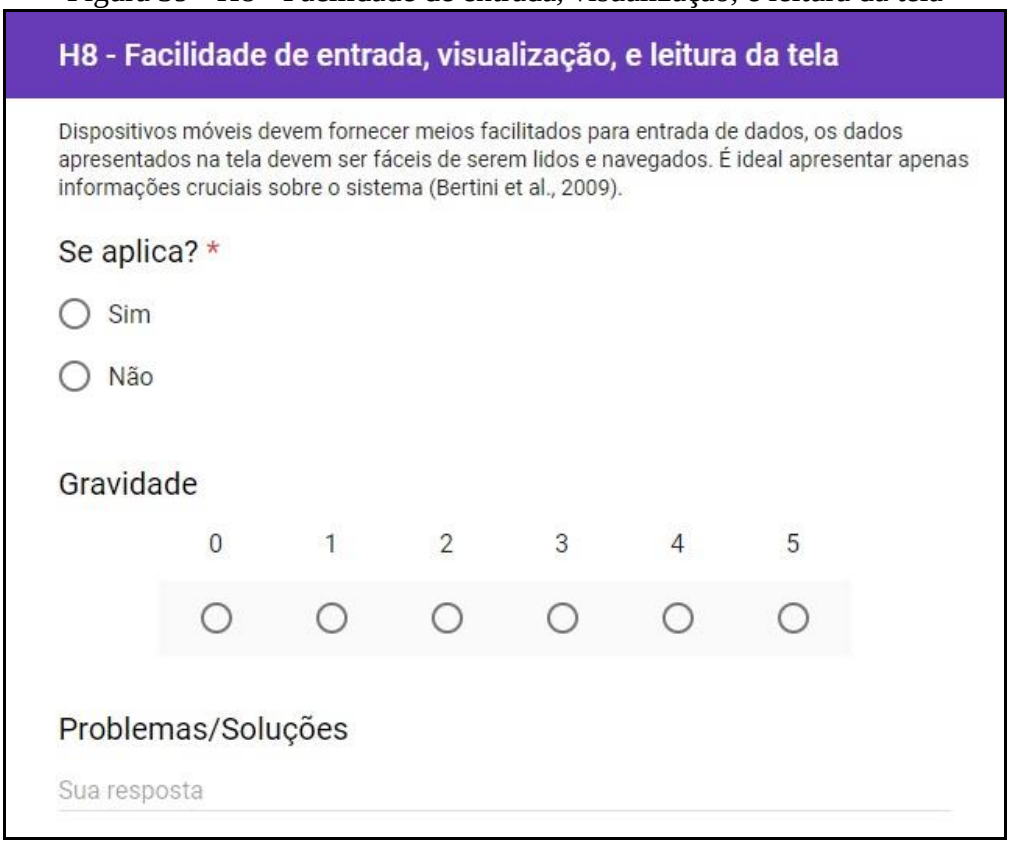
Fonte: elaborado pelo autor.

Figura 38 - H7 - Gerenciamento de erros



Fonte: elaborado pelo autor.

Figura 39 - H8 - Facilidade de entrada, visualização, e leitura da tela

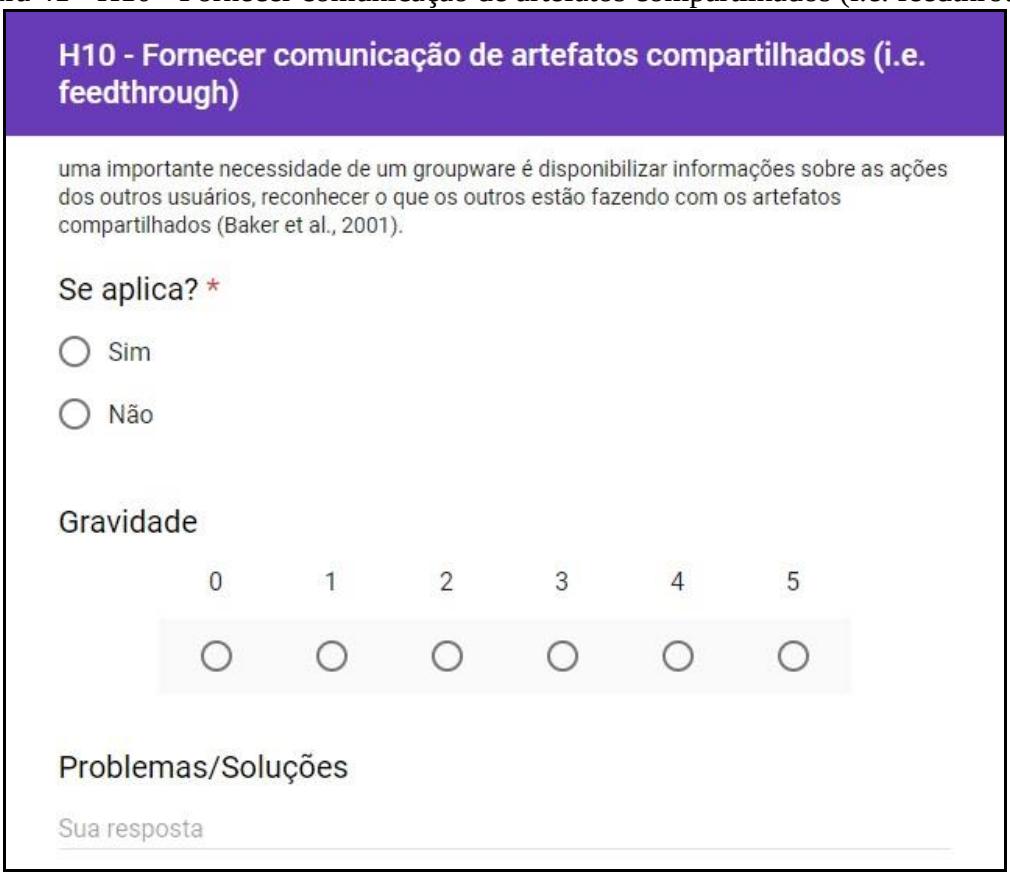


Fonte: elaborado pelo autor.

Figura 40 - H9 - Convenções estéticas, sociais e privativas

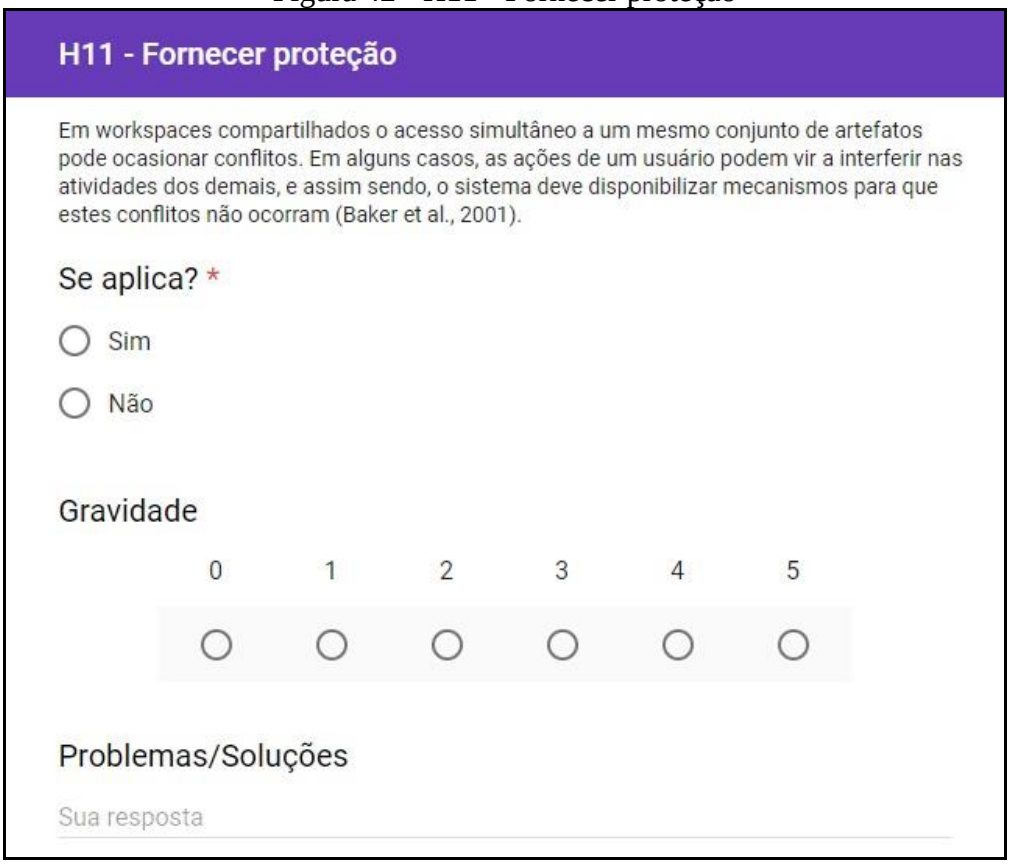
Fonte: elaborado pelo autor.

Figura 41 - H10 - Fornecer comunicação de artefatos compartilhados (i.e. feedthrough)



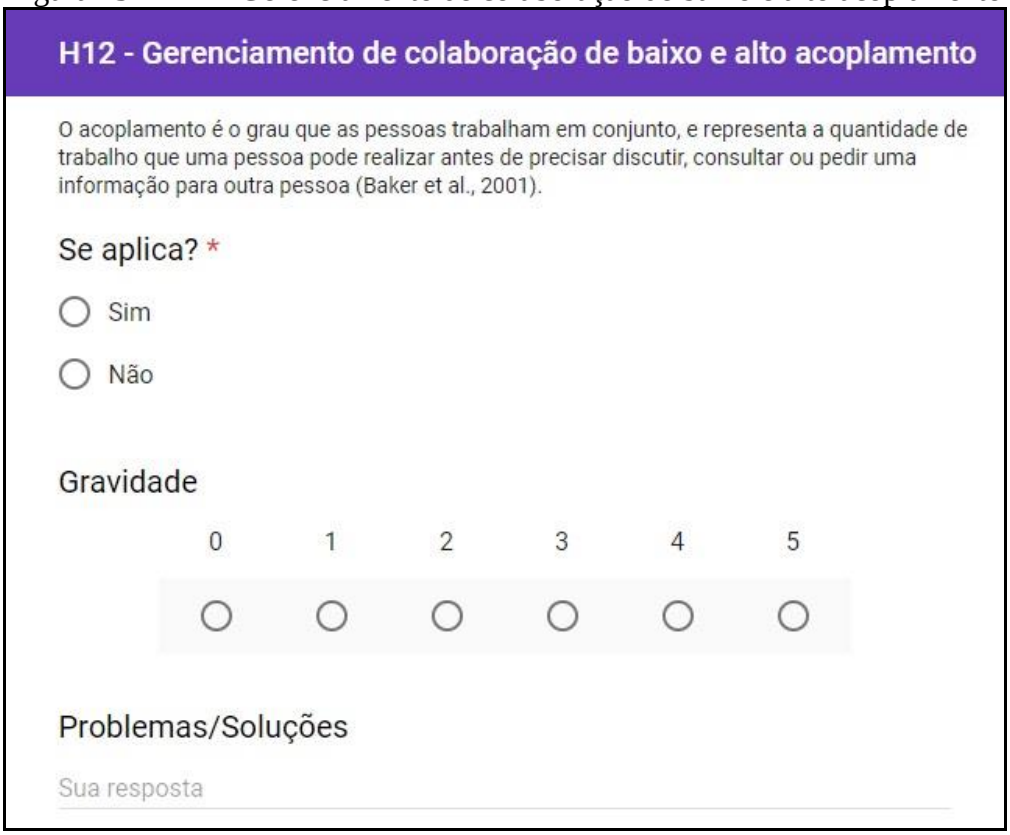
Fonte: elaborado pelo autor.

Figura 42 - H11 - Fornecer proteção



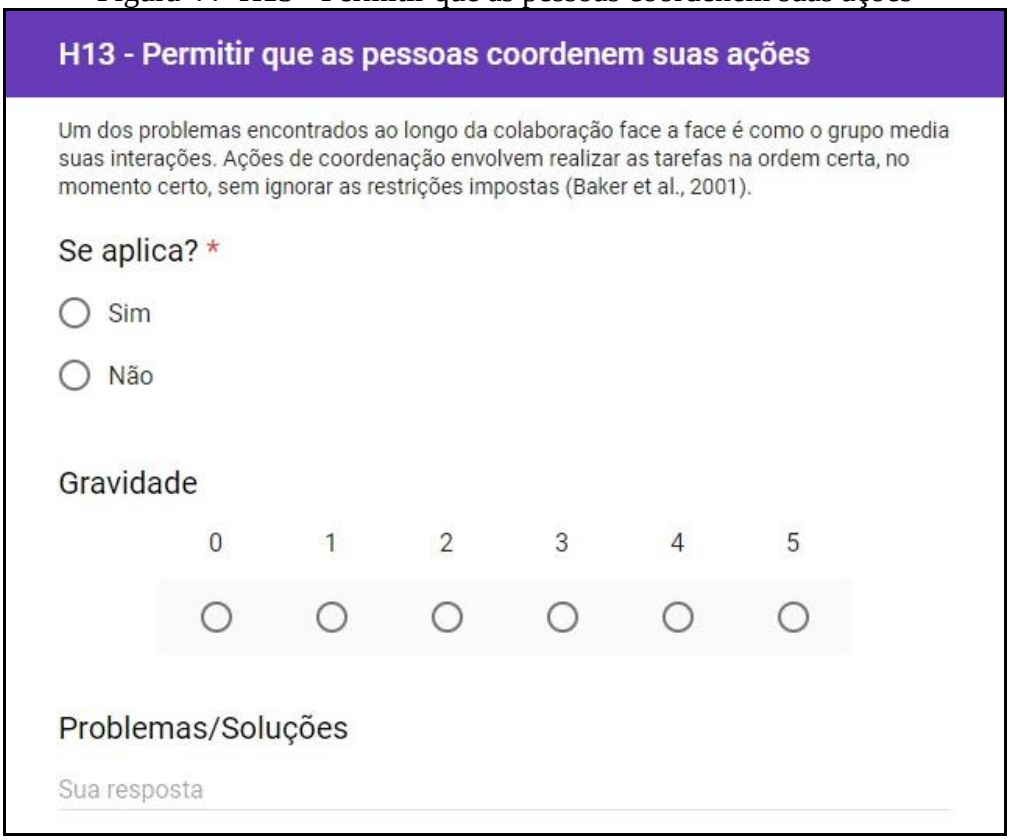
Fonte: elaborado pelo autor.

Figura 43 - H12 - Gerenciamento de colaboração de baixo e alto acoplamento



Fonte: elaborado pelo autor.

Figura 44 -H13 - Permitir que as pessoas coordenem suas ações



Fonte: elaborado pelo autor.