GABRIEL BRANDÃO DE LIMA

Orientador: Ricardo Augusto Rabelo Oliveira

APLICATIVO DE PRESTAÇÃO DE SERVIÇOS RELACIONADOS À CIDADE UNIVERSITÁRIA

Ouro Preto

Abril de 2021

Universidade Federal de Ouro Preto
Instituto de Ciências Exatas e Biológicas
Departamento de Computação
Bacharelado em Ciência da Computação

APLICATIVO DE PRESTAÇÃO DE SERVIÇOS RELACIONADOS À CIDADE UNIVERSITÁRIA

Monografia apresentada ao Curso de Bacharelado em Ciência da Computação da Universidade Federal de Ouro Preto como requisito parcial para a obtenção do grau de Bacharel em Ciência da Computação.

GABRIEL BRANDÃO DE LIMA

Ouro Preto
Abril de 2021

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

FOLHA DE APROVAÇÃO

Gabriel Brandão de Lima

Aplicativo de Prestação de Serviços Relacionados à Cidade Universitária

Monografia apresentada ao Curso de Ciência da Computação da Universidade Federal
de Ouro Preto como requisito parcial para obtenção do título de Bacharel em Ciência da Computação

Aprovada em 22 de Abril de 2021.

Membros da banca

Ricardo Augusto Rabelo Oliveira (Orientador) - Doutor - Universidade Federal de Ouro Preto
Carlos Frederico M. C. Cavalcanti (Examinador) - Doutor - Universidade Federal de Ouro Preto
Conrado Carneiro (Examinador) - Graduado - Use Mobile

Ricardo Augusto Rabelo Oliveira, Orientador do trabalho, aprovou a versão final e autorizou seu depósito na Biblioteca Digital de Trabalhos de Conclusão de Curso da UFOP em 22/04/2021.

Documento assinado eletronicamente por **Ricardo Augusto Rabelo Oliveira**, **PROFESSOR DE MAGISTERIO SUPERIOR**, em 27/04/2021, às 14:46, conforme horário oficial de Brasília, com fundamento no art. 6º, § 1º, do Decreto nº 8.539, de 8 de outubro de 2015.

A autenticidade deste documento pode ser conferida no site http://sei.ufop.br/sei/controlador_externo.php?acao=documento_conferir&id_orgao_acesso_externo=0 , informando o código verificador **0162967** e o código CRC **43837A63**.

R. Diogo de Vasconcelos, 122, - Bairro Pilar Ouro Preto/MG, CEP 35400-000
Telefone: 3135591692 - www.ufop.br

Resumo

Aplicativos para dispositivos móveis estão se tornando cada vez mais consumidos pelas pessoas, no intuito de agregar valor e facilitar suas vidas. A economia de compartilhamento, trouxe a percepção de integração entre serviço sob demanda e economia de plataforma, ou seja, economia sob demanda. Empresas como *Uber*, *iFood* e *Spotify*, abstraíram a junção desses conceitos para criar o seu tipo de serviço sob demanda, agregando valor e facilitando a vida de várias pessoas ao redor do mundo, fazendo com que se tornassem empresas de sucesso.

No ambiente universitário, não percebe-se nenhum tipo de aplicativo que tenha o mesmo intuito como esse para os estudantes, agregando valor e facilitando suas vidas durante esta fase. A ideia principal deste trabalho, é desenvolver um aplicativo Android que possa amparálos, de modo a funcionar como um *marketplace*[1] de serviços. A plataforma seria responsável por conectar todos esses estudantes e divulgar seus serviços, fazendo com que facilite a vida de quem procura um serviço e quem oferta um serviço.

O trabalho será dividido em três etapas para sua implementação. Revisões bibliográficas, para fins de estudo e conhecimento, desenvolvimento do projeto e análise de resultados. Após o término do aplicativo, o protótipo inicial foi satisfatório, com muitas críticas positivas dos usuários de teste, mostrando que seu propósito foi atingido.

Palavras-chave: Aplicativos; Android; Economia de Compartilhamento; Serviços; Universidade.

[1] do inglês, pode-se traduzir como "mercado".

Abstract

Mobile apps are getting increasingly consumed by people, in order to add value to their lives and make it better. The sharing economy, brought the perception of integrating on-demand service and platform economy, that is, on-demand economy. Companies like Uber, iFood and Spotify, abstracted the combination of these concepts to create their own type of on-demand services, aggregating value and making life easier for many people around the world, which made them successful companies.

At the university environment, it is not common any type of application with the same purpose, of adding some value and making students lives better during this phase. The main idea of this work is to develop an Android application, which will support them as a services marketplace. The platform would be responsible for connecting every student and publicizing their services, making it easier for those looking for a service and those offering a service.

The work will be divided into three stages for its implementation. Bibliographic reviews, for study and knowledge purposes, project development and analysis of results. After the application had finished, the initial prototype was satisfactory, with many positive feedbacks from test users, showing that its purpose was achieved.

Keywords: *Applications; Android; Services; Sharing Economy; University.*

Dedico este trabalho de conclusão de curso a tudo e à todos que de alguma forma contribuíram direta, ou indiretamente, para a minha jornada até aqui, cada um sabe a importância do seu papel na minha vida. Principalmente à toda minha família. Família Pasargadeana e à eterna República Pasárgada, verdadeiro significado de lar. À UFOP e ao DECOM pelo ensino gratuito de qualidade e em especial ao Conrado e ao Patrick, e à toda Usemobile pela oportunidade de crescimento e aprendizado pessoal e profissional.

Sumário

Lista de Figuras

Capítulo 1

Introdução

Ao longo da última década, a indústria 4.0 apresentou um crescimento exponencial devido ao grande avanço tecnológico e industrial. O desenvolvimento da área engloba as principais inovações tecnológicas dos campos de automação, controle e tecnologia da informação (SILVEIRA, 2018) e, se caracteriza, por um conjunto de tecnologias que permitem a fusão do mundo físico, digital e biológico. (BRASIL, 2013)

Com base nesses fatos, percebe-se que este *boom* tecnológico provocou a necessidade de expansão do acesso à informação para mais pessoas. Devido à evolução da rede celular, popularmente conhecida como 3G e 4G, concomitante ao desenvolvimento dos *smartphones*, ambos na área de tecnologia móvel, isto foi possível. (BERGHER, 2020)

De acordo com a BBC (ORGAZ, 2019), a Coreia do Sul, o Reino Unido e determinados estados dos EUA, já desfrutam da rede móvel 5G. Esta proporciona uma velocidade de transmissão de dados de até 20 Gbps por segundo, tornando tais transmissões praticamente instantâneas. Conforme a mesma fonte, tal tecnologia estará disponível no Brasil a partir de 2020. A operadora de telecomunicações Oi, realizou um teste com essa rede durante o Rock in Rio deste ano, segundo (COSSETI, 2019) . A capacidade da rede era de aproximadamente 30 Gb/s e foram trafegados mais de 173.000 terabytes de dados durante todo o evento.

Segundo a Fundação Getúlio Vargas (FGV EASP, 2019), em sua 30ª Pesquisa Anual do Uso de TI nas Empresas deste ano, o número de dispositivos móveis no Brasil já ultrapassa o número de habitantes do país, sendo cerca de 2 ou mais aparelhos por pessoa. São aproximadamente 230 milhões de celulares, contra 210 milhões de pessoas, segundo o censo do IBGE de 2019 (IBGE, 2019).

Aplicativo, ou *app*, é o nome dado para a solução tecnológica que abrange os conceitos supracitados, tornando-se em produtos digitais que estão revolucionando o mercado. Utilizam do potencial tecnológico que os *smartphones* nos proporcionam, como câmera, bússola, acelerômetro, giroscópio, acesso à geolocalização do dispositivo, para implementar formas de facilitar a vida das pessoas em grande escala.

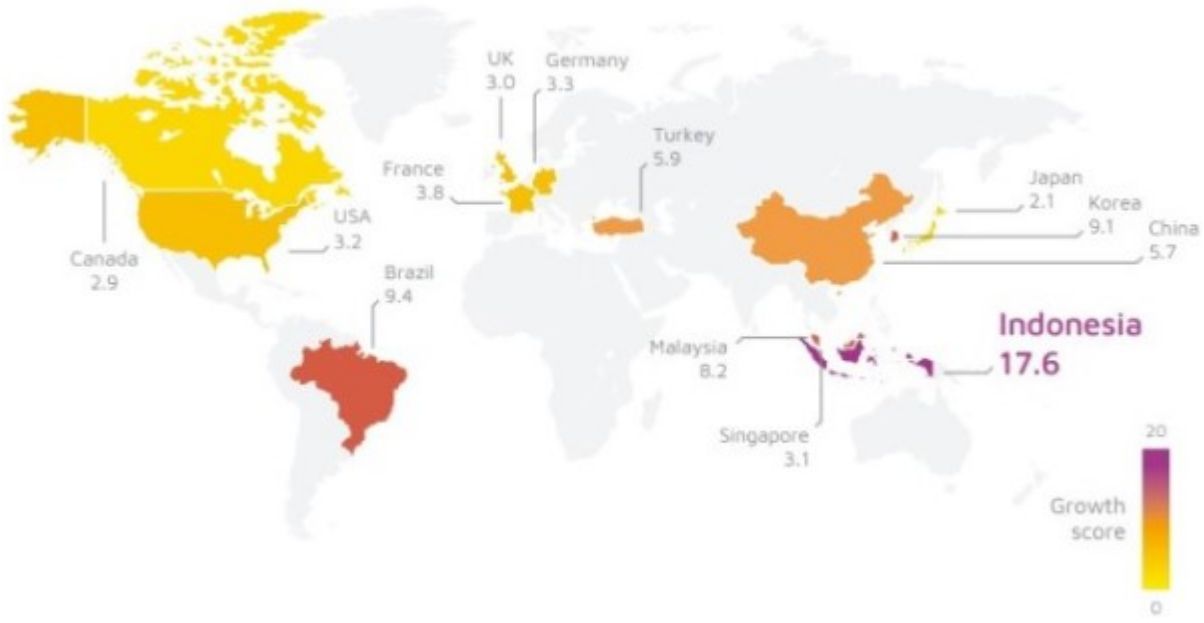

Figura 1.1: Índice de crescimento de aplicativos por país.
Fonte: (ADJUST, 2019)

Em 2018, o número de downloads dos 1000 aplicativos mais bem classificados do mundo, ultrapassou 194 bilhões. Esse levantamento foi feito pela Adjust, empresa de análise e prevenção de fraudes do setor (ADJUST, 2019). A empresa criou o índice de crescimento de aplicativos, que é definido como o número de instalações por mês, dividido pelo número de usuários mensais ativos. Os países que obtiveram o maior crescimento no índice foram, Indonésia, Brasil , Coreia e Malasia, sendo o Brasil em segundo colocado com uma pontuação de 9.4, em uma escala de 0 a 20, como observado na Figura 1.1.

Estes fatos requerem uma adaptação urgente do mercado para com tais tecnologias. Dessa forma pode-se estar acessível ao consumidor independente de seu local, ocasionando a expansão de produtos e serviços sob demanda e facilitando sua aquisição pela disponibilidade imediata na palma da mão.

1.1 Justificativa

O ingresso no ensino superior é uma transição que traz potenciais repercussões para o desenvolvimento dos jovens estudantes. Um dos fatores é que, atualmente, muitos deles saem das casas dos pais para morarem em outra cidade, ou estado, a fim de cursar uma universidade. Essa experiência pode ser recebida de duas formas: como algo difícil, em virtude de se sentirem sozinhos; e também como algo importante, devido à independência conquistada.

Nesse sentido, o ambiente universitário pode ser bastante desafiador para o estudante que, em muitos casos, estará morando fora de casa pela primeira vez. Essa ruptura de laços, traz consigo a necessidade de adaptação à esse meio e a criação de responsabilidade para com seus deveres acadêmicos, pessoais e/ou profissionais.

Tal adaptação torna-se mais árdua à medida em que o jovem vai se deparando diante a fatores como:

> "[...] a dificuldade de adaptação à universidade, o afastamento dos familiares, [...] a relação pessoal e interpessoal face às novas amizades e professores, ao próprio ambiente acadêmico e em relação ao estudo que deve a partir deste momento, assumir a responsabilidade de conciliar o tempo para se dedicar a vida acadêmica com sua vida social". (PORTO e SOARES, 2017), p.14.

A perda do contato diário com as figuras parentais traz a exigência de desenvolverem um senso maior de ter responsabilidade por si mesmo. Além disso, as atividades rotineiras de manutenção de uma casa, seja pelas atividades domésticas ou pelos gastos financeiros, demandam com que os jovens tenham a real perspectiva da importância de um suporte e aprendizado pessoal.

Sendo assim, tendo em vista o ambiente descrito e aplicando ao contexto universitário da Universidade Federal de Ouro Preto e de seus discentes, foi criado um aplicativo para ajudar o estudante a passar por esta etapa da vida e facilitar eventuais problemas inevitáveis decorrentes do dia-a-dia.

1.2 Objetivos Geral e Específicos

- Objetivo Geral

O objetivo geral do trabalho é desenvolver uma ferramenta que possa tornar melhor a condição financeira e social do aluno. A ferramenta irá proporcionar a oportunidade do jovem arrecadar uma renda extra, ao divulgar suas habilidades para determinado serviço, seja ele manual, intelectual, ou de qualquer natureza.

Será criado um ambiente virtual de ajuda mútua entre os estudantes universitários, realizando a intermediação entre eles. Cada usuário poderá ofertar e procurar o tipo de serviço desejado, de modo que possam economizar tempo, sem precisar procurar empresas que prestam o serviço, economizar dinheiro, ao contratar o serviço de outro colega universitário e ao mesmo tempo não pagar caro por um serviço profissional e isso proporcionará a ambas as partes a geração de uma renda extra, através da troca e/ou venda de serviços.

Suponha que um estudante saiba realizar reparos elétricos, em contrapartida, outro saiba instalações e reparos mecânicos, como máquinas lavadoras de roupas, ambos podem trocar o serviço prestado, ou comprar a hora de serviço do seu colega universitário por acordo estabelecido entre as partes via aplicativo. Posteriormente, ao finalizar o produto pretende-se disponibilizar para os estudantes da UFOP, através de uma versão de testes na loja da *Google Play Store*, afim de avaliar o desempenho e engajamento do aplicativo por seu principal público-alvo.

- Objetivo Específico

O objetivo específico deste trabalho é a implementação de um aplicativo para dispositivos móveis com sistema operacional Android, para estabelecer a interligação entre esses dois estudantes. Para isso foram aplicados os conhecimentos adquiridos durante todo o curso de modo a desenvolver o projeto proposto.

O foco foi desenvolver toda a arquitetura necessária para criação do aplicativo, contando com *front-end*[1] e *back-end*[2].

Para lançar o produto final, deve-se estudar a melhor maneira de atingir o público-alvo do aplicativo, levando em conta como melhorar todos os aspectos funcionais, práticos e a melhor usabilidade possível para o mesmo.

[1] Camada visível para o usuário final
[2] Camada não visível para o usuário final

1.3 Método e Organização do Trabalho

O trabalho foi dividido em quatro partes para compreensão e entendimento do produto proposto. Levando em conta, o estudo de referenciais teóricos, o levantamento das ferramentas e serviços necessários para implementação do aplicativo, especificação do escopo do projeto e a implementação do mesmo.

Para os referenciais teóricos, foram feitas a pesquisa de trabalhos, ferramentas, serviços e produtos existentes diretamente relacionados ao tema do trabalho. Após estes levantamentos, foi feita a escolha dos principais *frameworks* e recursos utilizados. Ao definir todo o escopo e recurso necessário, em seguida, deu-se início ao desenvolvimento do trabalho.

Os estudos feitos para a implementação do projeto levaram majoritariamente em conta:

- Computação Móvel
- Guia de Desenvolvimento Android Google
- Linguagem de Programação
- Comunicação com API's do Google
- Compreensão de experiência e interface de usuário - UX /UI

Posteriormente à implementação, foi realizado o teste e a validação do produto com usuários reais.

Capítulo 2

Revisão de Literatura

Para implementação do aplicativo, foi necessário conhecimento na área de desenvolvimento Android, comunicação com API's , manipulação de redes e acesso à geolocalização do dispositivo móvel. Além do conhecimento específico sobre a plataforma que executa a aplicação, dos *frameworks* que foram utilizados para o desenvolvimento, também foi necessário o conhecimento sobre o ramo e modelo de negócios em que o tema se encaixa.

2.1 Economia de Compartilhamento

A Economia de compartilhamento, segundo (FRENKEN e SCHOR, 2017), pode ser definida como:

> "consumidores que concedem um ao outro acesso temporário a ativos físicos subutilizados ou que estão com capacidade ociosa, em troca de um equivalente monetário"(p.2-3, tradução nossa)[1], ou seja, bens compartilháveis.

Estes bens compartilhados, muitas vezes podem se tornar algo ocioso para o consumidor, visto que o mesmo não utiliza desse produto durante todo o tempo. Isso lhe dá a oportunidade de emprestar ou alugar seus produtos a outros consumidores.

A noção de compartilhamento de capacidade ociosa é central para a definição de economia de compartilhamento, porque distingue a prática de compartilhamento de bens da prática de serviços pessoais sob demanda.

A noção de subutilização também é fundamental para a discussão atual sobre plataformas de compartilhamento doméstico, como o Airbnb. Quando o proprietário de uma casa está ausente durante férias, uma viagem de negócios, ou possui um quarto de hóspedes, o ativo não é utilizado. Ou seja, a casa desocupada pode ser considerada como capacidade ociosa temporária. Se, no entanto, uma pessoa comprasse uma segunda casa e alugasse para turistas

[1]Texto original: *"consumers granting each other temporary access to under-utilized physical assets ("idle capacity"), possibly for money."*

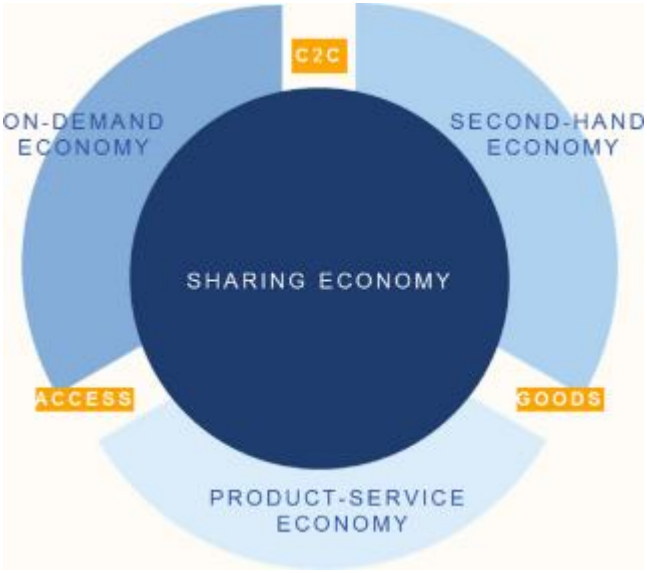

Figura 2.1: Economia de compartilhamento e formas relacionadas de economia de plataforma. Fonte: (FRENKEN e SCHOR, 2017)

permanentemente, isso constituiria a administração de um site de hospedagem comercial, como um *Bed and Breakfast* - B&B, ou hotel.

Com base nessa definição, a economia de compartilhamento pode ser diferenciada de três outros tipos de plataformas que estão compartilhando exemplos de economia anteriores à Internet. Na Figura 2.1, a economia do compartilhamento é colocada no centro, pois adere às três características que a definem: a interação consumidor-consumidor (c2c), acesso temporário e bens físicos. O trabalho aqui proposto se encaixa entre acesso temporário e c2c, na figura, esse nicho é referenciado como "economia sob demanda".

Finalmente, de acordo com (MARTIN, 2016), a importância da economia de compartilhamento se dá a partir do momento em que, indivíduos que trabalham ou ganham dinheiro com negócios frutos deste modelo, são celebrados como microempreendedores. Além disso a própria economia de compartilhamento é também definida como: uma inovação disruptiva do modelo de negócios impulsionada pelas tecnologias digitais; e, uma grande oportunidade comercial para empreendedores, empresas, indústrias e / ou países. Seu crescimento contínuo é enquadrado como desejável ou necessário com base em que: ela promove o empoderamento econômico individual; as tecnologias digitais que impulsionam a economia compartilhada a tornam inevitável; a economia compartilhada é uma resposta à mudança dos padrões de comportamento do consumidor e do trabalhador; e, a mesma promove uma utilização mais eficiente dos recursos de ambos.

2.2 Consumo Colaborativo

O Consumo Colaborativo (CC) , segundo (HAMARI et al., 2015), é nomeado assim por envolver o compartilhamento de bens e serviços através de atividades, o qual está posicionado na economia compartilhada como um fenômeno tecnológico. A prestação de serviços sob demanda, como dito anteriormente, se encaixa no modelo de economia sob demanda e também pode se encaixar no conceito de consumo colaborativo.

Devido ao grande desenvolvimento da tecnologia da informação, houve uma simplificação do compartilhamento de bens e serviços tanto físicos, quanto não-físicos. Pode-se enxergar o CC não apenas como consumo, mas como uma atividade em que tanto a contribuição, quanto o uso de recursos, estão entrelaçados por meio de redes ponto a ponto.

O mesmo autor ainda define o termo CC como a atividade ponto a ponto de se obter, dar ou compartilhar acesso a bens e serviços, coordenada por meio de serviços online baseados na comunidade, sendo assim um site on-line, um aplicativo móvel ou uma combinação usada e mantida continuamente pelos usuários. Portanto, também podemos encaixar o projeto proposto aqui no conceito de consumo colaborativo, uma vez que este se encaixa nas definições supracitadas.

2.3 Computação Móvel

A computação móvel veio à tona com o crescimento do uso de computadores portáteis e redes sem fio, trouxe a necessidade de desenvolver aplicativos e soluções específicas em que pudessem atender as demandas nesta área. O termo pode ser definido como o acesso à informação em qualquer lugar, a qualquer momento e com qualquer equipamento. Porém, temos também a computação pervasiva, que é o acesso à informação distribuída no ambiente de forma imperceptível e perceptível ao usuário. (MATSUMOTO, 2005)

O avanço da tecnologia de armazenamento em nuvem, a redução de custo da fabricação de componentes de hardware, bem como a expansão da tecnologia de telefonia celular, redes móveis e em redes sem fio, permitiu que praticamente todas as áreas de negócios e serviços utilizem os sistemas de dispositivos móveis e seus mais diversos componentes integrados (acelerômetro, bússolas, entre outros). (MINDS, 2017)

> "Computação móvel pode ser representada como um novo paradigma computacional que permite que usuários desse ambiente tenham acesso a serviços independentemente de sua localização, podendo inclusive, estar em movimento. Mais tecnicamente, é um conceito que envolve processamento, mobilidade e comunicação sem fio. A idéia é ter acesso à informação em qualquer lugar e a qualquer momento."(FIGUEIREDO e NAKAMURA, 2003)

2.4 Desenvolvimento Android

Segundo a (GLOBALSTATS, 2019), o sistema operacional Android é o mais utilizado no mundo hoje, como observamos na Figura 2.2, atingindo em Outubro de 2019 cerca de 76.67% de todo o mercado de sistemas operacionais para dispositivos móveis. Um dos fatores para que isso tenha ocorrido deve-se aos dispositivos em que o utilizam terem uma ampla variedade de marcas e preços, o que facilita ao consumidor de classe média e baixa a sua aquisição. Isso torna o Android um sistema operacional relevante para o trabalho, o qual possibilitará o uso do aplicativo proposto aqui pela maior parte dos usuários de dispositivos móveis.

Aplicações Android são escritas majoritariamente em linguagem de programação Java. As ferramentas do kit de desenvolvimento de *software* para Android (Android SDK - *Software Development Kit*), compilam todo o código escrito em um conjunto de dados e recursos, gerando assim um único arquivo cuja extensão possui a nomenclatura ".apk", um pacote Android. Esse arquivo possui todas as funcionalidades implementadas pelo desenvolvedor e o conteúdo da aplicação. São esses arquivos que os dispositivos móveis desenvolvidos para Android utilizam para instalar o aplicativo. O conjunto de ferramentas SDK, é composto por uma biblioteca, um emulador, exemplos de códigos com classes definidas por padrão, documentação, tutoriais e um depurador de código, assim o desenvolvedor tem todo o caminho para a construção de um aplicativo Android. (GOOGLE, 2019)

2.4.1 Linguagem de Desenvolvimento Android

O Android pode ser escrito em três tipos de linguagens orientadas a objetos, sendo elas: Kotlin, Java e C++. Dentre elas, a escolhida para esse projeto foi o Kotlin, por ter sido introduzida mais recentemente e ser reconhecida como linguagem oficial de desenvolvimento Android pela Google. É semelhante ao Java, mas é tida pela comunidade de desenvolvedores como mais acessível de se entender. (COMPUTERWORLD, 2019)

2.4.2 JAVA

Java é a linguagem nativa de desenvolvimento Android e a mais utilizada ainda hoje, por ser uma das mais familiares entre todos os desenvolvedores Android. Sua sintaxe a torna simples e compreensível o que consequentemente também torna fácil sua implementação. (COMPUTERWORLD, 2019)

2.4.3 C++

O Android Studio - a ser descrito na subseção 3.2.1 - suporta C ++ com Java NDK[2]. Esta linguagem permite codificação nativa, tornando-a muito útil para projetos como jogos.

[2] *Native Development Kit* - Kit de desenvolvimento nativo

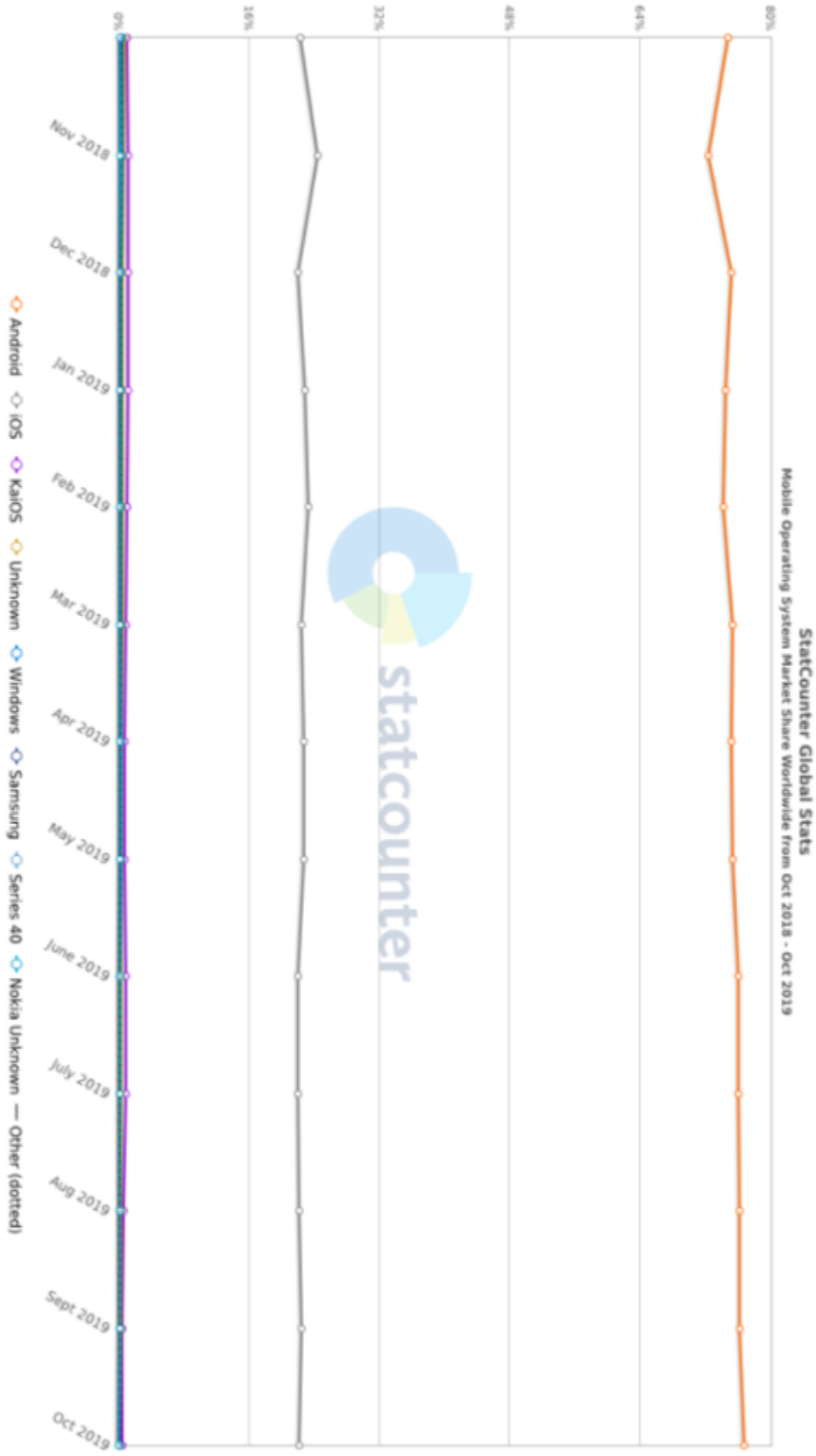

Figura 2.2: Participação no mercado dos sistemas operacionais móveis em todo o mundo. Fonte: (GLOBALSTATS, 2019)

Vale ressaltar, entretanto, que o C ++ pode ser um pouco mais complicado para iniciantes. (COMPUTERWORLD, 2019)

2.4.4 Kotlin

O Kotlin é uma linguagem moderna, multi plataformas e de vários propósitos. A Jetbrains iniciou seu desenvolvimento em 2010 e viu sua popularidade crescer rapidamente. Teve sua primeira aparição no ano de 2011 e lançamento oficial em 2016. Kotlin é um projeto *open source* gratuito licenciado pelo Apache 2.0. É uma linguagem baseada na programação funcional e altamente tipada. Sua sintaxe é mais concisa e expressiva em comparação ao Java. Possui funções de alta ordem, permite expressões lambda e por ser multi paradigma pode-se desenvolver funcionalmente ou orientado a objetos. (GRIN, 2019)

Foi anunciado recentemente, pela Google, como a linguagem mais recomendada para desenvolvimento Android (TRIPATHI, 2019). Possui um ótimo desempenho ao desenvolver aplicativos Android, trazendo todas as vantagens das linguagens modernas, sem impor nenhuma restrição.

Algumas vantagens da linguagem:

- Compatibilidade: O Kotlin é totalmente compatível com o JDK 6, garantindo que os aplicativos Kotlin possam ser executados em dispositivos Android mais antigos sem problemas;
- Performance: Um aplicativo Kotlin é executado tão rápido quanto um Java equivalente, graças à estrutura de bytecode muito semelhante. Com o suporte do Kotlin para funções embutidas, o código usando lambdas geralmente é executado ainda mais rápido que o mesmo código escrito em Java;
- Interoperabilidade: O Kotlin é 100% interoperável com Java, permitindo usar todas as bibliotecas Android existentes em um aplicativo Kotlin, incluindo o processamento de anotações;
- Segurança: Valida valores nulos em tempo de compilação, para evitar exceções em tempo de execução;
- Curva de Aprendizado: Para um desenvolvedor Java, é muito fácil iniciar o Kotlin. O conversor automatizado de Java para Kotlin incluído no plug-in Kotlin ajuda nas primeiras etapas. O Kotlin Koans oferece um guia sobre os principais recursos do idioma com uma série de exercícios interativos. (KOTLIN ORG, 2019)

Grandes empresas já adotaram a linguagem como preferencial e compartilharam suas experiências. Como no *app* do Android *Basecamp*, onde refatoraram 100% do código para Kotlin,

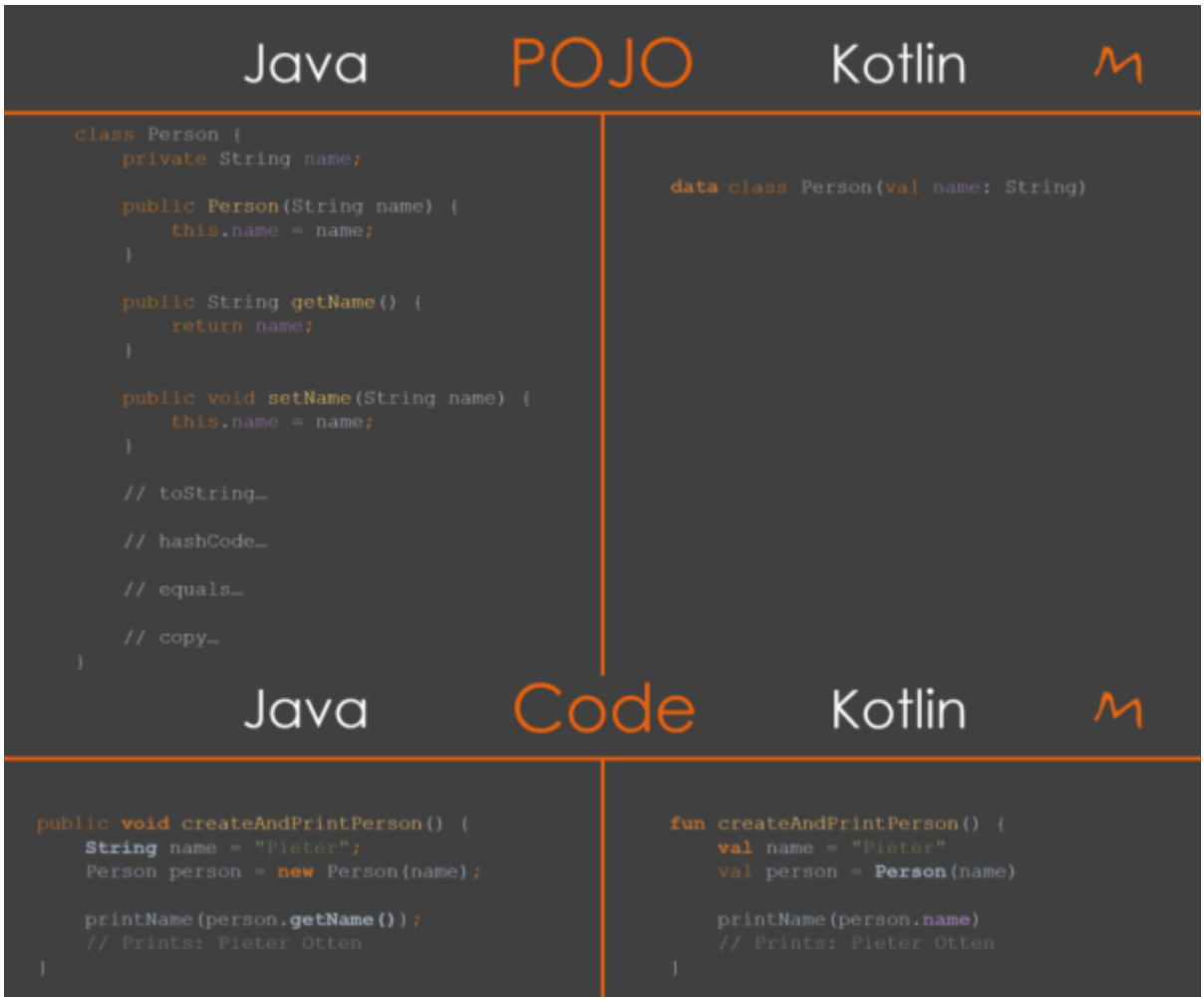

Figura 2.3: Kotlin x Java.
Fonte: (PAUL, 2018)

e relatam uma enorme diferença e grandes melhorias na qualidade e velocidade do trabalho, assim como melhor disposição do desenvolvedor para programar.

Pela Figura 2.3 pode-se observar a grande diferença ao se criar uma classe de objeto. Toda a necessidade de declaração de construtores, *getters* e *setters*, que são funções responsáveis por retornar ou atribuir um valor a cada atributo do objeto, como ***getName()*** e ***setName(String name)*** para manipulação do objeto Java da figura, é abstraída em um tipo de objeto em Kotlin chamado de *data class object*. Esse objeto pode receber acessos de *getters* e *setters* diretamente devido à linguagem Kotlin ser estaticamente tipada. Simplesmente ao declarar o nome da classe e seus construtores, todas as variáveis pertencentes ao objeto podem ser manipuladas. Para instanciar uma variável que é imutável utiliza-se a declaração ***val***, e para variáveis mutáveis a declaração ***var***. Além disso, no Kotlin, ao se declarar uma variável não é necessário informar o tipo da mesma, uma vez que a linguagem atribui tal tipo à variável

Figura 2.4: Arquitetura do Sistema Operacional Android.
Fonte: (SMIEH, 2019)

em tempo de compilação, assim como em JavaScript. Isso pode ser observado na figura pela diferença entre a construção da função ***createAndPrintPerson()*** ao ser escrita em ambas linguagens.

Com isso, vemos que o Kotlin é uma linguagem moderna de alto nível, que possui dos melhores recursos de programação funcional, como funções de alta ordem - funções que recebem ou retornam outras funções, exemplos: funções *map*, *fold*, *filter* e *find*, usadas para transformar, agregar, filtrar e encontrar, respectivamente, um elemento ou conjunto de dados - e orientada a objetos, permitindo ao desenvolvedor uma maior produtividade, rendimento e ao mesmo tempo reduzindo códigos clichês. (NORONHA, 2013)

2.4.5 Arquitetura do Sistema Operacional

Android, é um sistema operacional móvel *open source*[3], baseado no *Kernel* Linux, em que cada aplicativo é um usuário diferente. O sistema atribui a cada aplicativo um único código de usuário Linux, onde somente aquela aplicação tem conhecimento. Esse código é utilizado pelo sistema operacional e é desconhecido pelo aplicativo. O sistema define as permissões necessárias para todos os arquivos em um aplicativo , de tal maneira que somente a aplicação com aquele código de usuário específico possa acessar.

O código de um aplicativo é executado isoladamente de todos os outros , uma vez que cada processo tem a sua própria máquina virtual. Nativamente, cada aplicação executa seu

[3]do inglês, *open-source software* pode-se traduzir como software de código aberto.

próprio processo Linux. O Android inicia o processo quando qualquer dos componentes de uma aplicação precisa ser executado, e desliga o processo quando o mesmo não é mais necessário ou quando o sistema deve recuperar memória para outras aplicações.

Podemos ver pela Figura 2.4, que este possui uma arquitetura de *software* composta por múltiplas camadas, sendo elas:

- Aplicativos na camada superior;
- *Frameworks* necessários para processar os aplicativos;
- O sistema Android de tempo de execução e a biblioteca executando uma versão da *Java Virtual Machine (JVM)*, o qual é baseado no *OpenJDK* desde 2015;
- Bibliotecas e APIs escritas em C;
- E o *Kernel* do Linux na parte inferior

2.4.6 Fundamentos do sistema

O Android é a base para execução de cada aplicativo instalado em seu sistema e controla seus processos e funcionalidades. Ele implementa o princípio do privilégio mínimo, que torna a execução de cada aplicativo independente de qualquer outro. Desse modo, cada aplicativo ao ser executado, tem acesso somente aos componentes e recursos do dispositivo que são necessários para seu funcionamento. Isso traz uma maior segurança para o sistema em si e também para o usuário, no que tange sua privacidade e seus dados pessoais.

Esse gerenciamento de concessão de recursos de hardware e software feito pelo sistema, é controlado por meio de permissões. Cada aplicativo possui sua lista de permissões necessárias para seu funcionamento, isto é, todos os recursos de hardware do dispositivo que necessita ter acesso para funcionar corretamente. Entretanto, existem formas de dois ou mais aplicativos compartilharem informações para utilizarem dos mesmos serviços do sistema. O usuário do dispositivo necessita conceder explicitamente as permissões necessárias, antes de utilizar a funcionalidade correspondente a mesma.

2.4.7 Componentes de aplicativo

Um *app* Android tem seu alicerce de construção encima dos componentes de aplicativo, figura 2.5. Todo o comportamento geral do *app* é definido a partir de como será feita essa construção. Certos componentes dependem um do outro e estes são um ponto de entrada por onde o sistema, ou usuário, pode entrar no aplicativo.

Cada componente representa uma entidade independente e cada tipo de componente tem um ciclo de vida específico. Esse ciclo de vida define a criação e destruição do componente

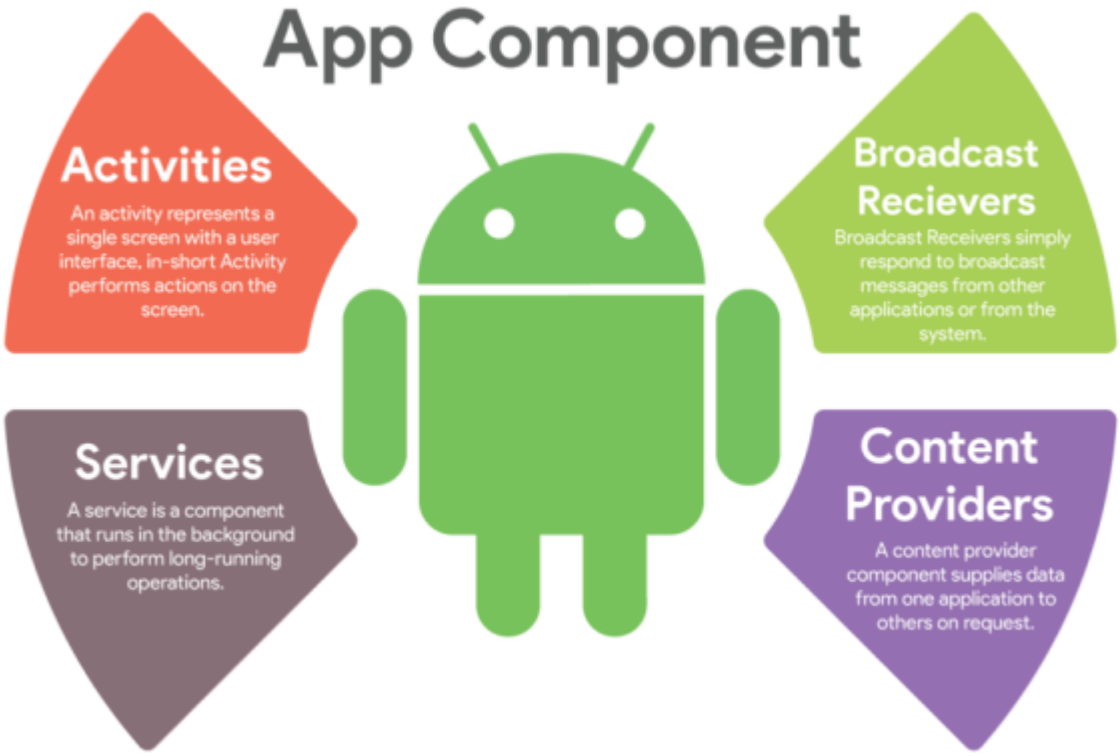

Figura 2.5: Componentes de Aplicativo Android.
Fonte: (GATTAL, 2018)

específico, determinando quando será instanciado e/ou destruído, seja para liberação de memória ou quando tiver finalizado seu processamento.

Os aplicativos Android são criados como uma combinações de componentes que podem ser invocados individualmente. Por exemplo, uma atividade é um tipo de componente do aplicativo que oferece uma interface de usuário.

Os componentes de aplicativo são classificados em quatro entidades principais:

- Atividades (*Activities*): Principal componente do aplicativo. Possuem uma única interface de usuário, definida por um *layout* previamente criado. As atividades são a porta de entrada para o aplicativo e as responsáveis por delegar todo o controle de fluxo de telas e coordenar a interação com o usuário.

  Analogamente, um aplicativo de banco pode ter uma atividade que mostra os lançamentos financeiros no extrato da conta, outra atividade que realiza uma transferência e outra que realiza pagamentos. Apesar de serem atividades pertencentes ao mesmo aplicativo, são funcionalidades diferentes, ou seja, é possível iniciar cada atividade separadamente a partir de um outro aplicativo, basta que o aplicativo do banco seja configurado para permitir essa inicialização externa. Um aplicativo de leitor de código de barras, por

exemplo, pode iniciar o aplicativo do banco para realizar o pagamento de um código de barras de um boleto lido por ele.

Uma atividade facilita muitas interações sistema-aplicativo, dentre elas temos as principais:

  - Acompanhamento do fluxo de interação do usuário, para saber o que realmente importa para ele. Dessa forma garante que o sistema continue executando este processo o qual é responsável por hospedar a atividade em questão;
  - Estabelecer uma pilha de processos prioritários a serem mantidos, sabendo quais atividades foram mais recentemente utilizadas e podem conter informações pertinentes às quais o usuário pode querer retornar;
  - Gerenciamento de memória e ciclo de vida, quanto aos processos interrompidos, para que o usuário possa restabelecer às atividades com o estado anterior restaurado.
  - Fornecer uma maneira dos aplicativos implementarem fluxos de usuários entre si e para o sistema coordenar esses fluxos. Sendo essa o tipo de interação sistema-aplicativo mais comum entre elas.

- Serviços (*Services*): Os serviços são componentes inicializados para processar tarefas em segundo plano, sejam elas de execuções longas ou rápidas, ou até para processos remotos. Eles mantém um aplicativo em execução em segundo plano, seja qual for o motivo. Não possuem interface de usuário, pois são chamadas de processos realizadas pelo aplicativo para que o sistema execute.

  Uma requisição de dados, por exemplo, é feita em segundo plano ao acessar um aplicativo de banco e selecionar a opção de visualizar o saldo. Para o aplicativo recuperar o saldo que o usuário possui, o mesmo necessita realizar uma conexão com o banco de dados e fazer uma chamada solicitando a ele para que lhe forneça qual o saldo do usuário. Após receber a resposta da chamada, o aplicativo realiza o processamento da informação e exibe para o usuário. Tudo é feito em segundo plano, de modo transparente para o usuário. Execução de músicas, downloads e sincronização de dados, todos esses serviços são feitos em segundo plano , graças a esse componente.

- Receptores de Transmissão/Notificação (*Brodcast Receivers*): Os receptores de transmissão permitem ao sistema entregar eventos ao aplicativo fora do fluxo padrão do usuário. Respondem às notificações a nível de sistema. Ou seja, independente de qual momento da aplicação esteja sendo executado em primeiro plano, ela receberá eventos externos para que possa responder a essas notificações.

Os receptores de notificações são mais uma porta de entrada para um aplicativo. Podem inicializar o mesmo, a partir de uma transmissão, mesmo que ele não esteja em execução. Apesar de não possuírem inerface de usuário, podem criar notificações na barra de status do sistema Android, para informar ao usuário sobre o evento que a aplicação recebeu, tendo já realizado alguma ação ou não.

Muitas transmissões se originam do sistema operacional, como alertas de bateria baixa, captura e bloqueio de tela, alarmes, entre outros. Aplicativos de alarmes e lembretes, podem programar notificações para alertar ao usuário sobre um evento futuro. Quando um alarme é ativado, o aplicativo não precisa permanecer em execução até o mesmo ser desativado. Trivialmente, um *broadcast receiver* é somente um *"gateway"* para qualquer outro componente do sistema. Nativamente, toda transmissão é feita intencionalmente de modo a realizar a menor quantidade de trabalho possível. Por exemplo, um serviço pode ser iniciado, para executar alguma tarefa, logo após receber um evento de transmissão.

- Provedores de Conteúdo (*Content Providers*): Um provedor de conteúdo gerencia um conjunto de dados compartilhados do aplicativo. Esses dados podem estar armazenados no banco de dados local do dispositivo do sistema, em um servidor remoto, em um banco de dados SQL - *Simple Query Language*, na web, ou qualquer outro local de armazenamento persistente acessível ao seu aplicativo. Através desse componente, outra aplicação ou serviço, pode acessar esses dados e realizar todos os tipos *CRUD*[4] de manipulações encima deles, porém o provedor de conteúdo deve permitir cada tipo de ação.

  Por exemplo, o sistema fornece um provedor de conteúdo para gerenciar informações de contatos do usuário. Sendo assim, qualquer outro aplicativo que detém das permissões necessárias para acessar esse tipo de conteúdo, poderá consultar parte desse provedor e ler ou escrever dados de alguma pessoa em particular.

  Content providers também são úteis para ler e escrever dados privados e que não são compartilhados da sua aplicação, como salvar notas pessoais ou visualizar fotos.

Uma funcionalidade única do aspecto de projeto do sistema Android, é que toda aplicação pode inicializar um componente de uma outra aplicação independente a ela, como gravar um vídeo, ou capturar uma imagem. Todo dispositivo que embarca uma câmera, possui um aplicativo nativo de captura de imagens e gravações de vídeos. Esse aplicativo pode ser inicializado por um outro aplicativo, de modo que o primeiro realize a captura de uma imagem e, ao concluir, envie-a ao último para posterior processamento, permitindo que utilize tal imagem em função de executar uma tarefa interna. Ou seja, caso o aplicativo que esteja sendo desenvolvido, necessite de capturar uma imagem para um fim específico, não é necessário

[4]Abreviação de: Create, Read, Update, Delete - Criar, Ler, Editar, Excluir

criar uma atividade e implementar esta funcionalidade internamente, basta utilizar o próprio aplicativo de câmera nativo. Em seguida, somente é necessário chamar o *app* de câmera a partir do aplicativo que necessita capturar a imagem e toda essa computação será transparente ao usuário.

Dentre os componentes de aplicativo referenciados acima, os três primeiros - Atividades, Serviços e Receptores de Transmissão - são ativados por Intenções, ou *Intents*. Uma Intenção é uma mensagem assíncrona que delibera uma ″intenção″ de executar determinada tarefa. *Intents* conectam componentes individuais com outros componentes em tempo de execução e requisitam uma ação de outros componentes, quer estes pertençam à sua aplicação ou não.

Quando o sistema ativa um componente, ele inicia o processo para aquela aplicação, somente caso ele ainda não esteja rodando, instanciando as classes necessárias para o componente. Tomando o exemplo da captura de imagem, a aplicação que solicitou o aplicativo da câmera é executada no processo da própria câmera e não do aplicativo solicitante. Ao contrário de outros programas e sistemas, aplicativos Android não possuem um único ponto de entrada para ser inicializado, como uma função ***main()***.

Um aplicativo não consegue ativar diretamente um componente de outro aplicativo, em virtude do sistema rodar cada programa em um processo independente. Cada processo detém as permissões de arquivos que restringem seus acessos às outras aplicações. Não obstante, o sistema Android é capaz de executar essa ação. Isso é possível mediante o uso de *Intents*. Estas, anteriormente supracitadas, despacham ao sistema uma mensagem especificando qual o componente específico necessário a ser inicializado. Imediatamente, o sistema ativa o componente.

Diferentemente dos componentes citados acima, um Provedor de Conteúdo não é ativado via intenção, mas sim quando impelido por uma requisição de um Resolvedor de Conteúdo, ou *Content Resolver*. O Resolvedor de Conteúdo gerencia todas as transações diretas entre o componente o qual requisitou a informação e o Provedor de Conteúdo. Por consequência, é estabelecida uma camada de abstração, onde toda a informação trafegada durante esta computação se torna mais segura, dado que o componente o qual está executando as transações com o provedor, não necessita realizar essa tarefa e, em oposição, delega tais chamadas aos métodos do objeto Resolvedor de Conteúdo.

Diante do exposto acima, para que um componente de aplicação seja inicializado pelo sistema Android, o mesmo necessita saber quais componentes exitem naquele aplicativo específico. O arquivo **"AndroidManifest.xml"** da aplicação, é o responsável por informar ao sistema quais são esses componentes. Todos os componentes presentes em um aplicativo são declarados nesse arquivo, o qual fica sempre no diretório raíz do projeto do *app*.

Além dos componentes de aplicativos, todos *apps* Android exigem recursos separados do código fonte como imagens, cores, fontes, arquivos de áudio e vídeo, animações e tudo que se relaciona à interface gráfica do aplicativo e experiência de uso do aplicativo. Esses recursos

facilitam a adaptação à configuração de diversos dispositivos diferentes e a atualização das propriedades do aplicativo, como cor de fundo, estilo de botão e comportamento de clique na tela. Todas essas definições devem ser declaradas em arquivos de extensão XML - *Extensible Markup Language*. (GOOGLE, 2019)

2.4.8 Ciclo de Vida

Toda Atividade de uma aplicação é definida por seu ciclo de vida, em que este controla a sua criação, interrupção e destruição. Durante a navegação de um usuário pelo fluxo de um aplicativo, a Atividade mantém salva seu estado de vida. A classe *Activity* fornece vários retornos de chamada que informam a mudança de estado da Atividade. Estes *callbacks* permitem saber se o processo hospedeiro da atividade está sendo criado pela primeira vez, re-criado, parando, ou destruindo, bem como os estados de transição entre cada um destes, como observado na figura 2.6.

Nos métodos de retorno de chamada do ciclo de vida, é possível saber qual estado a atividade se encontra e para qual estado irá em seguida. Isso permite ao desenvolvedor declarar qual será o comportamento da atividade, quando o usuário sair e retornar para a aplicação. Um aplicativo reprodutor de vídeos, por exemplo, poderá controlar seu comportamento de reprodução e utilização da rede utilizando desses conceitos. Quando o usuário alternar entre aplicativos, a reprodução é pausada e o tráfego de dados interrompido, ao retornar para o reprodutor, o vídeo é retomado de onde parou e o tráfego de dados para carregar o vídeo também pode ser reestabelecido. Ou seja, cada retorno de chamada permite executar, e controlar, as tarefas apropriadas a serem processadas pelo sistema e pelo aplicativo, com base em uma determinada mudança de estado. Fazer o trabalho certo no momento certo e manipular as transições adequadamente torna o aplicativo mais robusto e com melhor desempenho.

O conjunto principal dos retornos de chamada que a classe *Activity* fornece, para navegar pelas transições entre os estágios de seu ciclo de vida, são seis, sendo eles: ***onCreate()***; ***onResume()***; ***onPause()***; ***onStop()***; ***onDestroy()***. O sistema chama cada um desses retornos de chamada quando uma atividade entra em um novo estado.

Uma boa implementação dos retornos supracitados, pode ajudar a garantir que seu aplicativo evite muitos problemas, tais como:

- Parar de funcionar, enquanto estiver usando um aplicativo, caso o usuário receba uma ligação, ou alterne para outro *app*;
- Consumir recursos valiosos do sistema operacional, quando o usuário não estiver os usando ativamente;
- Perder o progresso do usuário, caso ele saia do aplicativo e retorne posteriormente, em um cadastro por exemplo;

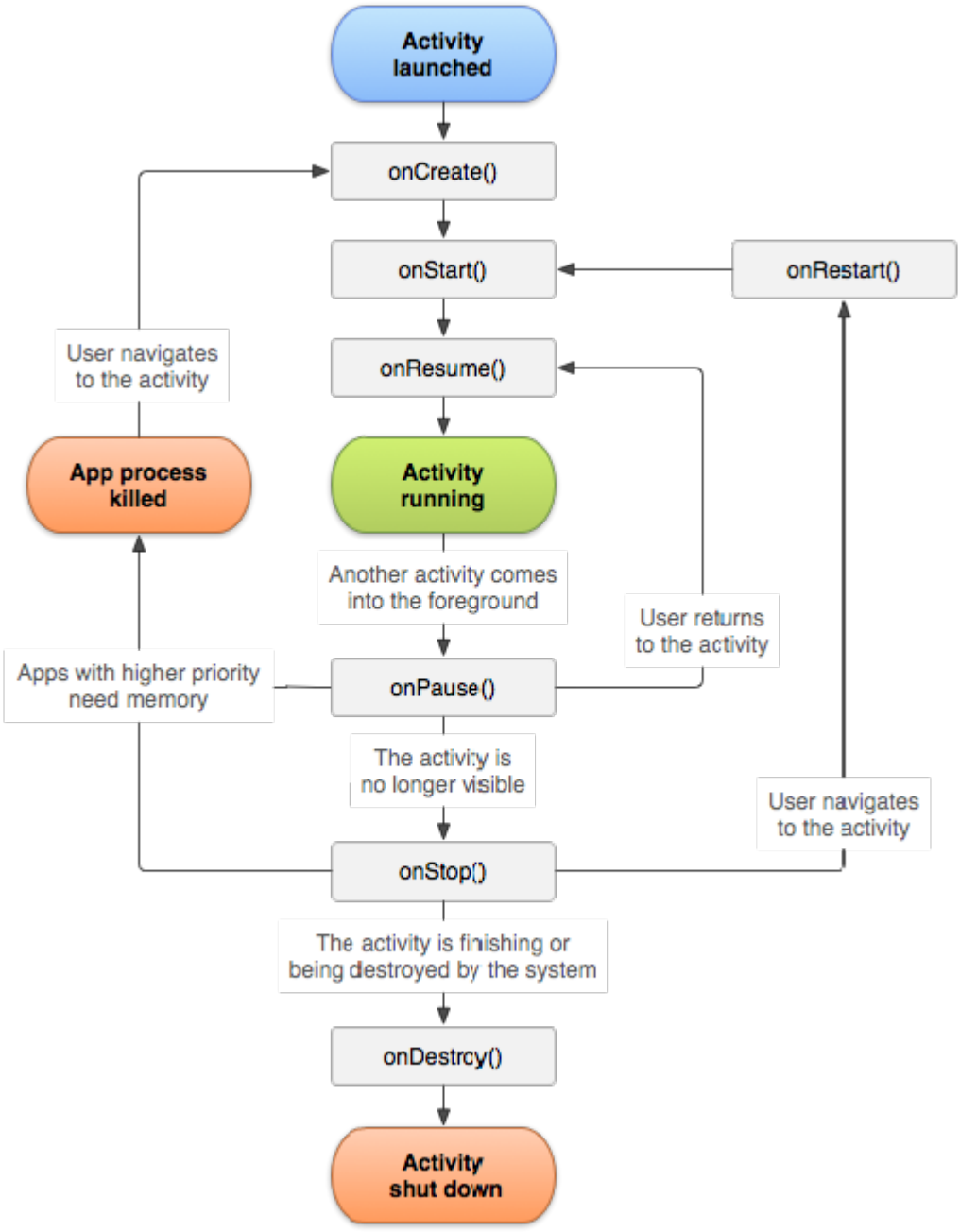

Figura 2.6: Ciclo de vida básico de uma Atividade.
Fonte: (GATTAL, 2018)

Figura 2.7: Grupo de Componentes do Android Jetpack.
Fonte: (GOOGLE DEVELOPERS, 2018)

- Parar de funcionar, ou perder o progresso do usuário, quando a tela girar entre a orientação paisagem e retrato.

Não se importar com o ciclo de vida de cada atividade pode causar muitos problemas que, consequentemente, venham a acarretar em tempo perdido tentando os resolver, ou até mesmo não conseguir. (GATTAL, 2018)

2.4.9 Android *Jetpack*

O Android *Jetpack* é um conjunto de componentes, ferramentas e orientações, desenvolvido para acelerar o processo de implementação de aplicativos Android, colaborando para um desenvolvimento mais simples, rápido e intensificando o desempenho de produção. Foi inspirado na biblioteca de suporte do Google, que consiste em um grupo de componentes criado para auxiliar o uso dos novos recursos do Android, mantendo a compatibilidade com versões anteriores do sistema. Aplicações implementadas utilizando esse agrupamento de recursos, se tornam mais robustas e eficientes.

Equitativamente, o Android *Jetpack* pode ser definido como: o conjunto da biblioteca de suporte, uma nova biblioteca de extensões para Kotlin e os componentes de arquitetura Android, que foram remodelados para obter melhor performance. A união de todos eles formam esse modelo-único de "entidade", chamada de *Jetpack*. Esses componentes reúnem a antiga Biblioteca de Suporte, os Componentes de Arquitetura existentes e os organizam em

quatro categorias, conforme Figura 2.7. Para melhor compreensão da importância e escopo de cada uma dessas categorias, temos:

- Arquitetura: Os Componentes de Arquitetura Android, permitem ao desenvolvedor construir aplicativos robustos, testáveis e manuteníveis. De todos os recursos oferecidos nesse conjunto de componentes, os mais relevantes fornecem ferramentas para gerenciar: persistência de dados, processos e serviços em segundo plano, ciclo de vida dos componentes de interface de usuário e a navegação entre fluxos de telas.

- Interface de Usuário (UI - *User Interface*): Estes componentes fornecem estruturas e ferramentas para tornar o uso do aplicativo fácil e agradável.

- Comportamento: Componentes de Comportamento auxiliam na integração do aplicativo aos serviços padrões do Android, como notificações, permissões, compartilhamentos e o assistente de uso e navegação.

- Fundamentação - Base do *app*: Componentes de fundamentação, trazem funcionalidades tranversais, como retrocompatibilidade, fornecem recursos primaciais do sistema, uma estrutura para testes unitários de UI e são totalmente compatíveis com a linguagem Kotlin.

Apesar da entidade, como um todo, ser representada pelas quatro categorias citadas acima, todos seus componentes são independentes entre si e podem ser usados individualmente, adaptando-se conforme a necessidade do aplicativo. O Android Jetpack foi arquitetado com base em padrões de projeto modernos, como separação de problemas, desacoplamento de módulos e capacidade de testes, visando também potencializar recursos de produtividade, como suportar nativamente a integração com Kotlin.

Os componentes podem ser programados tanto em Java quanto em Kotlin, porém o pacote *Jetpack* foi otimizado visando ter sua implementação pela última linguagem. Por exemplo, o conjunto das estruturas de dados fornecidas pelo Android KTX, é uma biblioteca de extensões para Kotlin que se encaixa na categoria de Fundamentação do *app*, ou seja, é um dos alicerces para uma boa eficiência do aplicativo.

- Kotlin Simples

```
view.viewTreeObserver.addOnPreDrawListener(
    object : ViewTreeObserver.OnPreDrawListener {
        override fun onPreDraw(): Boolean {
            viewTreeObserver.removeOnPreDrawListener(this)
```

```
                actionToBeTriggered()
                return true
            }
        }
    )
```

- Kotlin + Android KTX:

```
view.doOnPreDraw { actionToBeTriggered() }
```

Ao utilizar esta coleção de bibliotecas, o desenvolvedor pode focar em desenvolver aquilo que é fundamentalmente necessário para o aplicativo em questão, como suas funcionalidades, módulos e interfaces de usuário, contando com toda a infraestrutura por trás do código fornecido pela Google. Particularidades do sistema são abstraídas e tratadas internamente, facilitando e acelerando o desenvolvimento cada vez mais. (GOOGLE DEVELOPERS, 2018, 2019a; NGULELE, 2018; SAIYED, 2018)

2.5 Google *Maps*

O Google *Maps* SDK - *Software Development Kit*, permite adicionar mapas de todo o banco de dados da Google ao aplicativo Android. Também é possível adicionar marcadores, polígonos, polilinhas, sobreposições de solo e sobreposições de bloco a um mapa básico, possibilitando a sua modelagem conforme as particularidades da aplicação. A API - *Application Programming Interface*, gerencia automaticamente o acesso aos servidores do *Maps*, a aquisição dos dados geográficos, exibição do mapa e o comportamento de resposta a toques e interações no mesmo.

Para a integração do *framework* ao aplicativo Android, deve ser feito inicialmente um cadastro para utilização dos serviços de nuvem da Google, no console de seu gerenciador de API's, chamado *Google Cloud Platform*. Após o cadastro, basta criar um novo projeto e habilitar a ferramenta escolhida para integração, no caso o Google *Maps*. Logo após, deve-se gerar uma chave única de acesso à esta API pelo projeto criado, ela será a responsável por validar as credenciais de acesso e obtenção de mapas entre o servidor da Google e a aplicação Android. (GOOGLE DEVELOPERS, 2019e,d)

2.6 *Firebase*

O Google *Firebase*, é outra ferramenta de desenvolvimento criada sobre a infraestrutura da Google, a qual realiza o escalonamento automático de aplicativos. Permite criar *apps* rapida-

mente sem precisar gerenciar a infraestrutura. Dentre suas funcionalidades as mais utilizadas são: análises métricas e estatísticas, bancos de dados, mensagens e relatórios de erros. Consegue fornecer agilidade no processamento de informações, para uma melhor mineração dos dados e controle de usuários do aplicativo.

Dentre as ferramentas do *Firebase*, o *Firebase Realtime Database* realiza a sincronização de dados JSON[5] em tempo real. É um banco de dados, NoSQL - não relacional, hospedado em nuvem que permite armazenar e sincronizar dados entre seus usuários. Sempre que algum dado é alterado no servidor do *Firebase*, todos os dispositivos que possuem um aplicativo com a instância daquele banco, recebem a atualização desses dados simultaneamente.

Totalizando a praticidade do *Firebase Realtime Database*, suas regras de segurança são estruturadas de modo que definem quando os dados podem ser lidos e gravados. O *Firebase Authentication*, outra API da Google, ao ser integrado com o *Realtime Database*, permite aos desenvolvedores delimitar quem tem acesso a quais dados e como podem acessá-los.

Sua API foi implementada de modo a autorizar apenas operações de execução rápida. Isso possibilita um melhor serviço em sincronização de dados em tempo real, abrangendo milhões de requisições sem comprometer sua eficiência e *performance*. Assim há a importância de saber analisar como os dados serão acessados pelos usuários da aplicação, de forma a organizar esses dados adequadamente no banco.

Por fim, ao utilizar o *Firebase Realtime Database* é possível criar aplicativos avançados e colaborativos, concedendo acesso seguro ao banco de dados diretamente pelo código do cliente. Todos os dados são mantidos localmente e, mesmo off-line, os eventos em tempo real continuam sendo acionados, proporcionando uma experiência responsiva ao usuário final. (GOOGLE DEVELOPERS, 2019b,c,f)

[5] *do inglês, JavaScript Object Notation* - JSON é uma formatação leve de troca de dados, em formato texto e completamente independente de linguagem, pois usa convenções que são familiares às linguagens C e familiares, incluindo C++, C#, Java, JavaScript, Perl, Python e muitas outras. Estas propriedades fazem com que JSON seja um formato ideal de troca de dados. (CROCKFORD, 2021)

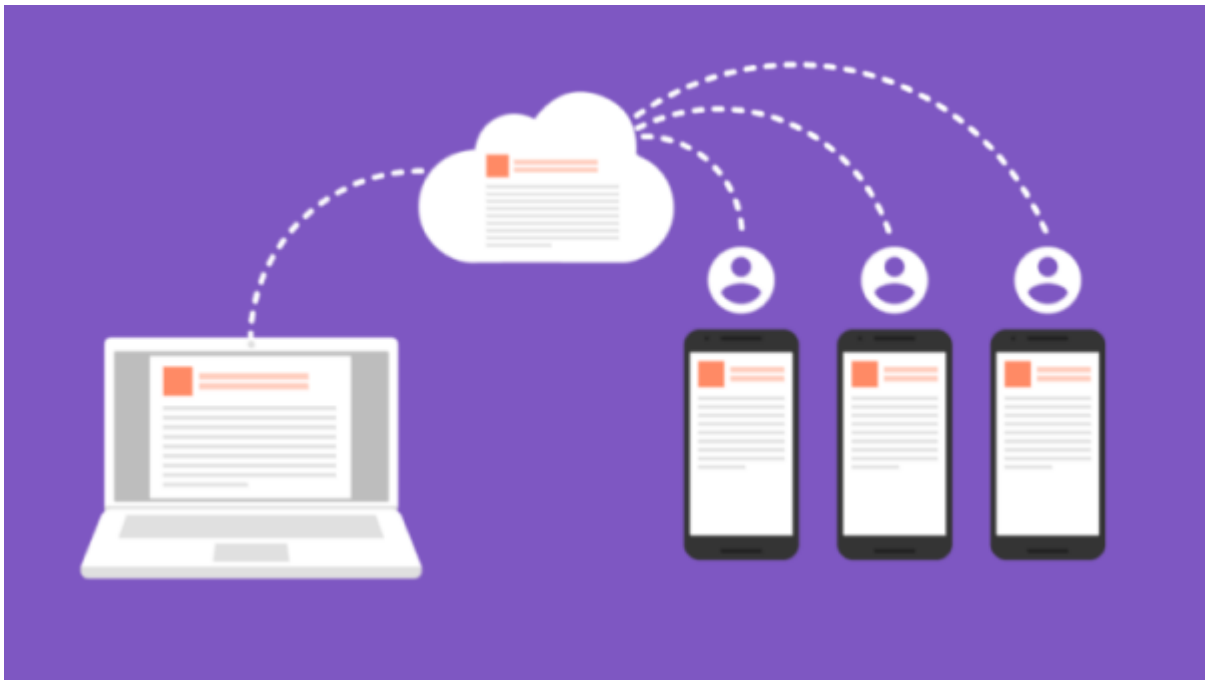

Figura 2.8: Sincronização em tempo real pelo *Firebase*, Web - Nuvem - Dispositivos.
Fonte: (GOOGLE DEVELOPERS, 2019b)

2.7 *Design* de Interface do Usuário - UI

Design de Interface do Usuário, ou *User Interface Design*, é a área do *Design* responsável por elaborar, criar e planejar a plataforma de comunicação, a qual o usuário irá interagir e controlar um determinado dispositivo. Esse dispositivo pode ser um sistema, *software*, produto, serviço, aplicativo, etc. É a única parte "visível" da aplicação a qual o usuário interage.

Todas as interações e controles são feitos por meio de uma interface. A interface pode ser composta de botões, menus e qualquer elemento visual que permita uma ação entre o usuário e o dispositivo. Para se obter uma boa interface, esta deve ser intuitiva, ou seja, simples e fácil de usar. Sua ideia principal é que o usuário consiga realizar o necessita e pretende realizar. Tudo isso proporciona uma experiência amigável.

UI *Design* não se limita somente à parte visual do projeto, mas como também à interação humano-computador, indo muito além da estética e aparência do sistema. Um bom projeto de UI, é projetado de forma a antecipar as reais necessidades de utilização do usuário. Dessa forma garante tanto a acessibilidade quanto a usabilidade do aplicativo, assim como uma boa experiência de interação, evitando ou minimizando dificuldades de utilização e frustrações ao utilizador. (ARTY, 2018)

Figura 2.9: Prototipação de Projeto Utilizando UI *Design*.
Fonte: (ARTY, 2018)

2.8 *Design* de Experiência do Usuário - UX

Diferentemente do *Design* de UI, o *Design* de Experiência do Usuário - *User Experience Design*-, lida com as emoções e experiências dos usuários. A área engloba o estudo de como as pessoas se sentem ao utilizar algo, como e quais são os sentimentos que elas tem ao utilizar um serviço, produto ou marca. Ou seja, concentra-se exclusivamente na relação do usuário com o produto ou serviço.

Enquanto a UI foca na satisfação do usuário quanto ao aspecto visual do programa, a UX foca nos **serviços** e **produtos**. Tem como objetivo oferecer um serviço, ou produto, de ótima satisfação e que garanta uma boa experiência, evitando insatisfações ao usuário. Concomitante ao *Design* de UI, o *Design* de UX também torna a experiência do usuário mais amigável, natural e simples.

Profissionais responsáveis pela UX *Design* buscam tornar a experiência do usuário em algo positivo, grandioso, agradável e que alcance as suas expectativas. As estudam e projetam com base no comportamento humano. Realizam todos os ajustes e adaptações necessárias com base, essencialmente também, nos comportamentos do público usuário em questão. Sendo assim, o UX *Design* está relacionado, paralelamente, ao uso do serviço ou produto, no que

Figura 2.10: Planejamento de Aplicativo e Conceito de UX.
Fonte: (ARTY, 2018)

tange sua interação, e em como isso afeta as emoções e experiências do usuário.(ARTY, 2018, 2014), grifos do autor.

2.9 Uberização

Em 2010, o aplicativo Uber foi lançado e, com ele, surgiu a ideia de colocar usuários para trabalhar. Quem estava fora do mercado de trabalho teve uma oportunidade de ganhar dinheiro mesmo em crise. A mesma empresa foi criada no auge da crise econômica de 2008, e se tornou um negócio avaliado em mais de 60 bilhões de dólares.

O termo Uberização, é derivado do nome da empresa citada, a mesma transformou o segmento de transporte que há muito tempo não oferecia nada de inovador para seu consumidor. Pode ser visto como um sinônimo de Economia de Compartilhamento (2.1), ou então, todas as empresas que promovem o uso compartilhado de bens através de um serviço.

Define a criação de um modelo de negócios baseado no produto e em como ele é oferecido através de um serviço. Além disso, o termo estipulou para o jargão da sociedade, o verbo "Uberizar"como uma ação de transformar a maneira de consumir bens e serviços, por meio da criação de um modelo de negócios disruptivo. Tal modelo se comporta de maneira que,

caso a empresa declare falência, seus funcionários ainda mantêm o meio de produção - carros e celulares, por exemplo -, sendo esse um fator primordial para que mais pessoas queiram trabalhar para estas empresas.

A Uberização pode ser aplicada em diversos tipos de serviços, como serviços para casa, serviços de reparos, delivery de objetos ou alimentos, serviços de hospedagem, e qualquer outra variação dos anteriores. No Brasil a expansão desse tipo de serviço começou em 2016 e ainda existe muito espaço para crescimento. Um dos elementos que favoreceu o crescimento da Uberização, no Brasil, foi a crise econômica, a qual levou pessoas a buscar renda extra, assim como atiçou a cobiça dos mais jovens por trabalhar de forma menos formal, não se submetendo às amarras de uma empresa ordinária. (FONSECA, 2017; GALANTE, 2018; RICCIARDI, 2016; UBER TECHNOLOGIES, 2019)

2.10 Trabalhos Relacionados

Os trabalhos relacionados escolhidos como incentivo e referência para o presente trabalho, são produtos e serviços já existentes no mercado. Os mesmos vêm de uma linha de modelo de negócios, os quais se encaixam na dita "Economia de Compartilhamento", fundamentada na seção 2.1. Esta foi fomentada, e intensificada, por inúmeras empresas que funcionam sob plataformas de serviços sob demanda e que utilizam dos dispositivos móveis e de seus recursos para levar tais serviços à seus usuários por meio de aplicativos. Dentre os inúmeros aplicativos fornecedores de serviço sob demanda, os que mais se assemelham ao trabalho aqui proposto são:

2.10.1 AirBnb

O Airbnb[6] é uma empresa que utiliza o recurso de serviço *on-line* sob demanda. O modelo de negócios da empresa consiste em conectar pessoas ditas viajantes e pessoas que desejam alugar, por alguns dias, uma casa, um quarto, ou mesmo um espaço no sofá. Hoje, o aplicativo é uma das maiores plataformas de hospedagem alternativa do mundo, formando uma comunidade que conecta pessoas em 191 países, possibilitando-lhes viajar, hospedar-se e viver, mesmo que por alguns dias, como moradores locais.

No Brasil, o Airbnb conta com mais de 80 mil anúncios nos 27 Estados e em 670 cidades. O país é o maior mercado da empresa na América Latina, tanto em número de anfitriões como de viajantes. (FORBES, 2016)

O aplicativo do AirBnb possui um design limpo e intuitivo. Para o usuário realizar seu login, há várias formas de cadastro pelas redes sociais como pode ser visto na figura 2.11. A tela principal do aplicativo possui navegação pela barra inferior, como pode ser vista na figura 2.12. O aplicativo proposto nesse trabalho também possuirá login via rede social Google para

[6] `https://www.airbnb.com.br`

cadastro do usuário, porém a sua navegação se dará por um menu lateral ao invés de barra inferior. O Airbnb mantém a mesma plataforma para os dois tipos de usuários, seja viajante ou passageiro, assim como o aplicativo aqui proposto fará.

2.10.2 GetNinjas

A GetNinjas[7] é uma plataforma de contratação de serviços no Brasil. Conectam profissionais de todo o país a pessoas solicitando serviço, atendendo com qualidade, facilidade e rapidez todos os tipos de necessidade. Possuem o aplicativo e uma plataforma *web* onde ambos podem realizar o cadastro de usuários e utilizar a ferramenta. (GETNINJAS, 2021)

O aplicativo, antes do cadastro do usuário, fornece a opção de escolha entre cliente e profissional, o que o trabalho proposto aqui não fará. Ver figura 2.13. O cadastro do cliente, é feito apenas pelo número de celular do usuário. O cadastro do profissional exige número de celular e e-mail de contato. Os fluxos principais do aplicativo diferem, quando o usuário é um cliente ou um profissional, sendo assim cada tela e opção de funcionalidade tem seu próprio *design*, e nada é reaproveitado. No trabalho proposto, a ideia é manter um único aplicativo para os dois usuários e reutilizar as telas para fins de implementação mais rápida.

2.10.3 Sem Patrão

Como o aplicativo citado anteriormente, o Sem Patrão[8] também é um *marketplace*[9] de serviços. Diferentemente do GetNinjas, sua plataforma *web*, apenas realiza consulta de serviços e cadastro de clientes. O cadastro de profissionais é realizado apenas pelo aplicativo, iOS - sistema operacional dos dispositivos móveis da Apple[10] - ou Android. Para cadastro é utilizado o número de celular do usuário e posteriormente enviada a uma tela de cadastro do perfil com todas as informações necessárias. Após o cadastro o usuário é enviado à tela principal onde é possível navegar por um histórico de postagens de outros usuários com seus serviços, em formato de rede social, e navegação via barra inferior, assim como no aplicativo do AirBnb. Ver figuras 2.14 & 2.15.

No trabalho aqui proposto, não haverá diferenças entre cadastro de usuários, todos serão feitos por meio de login via Google, sendo assim, o usuário precisa apenas de uma conta cadastrada em seu dispositivo móvel vinculada ao Google. Para realizar um cadastro com número de celular, como neste trabalho relacionado e outros citados acima, seria necessário um *backend* muito mais robusto para realizar as confirmações de números de celulares e envios de mensagens de texto, com um serviço específico para isso, o que não entra no escopo desse projeto.

[7] `https://www.getninjas.com.br/`

[8] `https://web.sempatrao.com.br/`

[9] do inglês, pode-se traduzir como "mercado".

[10] empresa de tecnologia fundada por Steve Jobs `https://www.apple.com/br/`

Figura 2.11: Tela de Cadastro Airbnb.
Fonte: Aplicativo AirBnb

Figura 2.12: Tela Principal de Navegação do Airbnb.
Fonte: Aplicativo Airbnb

Figura 2.13: Tela Inicial de Cadastro Aplicativo GetNinjas.
Fonte: Aplicativo GetNinjas.

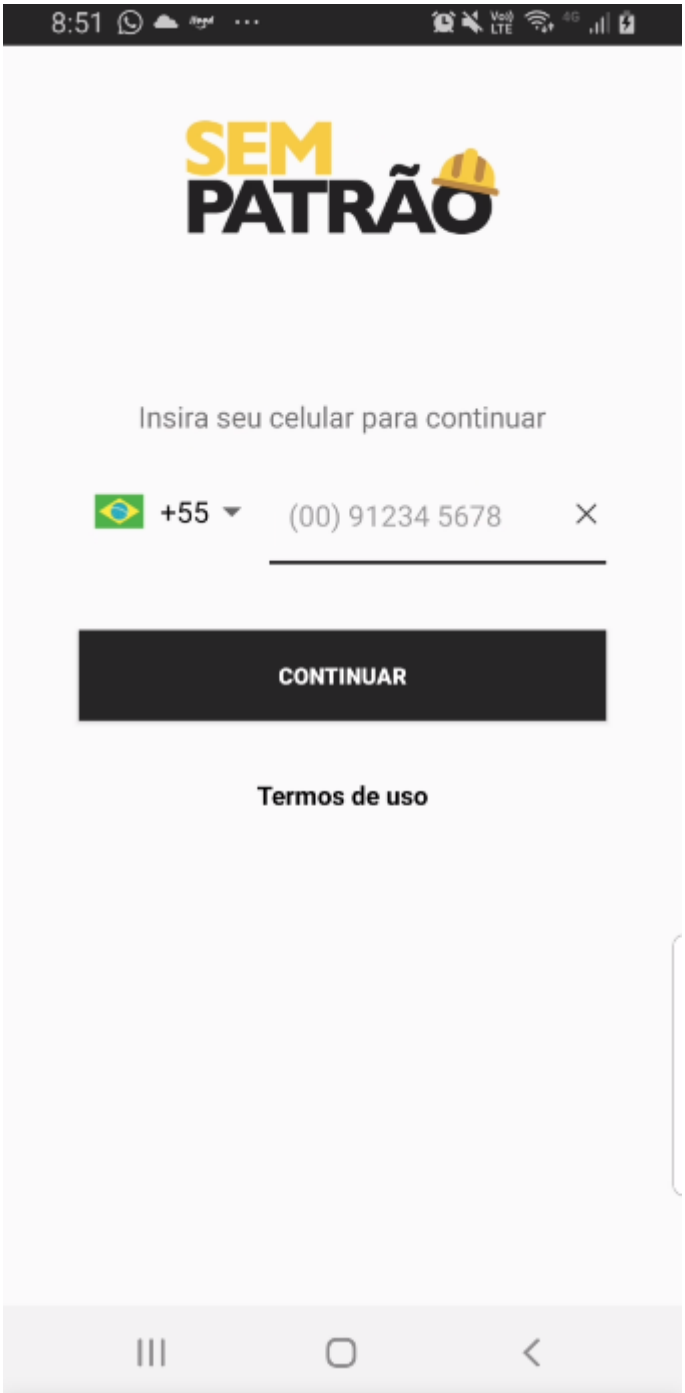

Figura 2.14: Tela Inicial de Cadastro Sem Patrão.
Fonte Aplicativo Sem Patrão.

Figura 2.15: Tela Principal de Navegação Sem Patrão.
Telefone censurado para preservar a privacidade do usuário. Fonte: Aplicativo Sem Patrão.

2.10.4 Uber

O aplicativo da empresa Uber[11] coloca donos de automóveis e pessoas que precisam se deslocar, em contato direto. O proprietário do carro sendo uma pessoa física, passa a se portar como uma empresa e faturar como uma, de forma autônoma. Como explicado na seção 2.9, a "uberização", leva profissionais a buscar renda extra e de forma menos formal. (FORBES, 2016)

A Uber conquistou o cliente pela qualidade de seu serviço. Além de ter carros selecionados e novos, oferece mais comodidade e conectividade ao usuário durante o trajeto. A empresa também garante ao cliente um bom tratamento pelos funcionários, os quais podem ser avaliados, e que o valor cobrado ao final da corrida será justo – ou o usuário terá seu dinheiro de volta, uma questão que foge do comum quando se olha para as garantias de taxistas convencionais. (MENDONÇA, 2015)

Assim como a Uber, a proposta desse trabalho também conecta profissionais a clientes, mas não apenas para fins de deslocamento, para qualquer tipo de serviço cadastrado pelo usuário profissional. Diferentemente da Uber, este trabalho não irá discernir entre aplicativos de cada tipo de usuário, como passageiro e motorista, por exemplo. Vide figuras 2.17 & 2.16. Tais aplicativos possuem um design simples e intuitivo para o usuário. O aplicativo do passageiro permite realizar o login apenas com seu número de celular, posteriormente vinculando à uma conta do Google. O aplicativo do motorista permite realizar o login apenas após o cadastro do usuário com toda a documentação necessária para prestação do serviço.

[11] `https://www.uber.com/`

Figura 2.16: Tela inicial app Uber Passageiro.
Fonte: Aplicativo Uber Passageiro

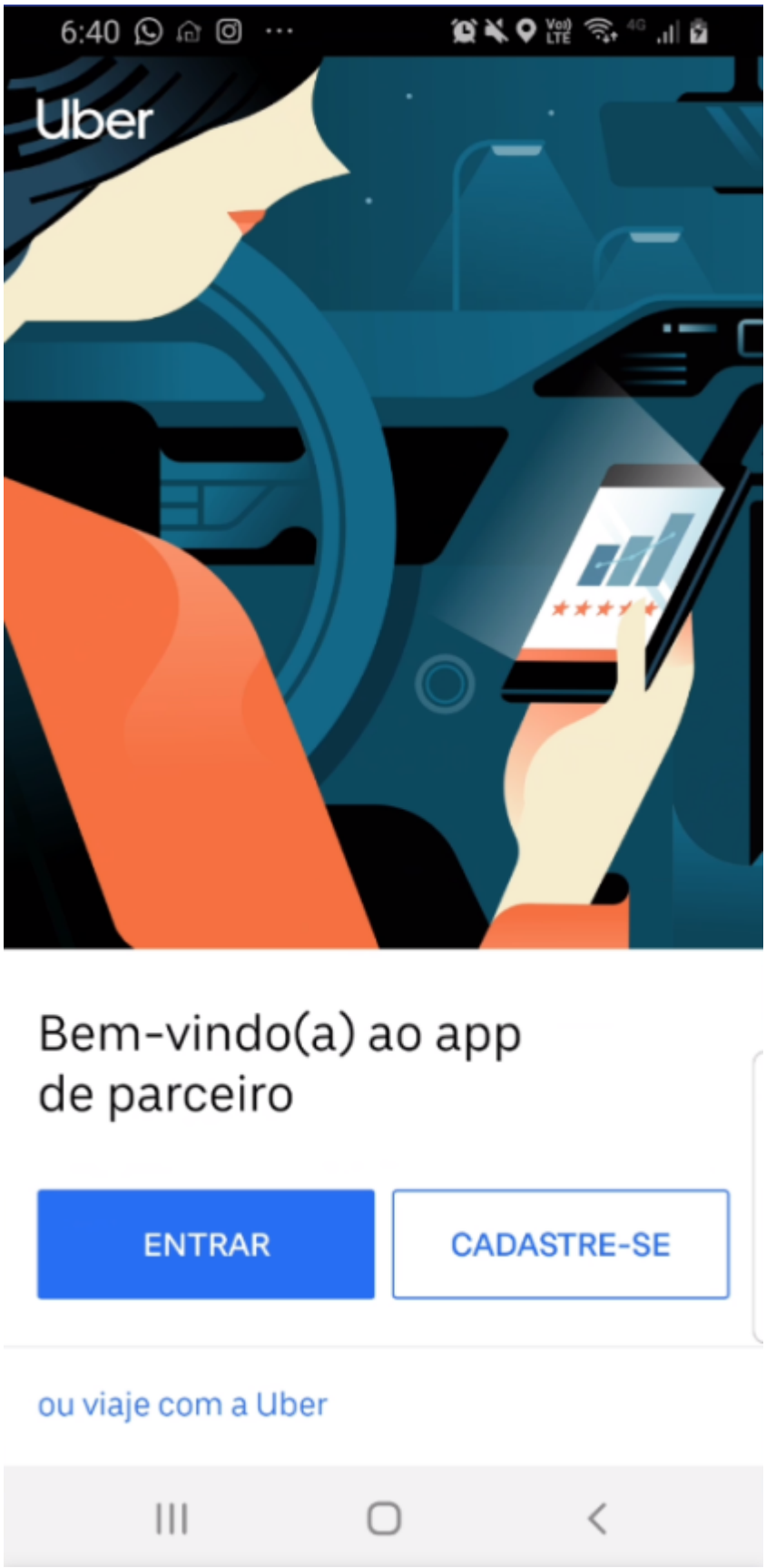

Figura 2.17: Tela inicial app Uber Motorista.
Fonte: Aplicativo Uber Driver

Capítulo 3

Desenvolvimento

Neste capítulo será abordado as etapas do desenvolvimento do projeto como um todo. Estrutura, tecnologias e ferramentas utilizadas para ter os resultados satisfatórios para o funcionamento do aplicativo.

3.1 Protótipo de Baixa Fidelidade

Como início de desenvolvimento propõe-se, a seguir, o protótipo de baixa fidelidade, desenvolvido pelo autor, contendo as seguintes telas:

- Tela de Login & Tela Principal (Figura 3.1);
- Procurar Serviço (Figura 3.2);
- Avaliar Serviço (Figura 3.3);
- Ofertar Serviço (Figura 3.4);
- Menu Lateral (Figura 3.5);

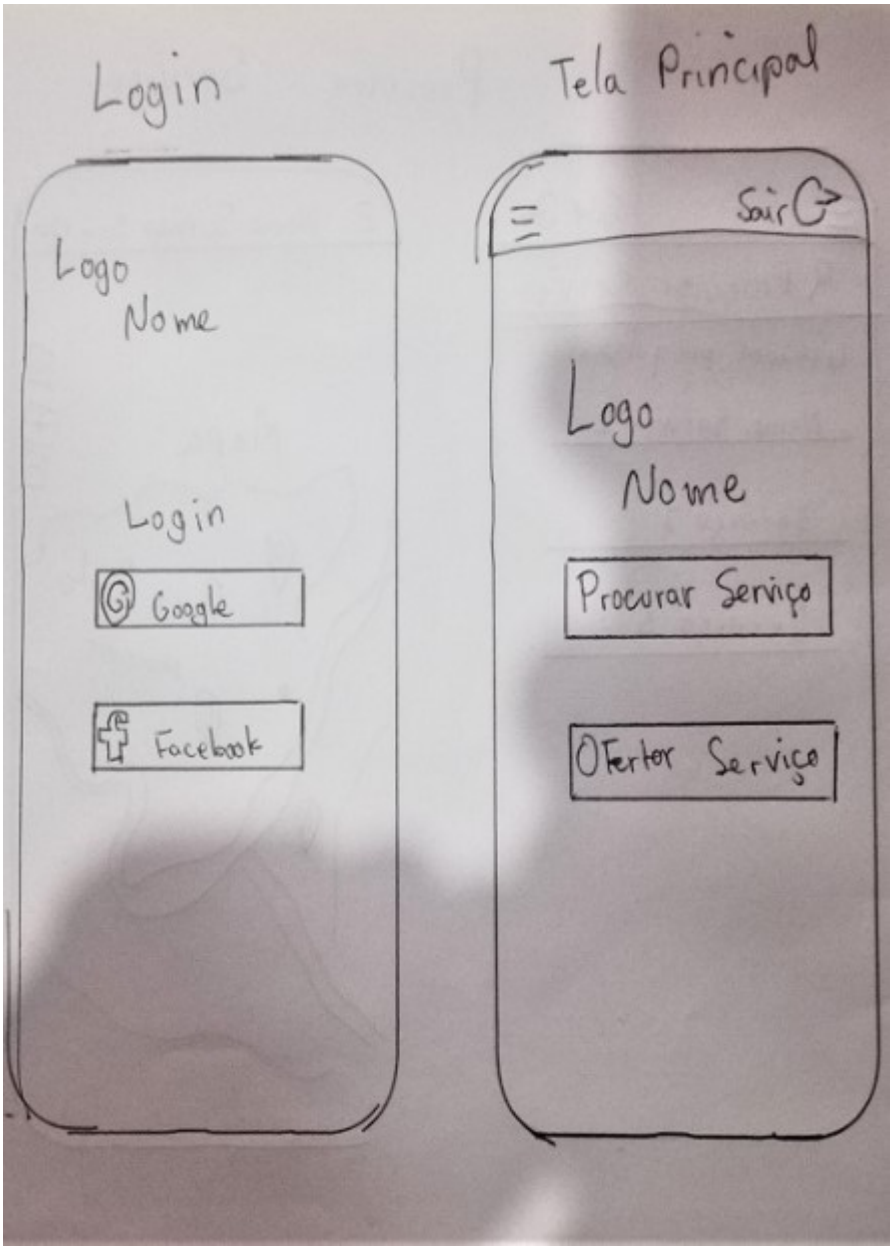

Figura 3.1: Protótipo Tela de Login & Tela Principal.
Fonte: Elaborado pelo autor

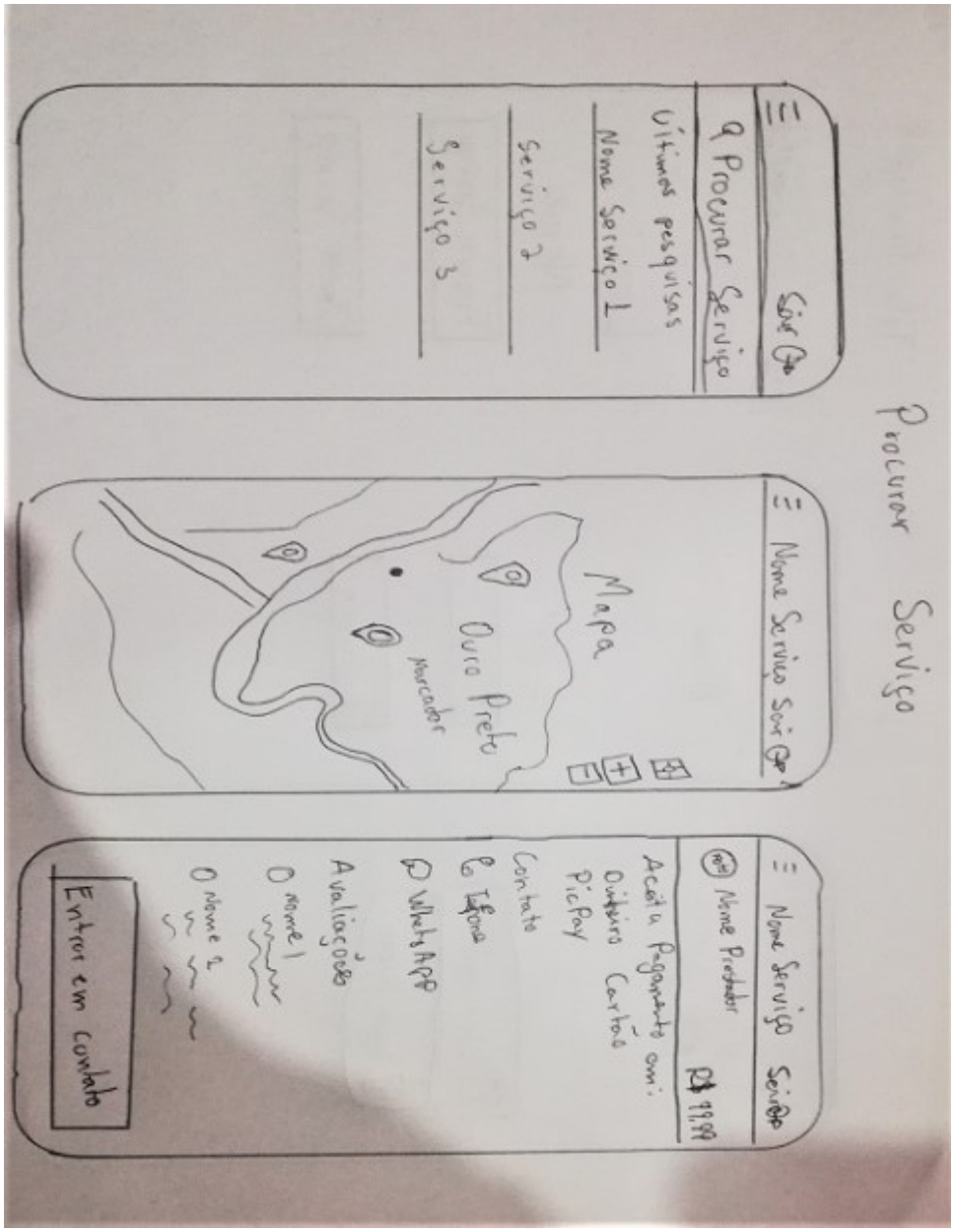

Figura 3.2: Protótipo Tela de Procurar Serviço.
Fonte: Elaborado pelo autor

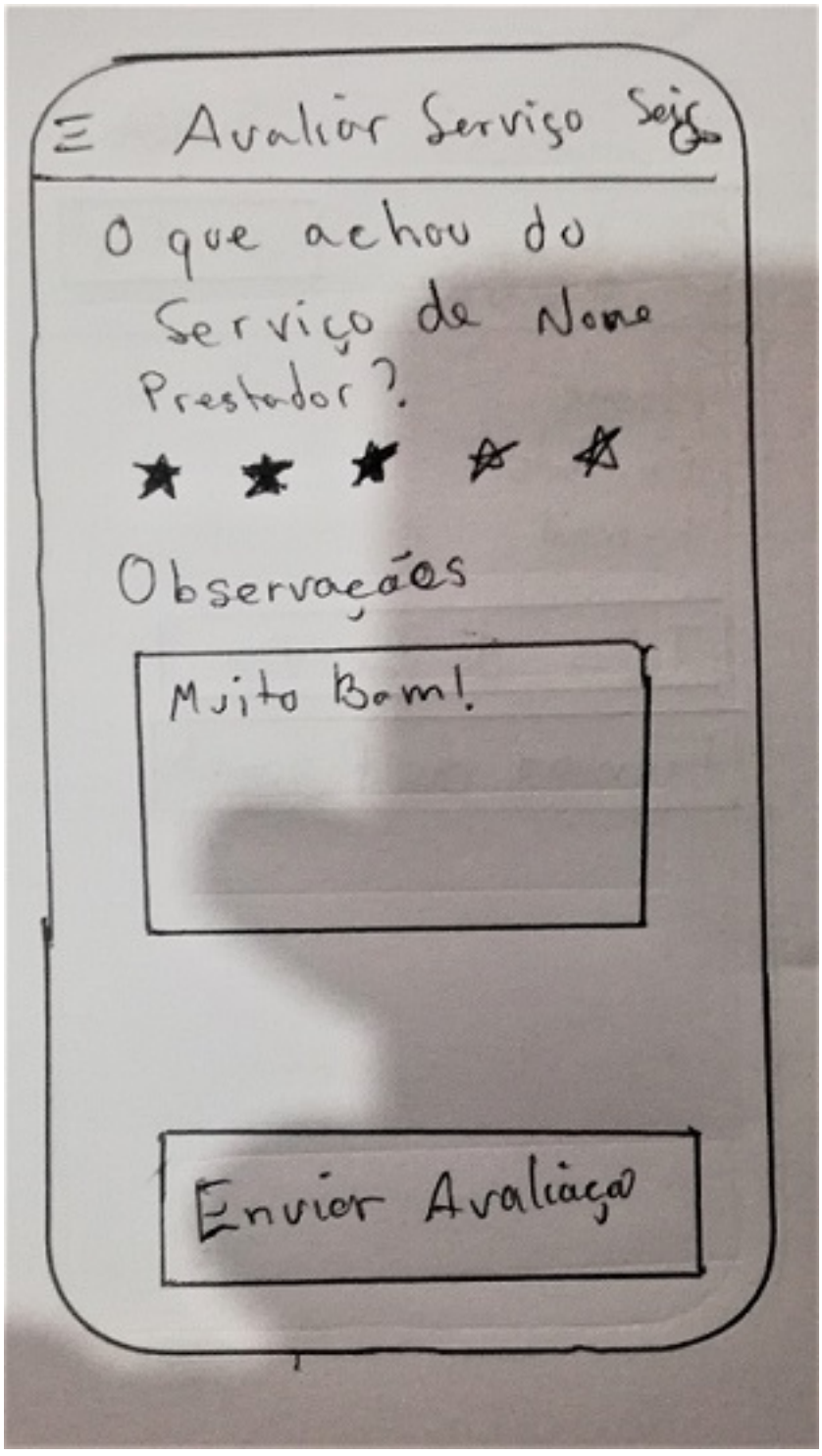

Figura 3.3: Protótipo Tela de Avaliar Serviço.
Fonte: Elaborado pelo autor

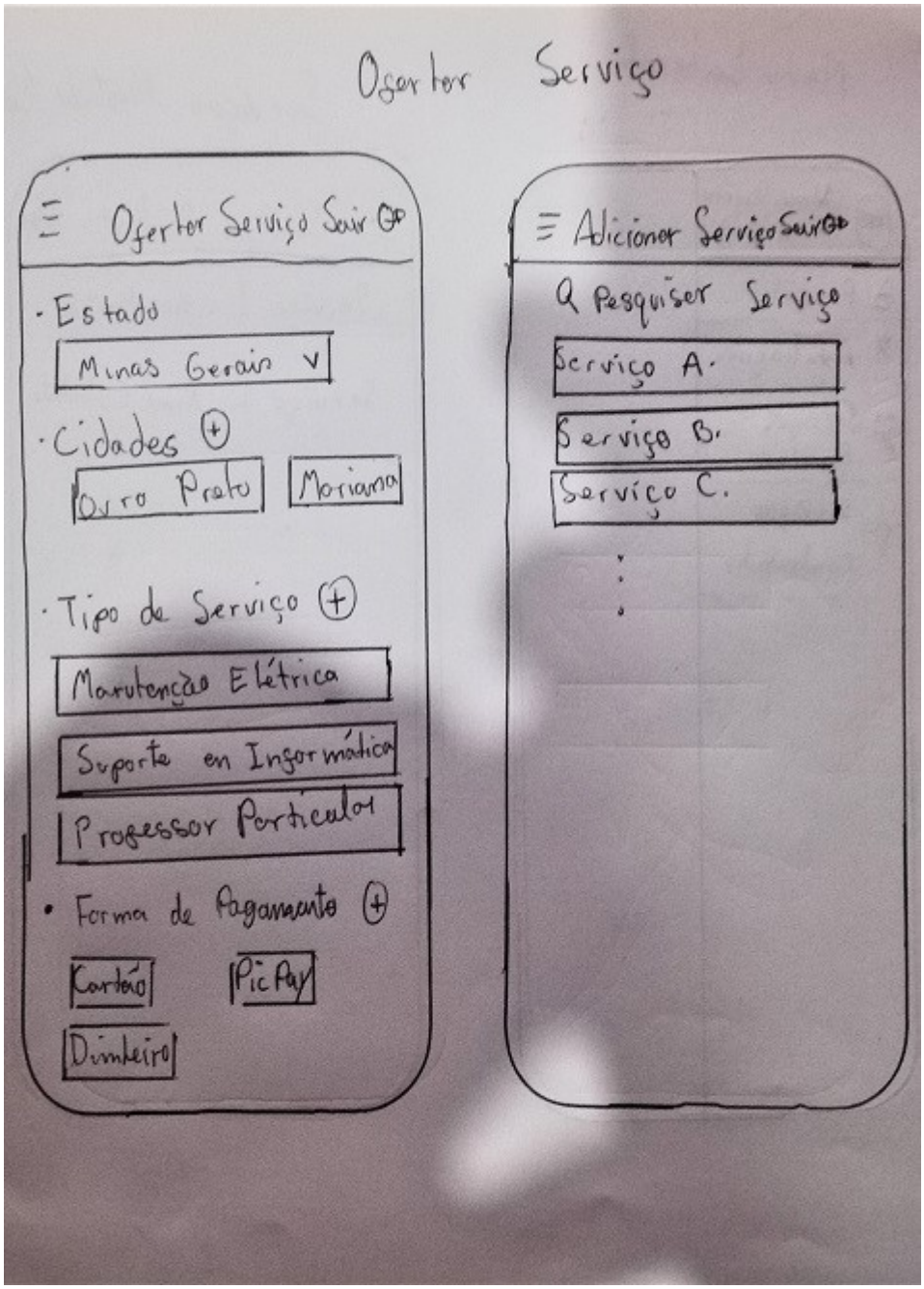

Figura 3.4: Protótipo Tela de Ofertar Serviço.
Fonte: Elaborado pelo autor

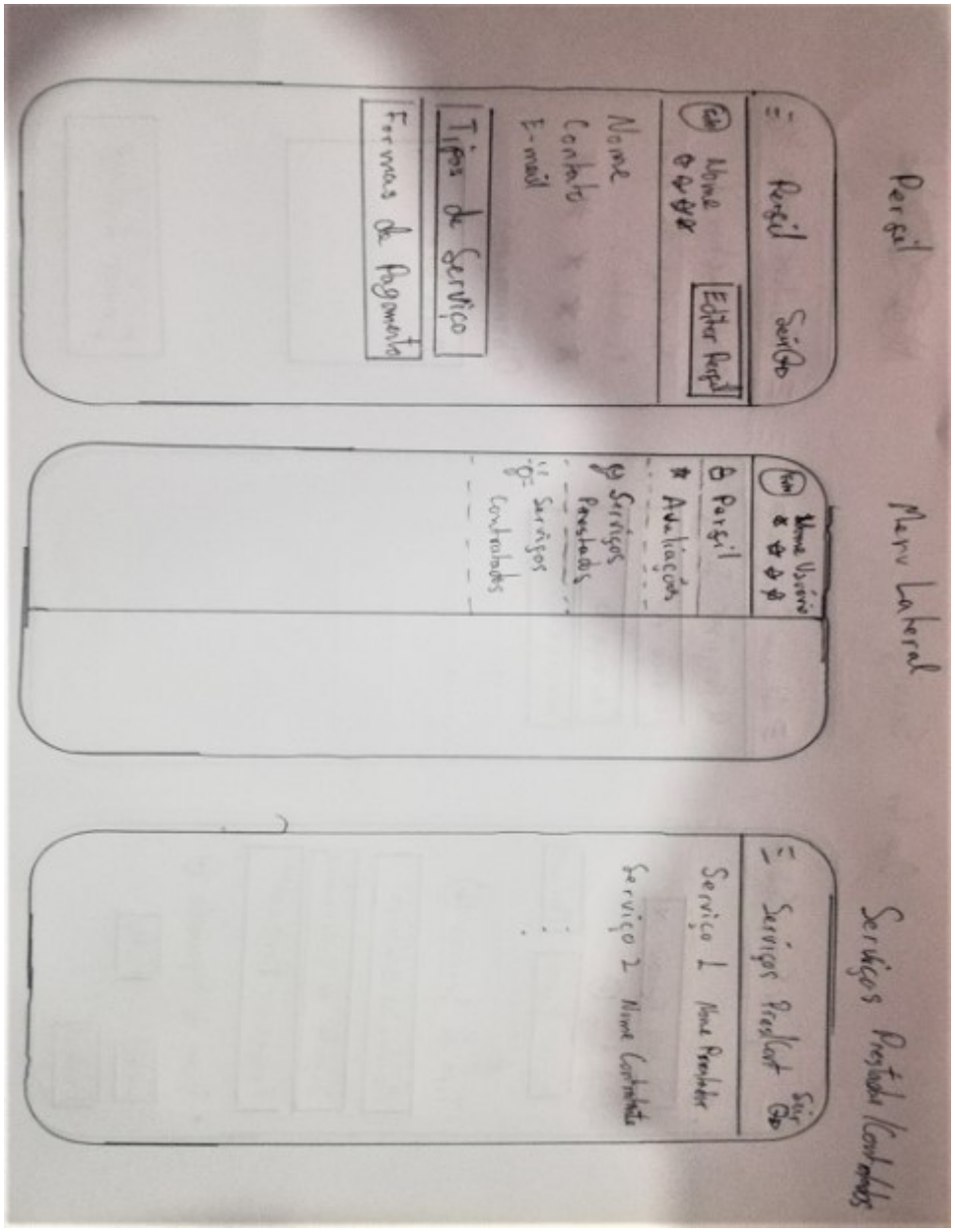

Figura 3.5: Protótipo Tela do Menu Lateral.
Fonte: Elaborado pelo autor

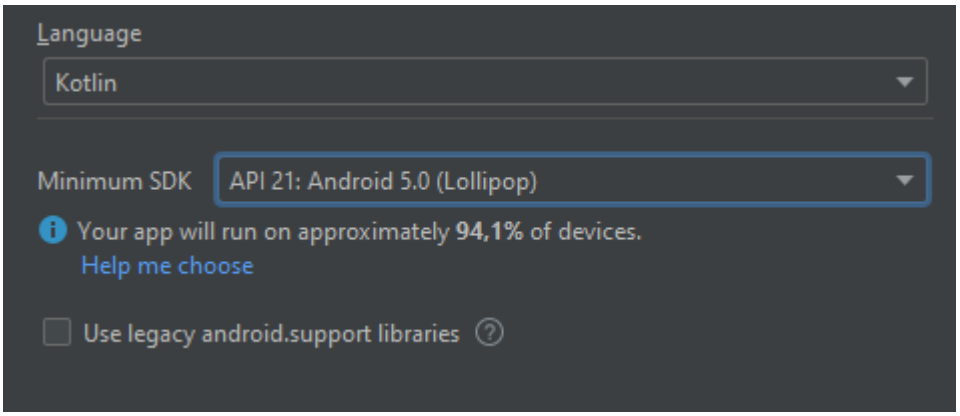

Figura 3.6: Seleção da API 21 Android 5.0.
Fonte: Elaborado pelo autor

3.2 Criação do Projeto

Como a intenção do trabalho é poder estar acessível para a maior parte de estudantes possíveis, o aplicativo foi desenvolvido com API[1] mínima 21, correspondente ao Android 5.0 *Lollipop*. Dessa forma o aplicativo será executável em dispositivos com Android 5.0 ou superior. Esta API foi lançada em Novembro de 2014 (GOOGLE DEVELOPERS, 2020) e suporta até 94.1% de dipositivos atualmente (Figura: 3.6). Versões de API inferiores à 21 estão muito obsoletas, possuem implementações de bibliotecas e funções nativas que já estão descontinuadas, sendo assim se torna inviável a manutenção e manipulação do código.

3.2.1 Ambiente de Desenvolvimento - Android Studio

A *IDE*[2] utilizada para desenvolvimento do projeto, foi o Android Studio[3], ferramenta de desenvolvimento oficial de apps Android, baseado no *IntelliJ IDEA*[4], da *Jetbrains*. Junto com o editor de código, a ferramenta inclui plugins avançados do IntelliJ, para facilitar o desenvolvimento e aumentar a produtividade na criação dos aplicativos, tais como:

- Um sistema de compilação flexível baseado no Gradle;
- Emulador de dispositivos móveis, podendo escolher qual API do sistema operacional Android rodar, o tamanho de tela, além de outras configurações de hardware do dispositivo, podendo testar a compatibilidade do aplicativo em desenvolvimento com os inúmeros dispositivos disponíveis à venda no mercado;

[1] do inglês, Application Programming Interface - Interface de Programação de Aplicação é um conjunto de rotinas e padrões estabelecidos por um software para a utilização das suas funcionalidades por aplicativos que não pretendem envolver-se em detalhes da implementação do software, mas apenas usar seus serviços. (HOWE, 2021)

[2] do inglês, *Integrated Development Environment* ou Ambiente de Desenvolvimento Integrado

[3] `https://developer.android.com/studio`

[4] `https://www.jetbrains.com/idea/`

- Integração com sistemas de controle de versão, como Github, Gitlab, Bitbucket, entre outros;

  (GOOGLE DEVELOPERS, 2021d)

3.2.2 Controle de Versão

Será utilizada a plataforma *GitHub* para a criação, hospedagem e controle de versionamento do projeto. [5]

3.3 Implementação

3.3.1 Padrão de Projeto

Em todo projeto de software é importante definir qual o padrão de projeto que será utilizado. Padrões de projeto são soluções típicas para problemas comuns em projeto de software. O padrão não é um pedaço de código específico, mas um conceito geral para resolver um problema em particular. Permitem customizar classes para resolver um problema de projeto recorrente em seu código. (GURU, 2021)

Os padrões de projeto podem ser divididos em dois critérios. O primeiro é a finalidade, reflete o que um padrão faz. As finalidades podem ser criacionais, comportamentais e estruturais. Padrões de criação descrevem técnicas para instanciar objetos ou classes, e possibilitam organizá-los em estruturas maiores. Padrões de comportamento caracterizam-se pelo modo os quais classes e/ou objetos interagem e distribuem responsabilidades entre si. Padrões estruturais tratam a composição de classes e/ou objetos. O segundo critério é o escopo do padrão - especifica se o padrão é aplicado à classe ou ao objeto. (LEITE, 2005)

3.3.2 MVVM - *Model View ViewModel*

O padrão de *software* arquitetural utilizado no projeto foi o MVVM - *Model View ViewModel*. Padrões de arquitetura, são tipos de padrões de projeto, que podem ser classificados como uma solução genérica e reutilizável para resolver algum problema da arquitetura do projeto, dado seu contexto específico. Aplicações Android realizam várias chamadas de serviços simultaneamente de maneira assíncronas, como serviços de autenticação, banco de dados, *download*, *upload* entre outros tipos de serviço durante seu ciclo de vida. O objetivo dessa arquitetura é de separar essas responsabilidades em classes diferentes, abstraindo cada camada do aplicativo.

- *View*: Entidades responsáveis por definirem a estrutura, *design* e aparência do que será exibido na tela. No Android, as *Views* podem ser *Activities, Fragments* e elementos visu-

[5] `https://github.com/gabrielbrandaodelima/studentaid`

ais criados para serem disponibilizados na tela. Também são responsáveis pela interação com o usuário.

- *Model*: Estruturas do modelo de domínio da aplicação, composto por modelo de dados, regras de negócio e validações de lógica.

- *ViewModel*: Estruturas livres de contexto, que agem como intermediário entre a *View* e o *Model*, é o responsável por manusear o *Model* para ser utilizado pela View, notificar mudanças aos observadores e escutadores de dados de servidor, por exemplo, para atualizar a camada de interface do usuário (*View*).

(SOUTO, 2018)

3.3.3 Arquitetura Primária

A estrutura principal do projeto, composta pela Atividade de navegação principal do aplicativo, foi criada utilizando a arquitetura de *SingleActivity*, paralelamente ao padrão MVVM. A ideia de tal arquitetura é poder ter apenas uma atividade que controla vários fragmentos da aplicação, e assim, permite compartilhar todos os tipos de informações de dados entre eles, como referência ao usuário logado e os outros usuários do banco de dados. Nesse caso a atividade funciona como um contêiner de telas e os fragmentos são as telas a serem manipuladas.(KITPITAK, 2020)

Apesar do aplicativo possuir em seu fluxo principal de navegação a estrutura de *SingleActivity*, foi necessário criar uma atividade responsável por tratar apenas do login do usuário. A tratativa de implementar o fluxo de login dentro do fluxo principal do aplicativo, estava gerando alguns problemas de comportamento e por isso foi necessário o particionamento dessas camadas em duas atividades independentes.

Como pode-se observar na figura 3.7, o fluxo de login é composto apenas por duas classes, responsáveis por autenticar e salvar o usuário no banco de dados, a atividade *LoginActivity* e o fragmento *LoginFragment*, e o fluxo principal da aplicação é composta pelas classes: *MainActivity* como atividade principal de controle das telas, e os fragmentos *MainFragment* contém a exibição do mapa e suas interações, *ProfileFragment* para manipulação do perfil do usuário e de outros usuários, *ProvidedServicesFragment* para exibição e gerenciamento dos serviços prestados pelo usuário, *ContractedServicesFragment* para exibição e gerenciamento dos serviços contratados pelo usuário.

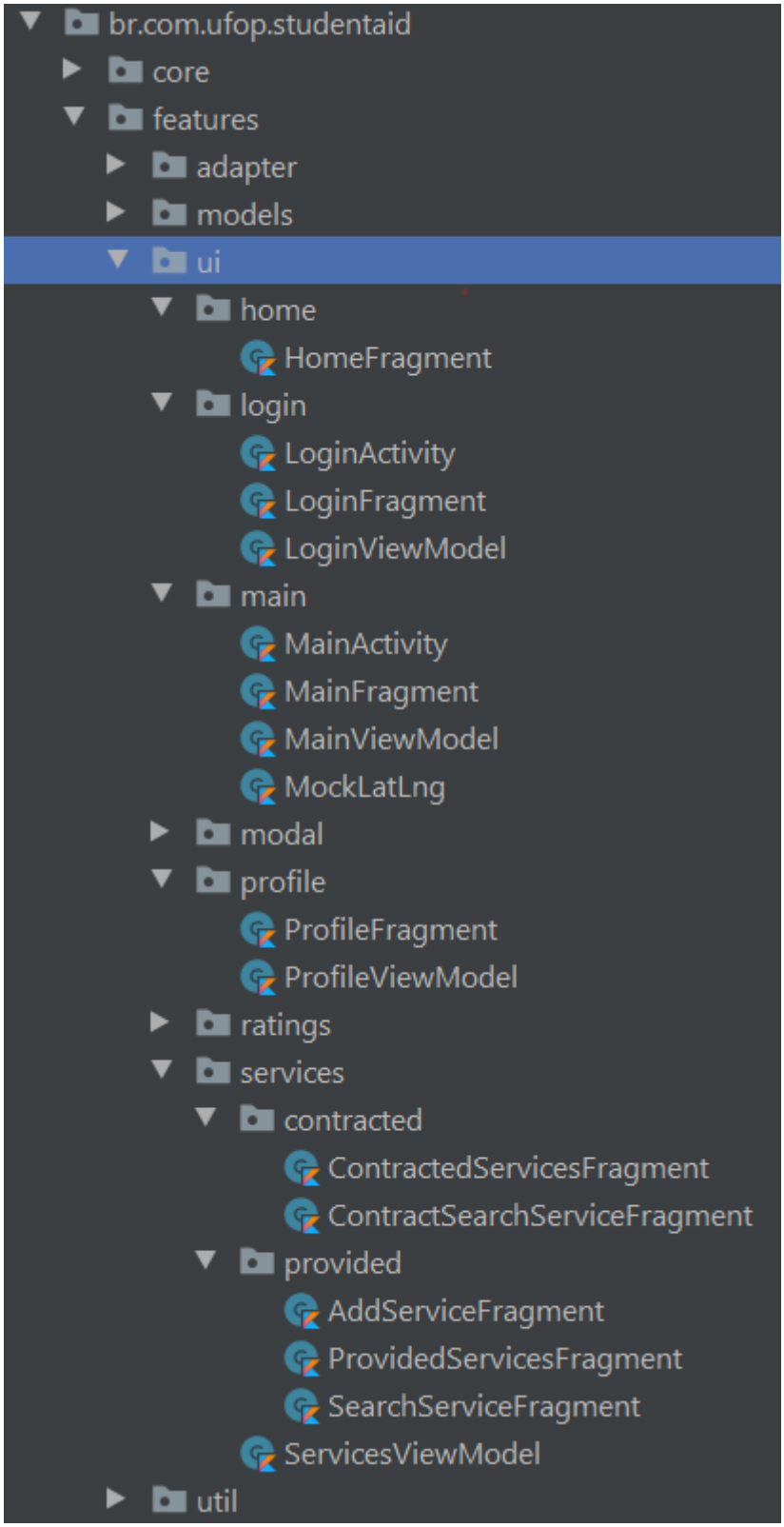

Figura 3.7: Estrutura principal de classes do projeto.
Fonte: Elaborado pelo autor no Android Studio

3.3.4 Gráfico de Navegação do Fluxo Principal

Ao finalizar o projeto seu gráfico de navegação seguiu o modelo da figura 3.8 :

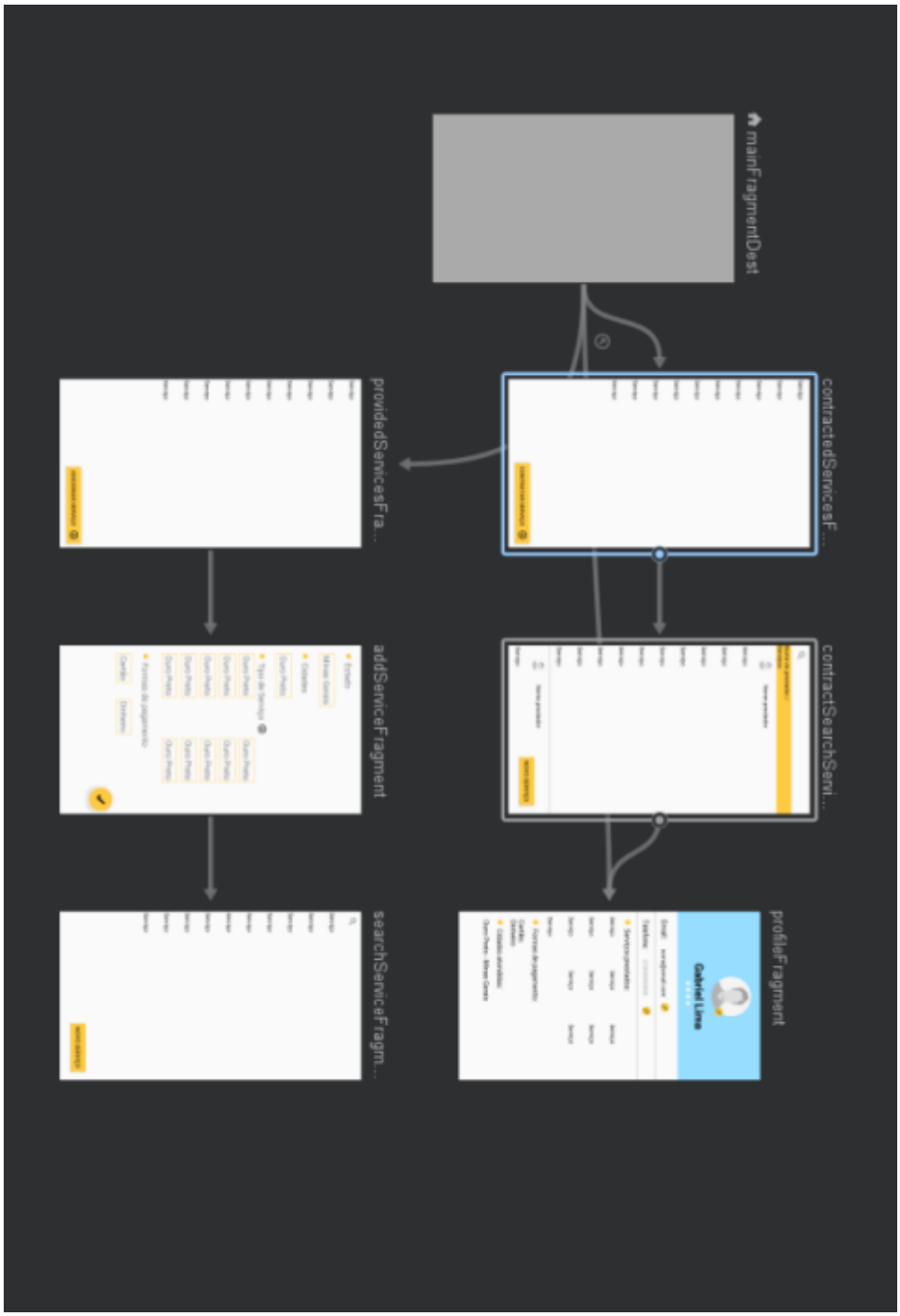

Figura 3.8: Estrutura do gráfico de navegação do fluxo principal do aplicativo.
Fonte: Elaborado pelo autor no Android Studio

3.4 Banco de Dados

O banco de dados utilizado para integrar ao aplicativo foi o *Google Firebase*[6]. O *Firebase* fornece várias ferramentas para desenvolver aplicativos de alta qualidade e ampliar sua base de usuários, podendo posteriormente realizar avaliações e análises métricas. Como revisado na seção 2.6.

3.4.1 *Firebase Authentication*

A maioria dos apps precisa reconhecer a identidade do usuário. Ter essa informação permite que um app salve os dados do usuário na nuvem com segurança e forneça a mesma experiência personalizada em todos os dispositivos do usuário. Ele oferece suporte à autenticação usando senhas, números de telefone, provedores de identidade federados conhecidos, como Google, Facebook, Twitter, entre outros. (GOOGLE DEVELOPERS, 2021c)

3.4.2 *Firebase Cloud Firestore*

O *Firebase Cloud Firestore*[7] é um banco de dados *NoSQL*[8] hospedado em nuvem da Google para armazenar e sincronizar dados para o desenvolvimento do cliente e do servidor. Bancos de dados não relacionais são criados para modelos de dados específicos e têm esquemas flexíveis para a criação de aplicativos modernos. Se encaixam em uma classe definida de banco de dados que fornecem um mecanismo para armazenamento e recuperação de dados que são modelados de formas diferentes das relações tabulares usadas nos bancos de dados relacionais. (AMAZON, 2021)

Assim como o *Firebase Realtime Database*, mantém os dados em sincronia nos aplicativos cliente por meio de *listeners*[9] em tempo real. Além disso, oferece suporte *off-line* para dispositivos móveis e *web* a fim de que se possa criar aplicativos responsivos que funcionem independentemente da latência da rede ou da conectividade com a internet. O *Cloud Firestore* também oferece integração perfeita com outros produtos do *Firebase* e do *Google Cloud*.

Com o modelo de dados *NoSQL* do *Cloud Firestore*, é possível armazenar todos e qualquer tipo de dados em documentos que contêm mapeamentos de campos para valores. Tais documentos são armazenados em coleções, que se tornam contêineres dos documentos, os quais pode-se usar para organizar e consultar dados. As consultas são expressivas, eficientes e flexíveis. É possível realizar consultas superficiais para recuperar dados no nível do documento sem precisar recuperar a coleção inteira ou qualquer subcoleção aninhada. As estruturas de dados podem ser armazenadas como muitos tipos de dados diferentes, seja cadeia de caracte-

[6] `https://firebase.google.com/`

[7] `https://firebase.google.com/docs/firestore`

[8] *NoSQL* é um termo genérico que representa os bancos de dados não relacionais.

[9] traduz-se "escutadores", que são estruturas de dados as quais recebem atualizações em tempo real de dados manipulados em servidores

res, números simples ou objetos complexos. Tudo isso faz com que a integração desse tipo de banco de dados com um aplicativo seja simples e rápida. (GOOGLE DEVELOPERS, 2021b)

No projeto, o *Firestore* é responsável por armazenar os dados de todos os usuários como: dados de login (foto, e-mail, nome, telefone), dados de localização (latitude e longitude), dados de serviços prestados e contratados, além do ID único (uid) de cada usuário para localização do mesmo no banco de dados. Devido ao escopo do aplicativo, o *Cloud Firestore* foi escolhido por ter consultas mais avançadas e rápidas, melhor escalabilidade e facilidade de integração e implementação quando comparado ao *Realtime Database*, banco previamente citado na seção 2.6 e um pouco mais robusto. (GOOGLE DEVELOPERS, 2021a) Na figura 3.9 observa-se o painel do *Cloud Firestore* e a tabela de usuários salvos com seus respectivos identificadores únicos e atributos de usuário.

3.5 Login - Autenticação

A autenticação é responsável por controlar quem acessará o aplicativo, recuperar as informações desse usuário e salvar no banco de dados do *Firebase Authentication* 3.4.1. No caso do projeto proposto, foi utilizado apenas a autenticação via *Google Sign-in*, dessa forma, só poderá se cadastrar e utilizar as funcionalidades do aplicativo quem possuir alguma conta no Google, utilizando seu e-mail e senha do gmail salvos no dispositivo móvel. Vide figura 3.10.

3.6 Fluxo Principal do Aplicativo

Após feito o login no aplicativo, o usuário é redirecionado para o seu fluxo principal, composto pelas telas a serem detalhadas em imagens e funcionalidades no capítulo 4:

- Mapa com a localização de todos os usuários cadastrados, representados por um ícone de marcador no mapa;
- Menu lateral com prévia de perfil do usuário e opções de navegação;
  - Perfil
    - Visualizar/Editar Perfis
    - Realizar chamada
  - Serviços Prestados
    - Visualizar Serviços Prestados
    - Adicionar Serviço Prestado
  - Serviços Contratados
    - Visualizar Serviços Contratados
    - Listar / Contratar Prestadores e Serviços

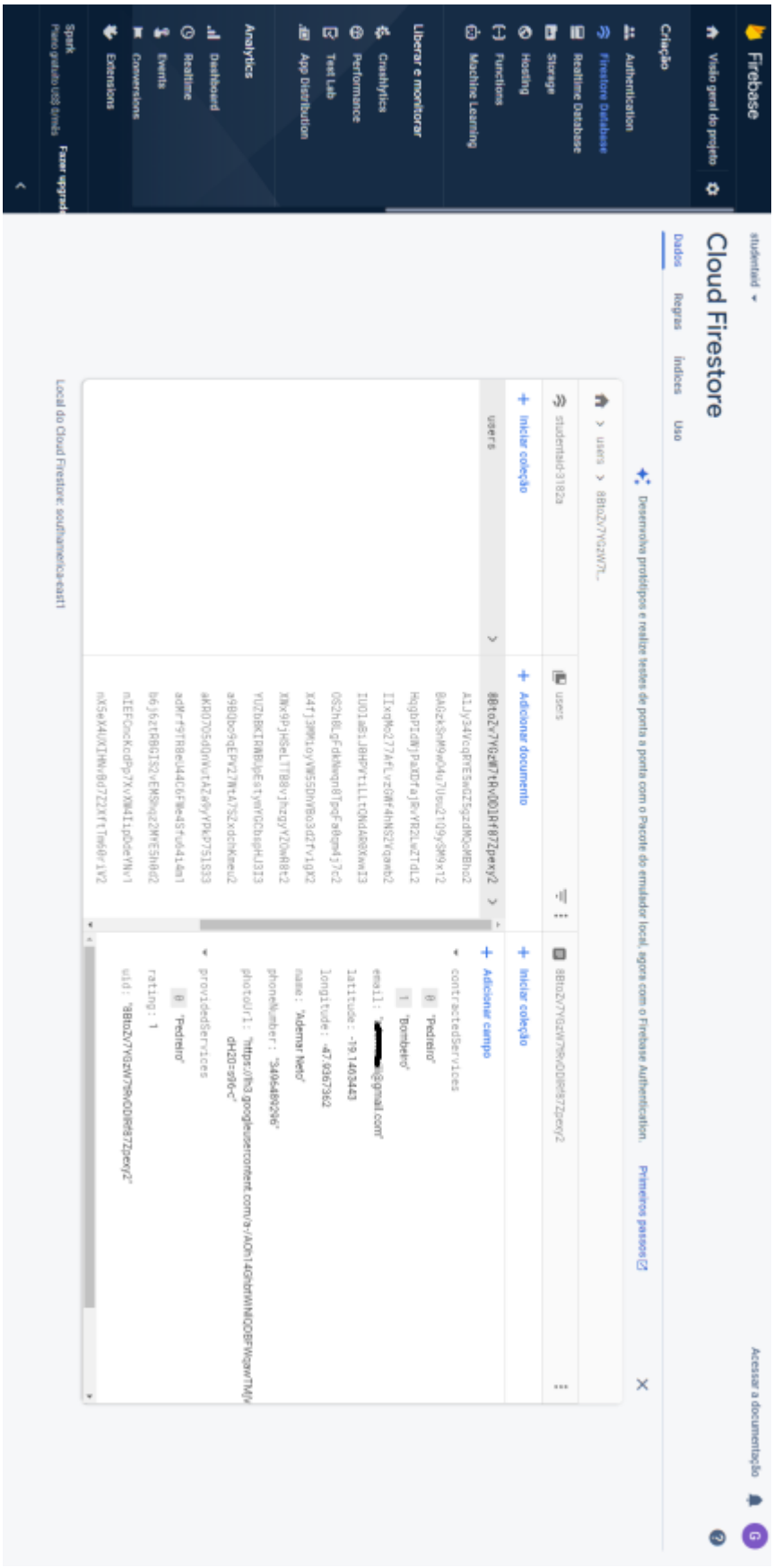

Figura 3.9: Painel principal do *Cloud Firestore* e tabela de ***users***.
Informações censuradas para preservar a privacidade do usuário. Fonte: Elaborado pelo autor

Figura 3.10: Painel principal do *Firebase Authentication* e lista de e-mails de usuários logados pelo *Google Sign-in*.

Fonte: Elaborado pelo autor

Capítulo 4

Resultados

Neste capítulo é feita a apresentação dos resultados obtidos pelo desenvolvimento do aplicativo. São apresentadas as telas, suas tecnologias e funcionalidades implementadas no decorrer do trabalho e é feita a sua análise. A metodologia utilizada foi a seguinte: após implementação do aplicativo como um todo, o mesmo foi enviado para vários usuários reais testarem o aplicativo e popularem seu banco de dados, bem como realizarem suas críticas e/ou sugestões para melhoria na usabilidade e *design* de cada tela. Os resultados são apresentados em imagens e descrições, de modo a proporcionar uma tal conjunção, que permita uma maior facilidade de leitura e análise.

4.1 Login do Usuário

O usuário deve realizar o login, apenas clicando no botão "Fazer login". Em seguida, deve-se selecionar um e-mail de uma conta Google salva em seu dispositivo. Ver figuras 4.1 & 4.2.

4.2 Fluxo Principal do Aplicativo

As subseções a seguir descrevem as funcionalidades em que o fluxo principal do aplicativo é composto.

4.2.1 Solicitar Permissão de Acesso à Localização

Após realizar o login no dispositivo, caso o usuário ainda não tenha permitido o acesso à sua localização via GPS interno do seu dispositivo, será solicitada a permissão para tal, como podemos observar na figura 4.3

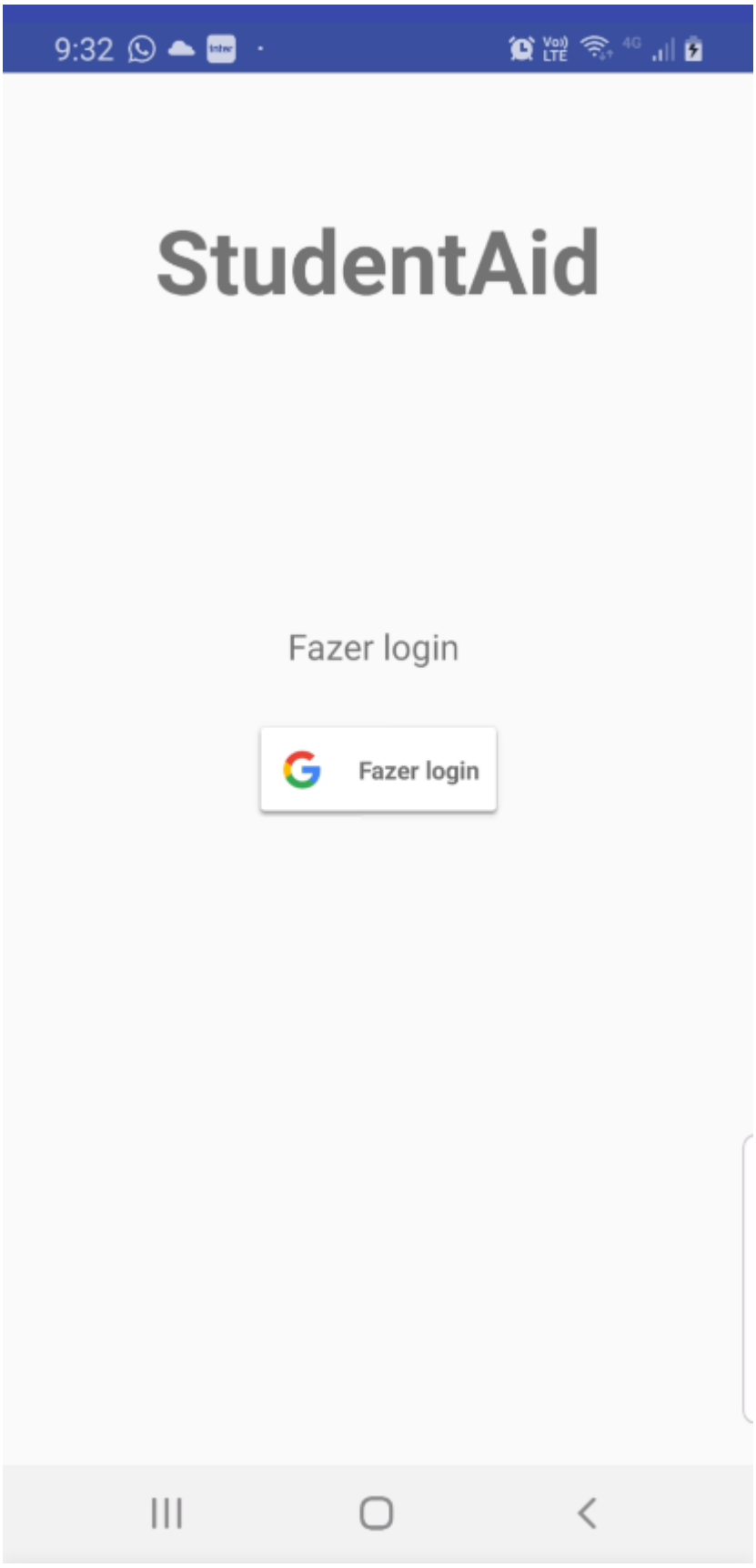

Figura 4.1: Tela de Login do aplicativo com botão para "Fazer Login"
Fonte: Desenvolvido pelo autor

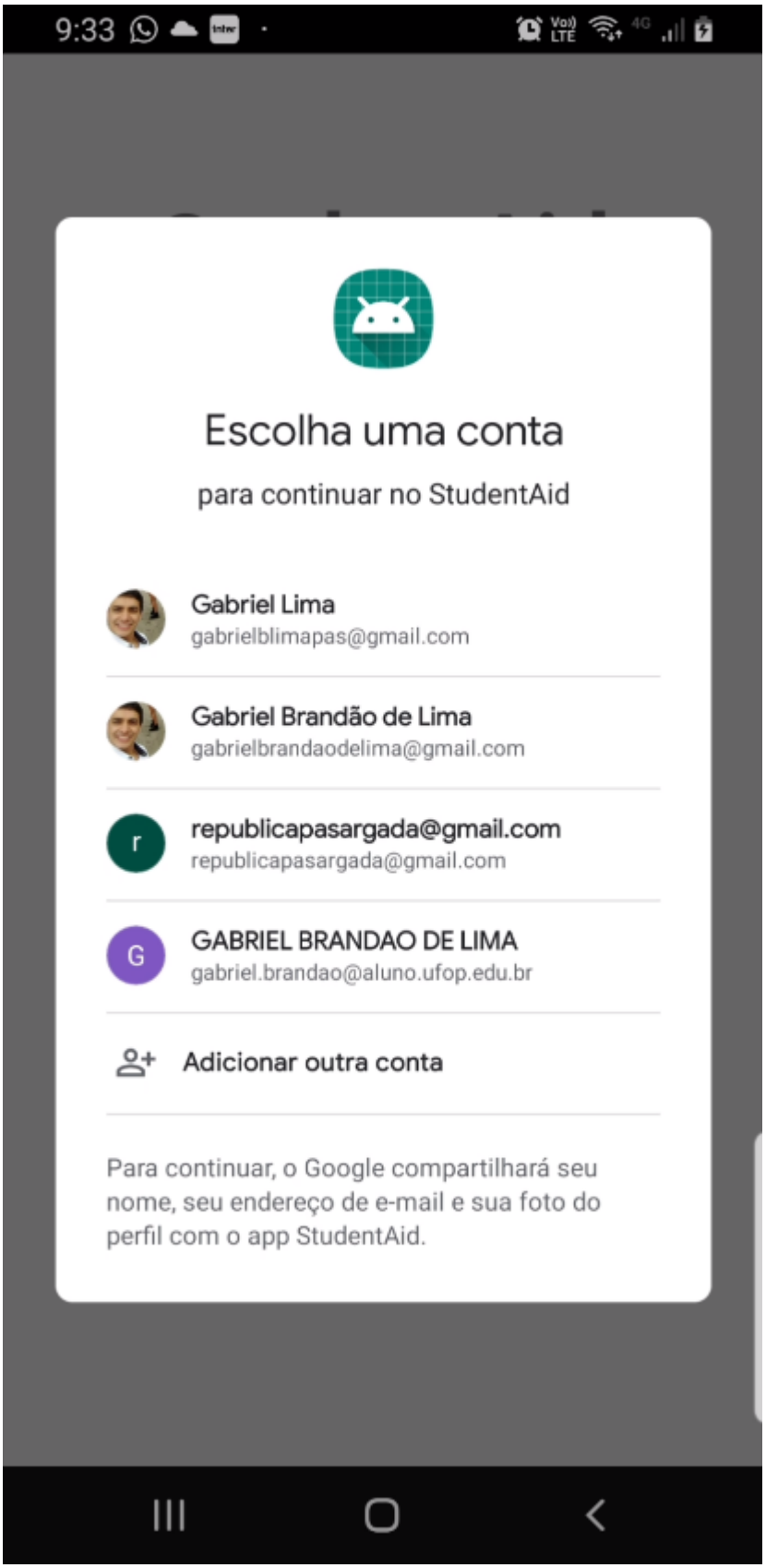

Figura 4.2: Modal para selecionar a conta a ser usada via *Google Sign-in*
Fonte: Desenvolvido pelo autor

Figura 4.3: Solicitação ao usuário de permissão ao acesso da sua localização.
Fonte: Desenvolvido pelo autor

4.2.2 Mapa de Exibição dos Usuários

Ao realizar o login no aplicativo e, se e somente se, a permissão de acesso à localização do dispositivo for concedida pelo usuário, observa-se a revelação do mapa em zoom mínimo, exibindo sua aparência e a localização de todos os usuários cadastrados no aplicativo. Vide figura 4.4 & figura 4.5.

Ao clicar encima de um marcador, como mostrado na figura 4.6 é possível ver o nome do usuário que está naquela localização. Ao clicar no nome do usuário, é inflada a tela de perfil daquele usuário, com suas respectivas informações, veja subseção 4.2.4.

Caso o marcador seja da posição do usuário que está logado, observa-se o texto "Você"exibido no lugar do nome do usuário, como é possível ver na figura 4.7. Percebe-se, nesta figura, uma falta de acurácia entre a latitude e longitude recuperada pelo GPS e a posição real do usuário, representada pelo círculo azul. A posição do usuário, gravada no banco de dados, é recuperada logo após o login e não é atualizada em tempo real, podendo ocasionar tal imprecisão.

4.2.3 Menu Lateral

Na figura 4.8 está representado o menu lateral do aplicativo e suas respectivas opções de navegação. Composto por um cabeçalho, onde é apresentada as informações básicas do usuário logado no aplicativo, como seu nome, e-mail e foto de perfil. Também é apresentado, logo abaixo do e-mail, 5 estrelas as quais representariam a avaliação atual do usuário pelos outros clientes, porém tal funcionalidade de avaliar serviço, não foi possível ser desenvolvida devido à falta de um *back-end* ideal para criar as relações entre usuários e cálculos de avaliações. Além disso, essa funcionalidade não feriu o escopo do projeto. Tais estrelas foram feitas ao implementar a tela tendo como base apenas o protótipo de baixo nível elaborado pelo autor na seção 3.1 do capítulo 3.

4.2.4 Perfil

A tela de perfil contém as informações do usuário/prestador salvas no banco de dados. Ver figura. É possível editar algumas informações como e-mail, foto de perfil e telefone celular. Não é possível alterar o nome de cadastro da conta. Nesta mesma tela são exibidas as cidades atendidas por eles, formas de pagamento aceitas e os serviços prestados por ele, seja usuário logado ou não. Ver figuras 4.9, 4.10 & 4.11.

No perfil do prestador de serviço, não é possível editar nenhuma informação. Apenas visualizar suas informações, e realizar uma ligação para seu número de celular, ao clicar no ícone de telefone, figura 4.12.

Figura 4.4: Exibição do mapa em zoom mínimo.
Fonte: Desenvolvido pelo autor

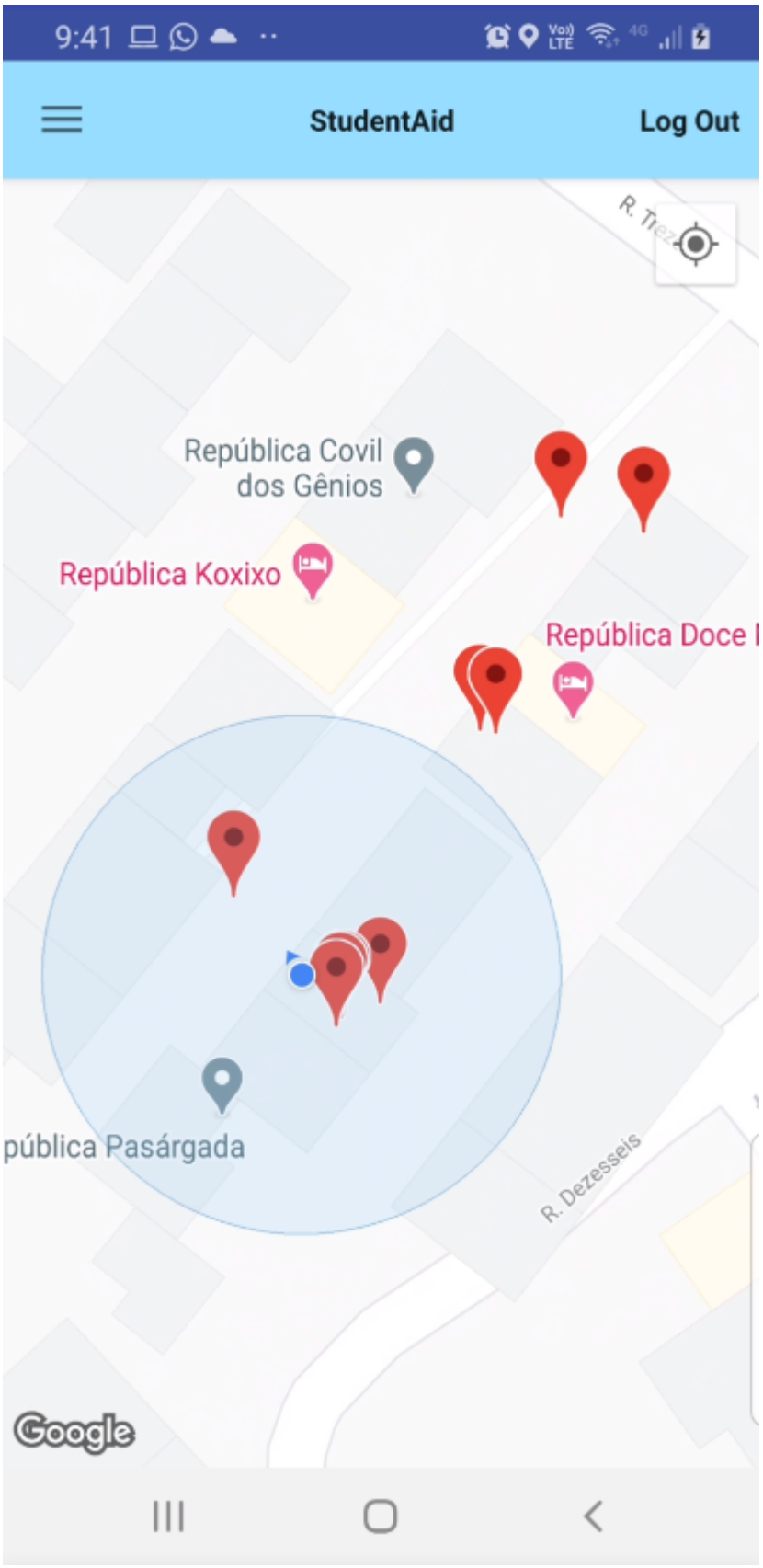

Figura 4.5: Exibição do mapa em zoom máximo.
Fonte: Desenvolvido pelo autor

Figura 4.6: Marcador de um usuário real do aplicativo com seu respectivo nome e localização.
Fonte: Desenvolvido pelo autor.

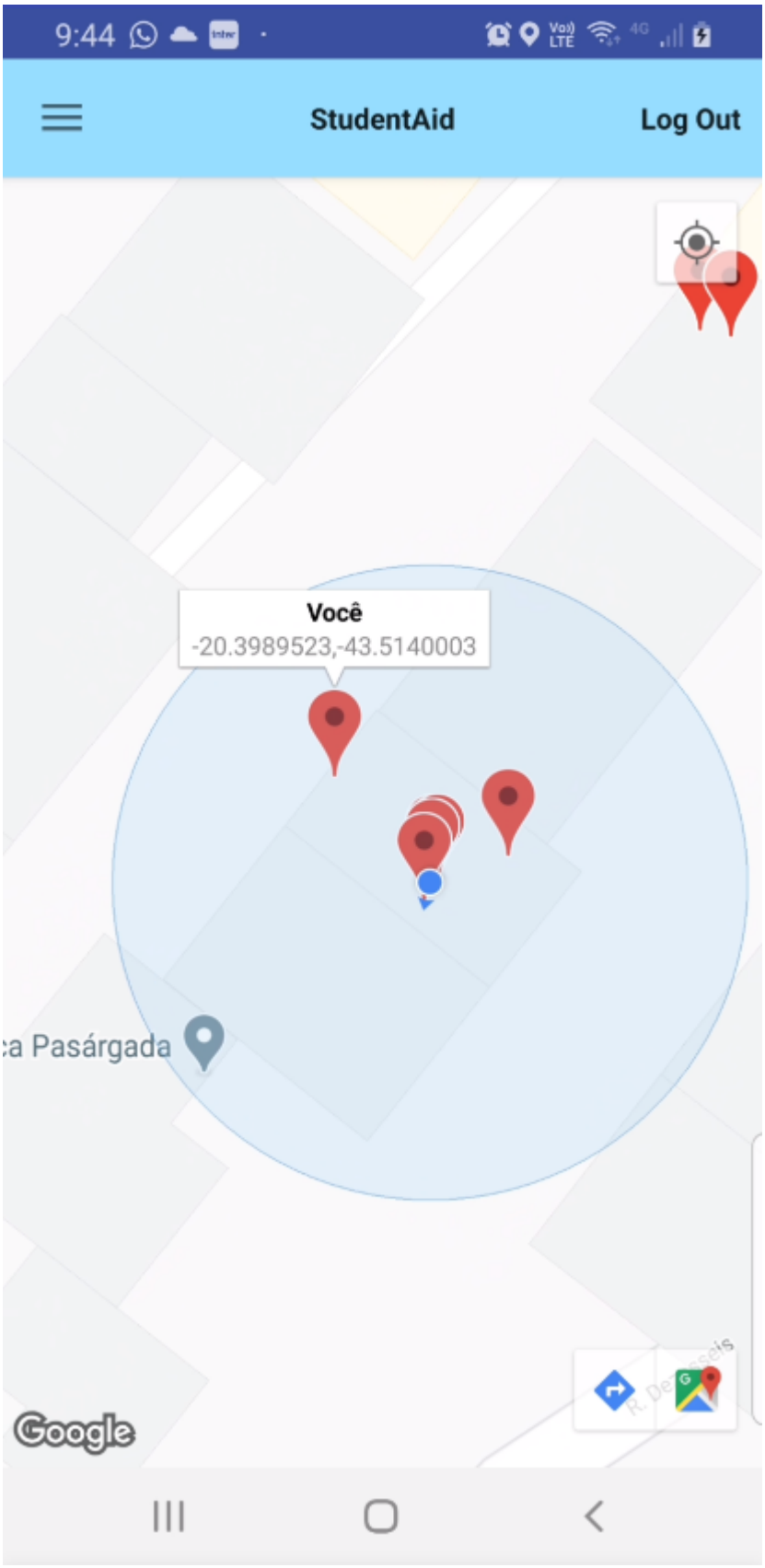

Figura 4.7: Marcador do usuário logado no aplicativo e sua posição real.
Fonte: Desenvolvido pelo autor.

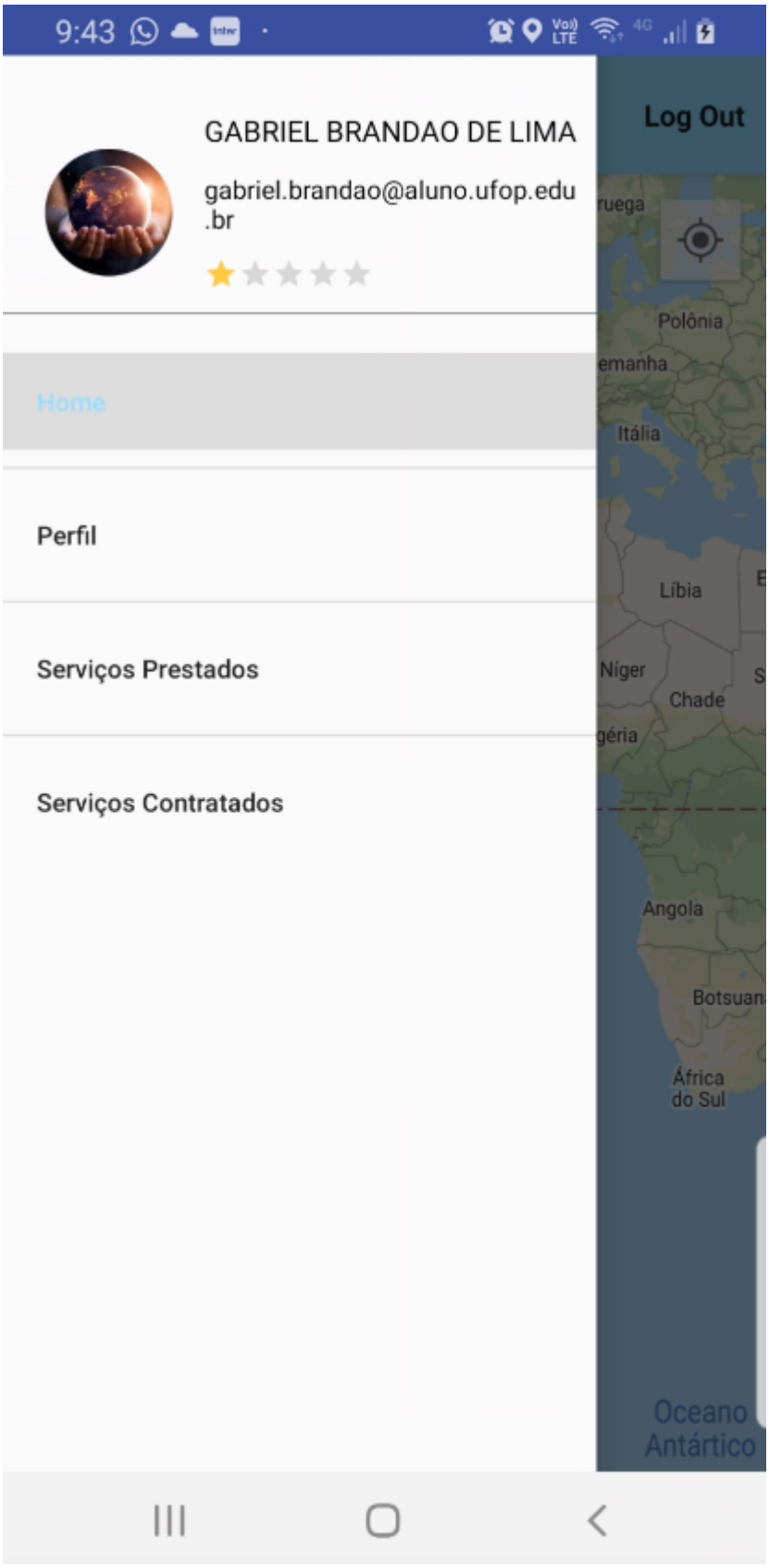

Figura 4.8: Menu lateral com detalhes do usuário e opções de navegação.
Fonte: Desenvolvido pelo autor.

Figura 4.9: Perfil do usuário - autor - logado no app.
Fonte: Desenvolvido pelo autor.

Figura 4.10: Perfil do prestador de serviços.
Informações censuradas para preservar a privacidade do usuário. Fonte: Desenvolvido pelo autor.

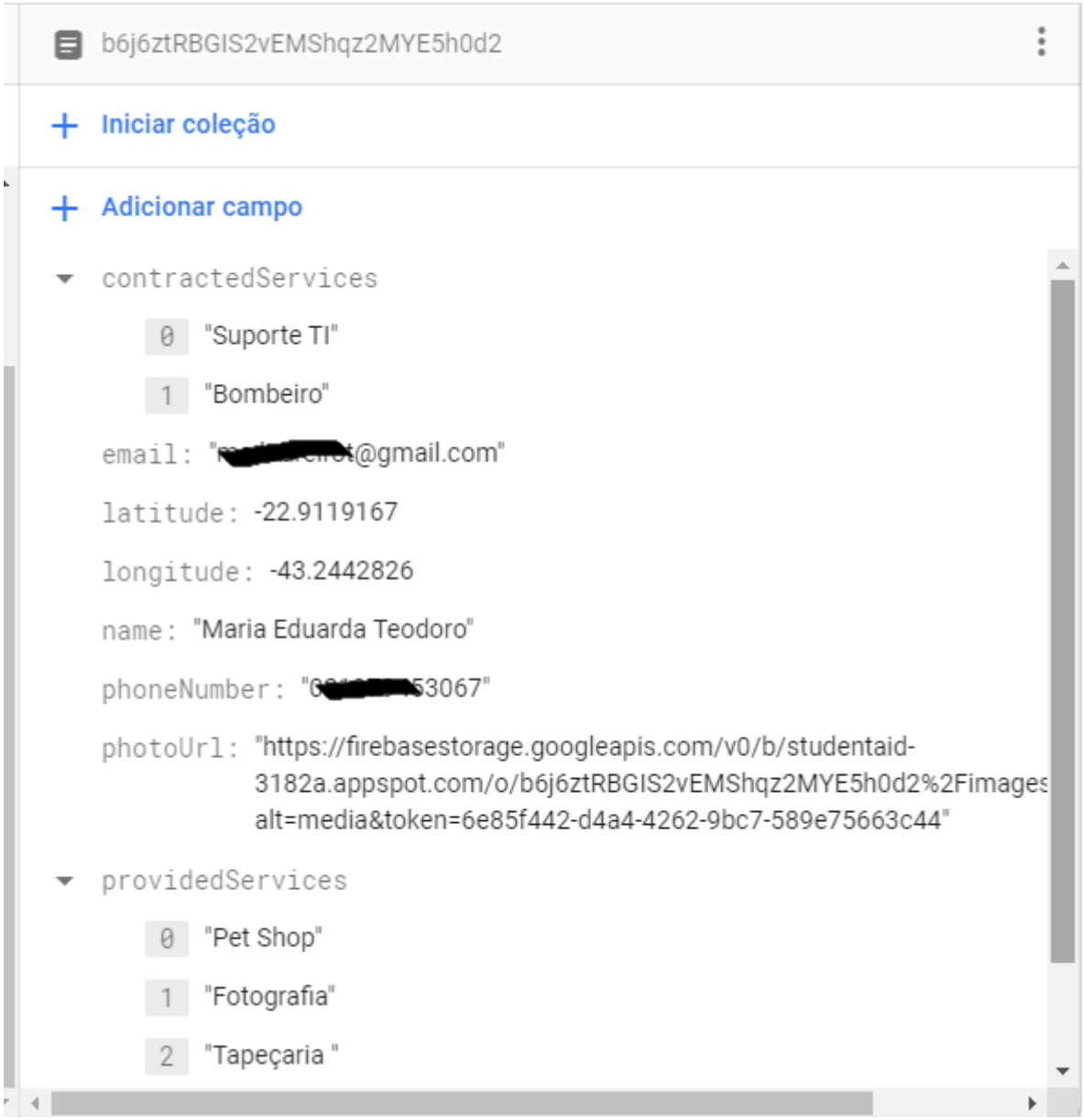

Figura 4.11: Informações do usuário/prestador salvas no banco de dados. **Informações censuradas para preservar a privacidade do usuário.** Fonte: Desenvolvido pelo autor.

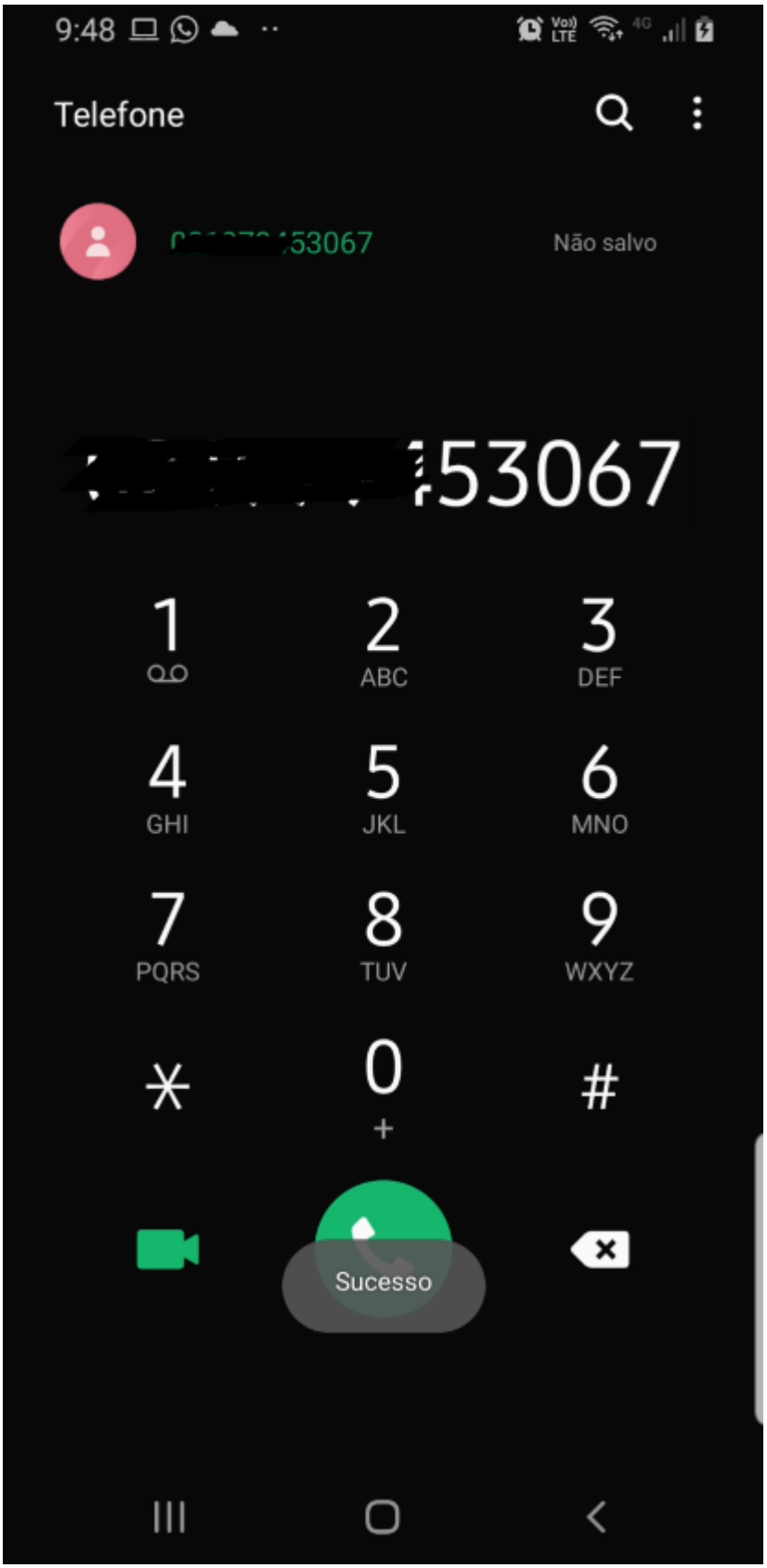

Figura 4.12: Realizar chamada para prestador ao clicar no ícone de telefone vide figura 4.10. **Informações censuradas para preservar a privacidade do usuário.** Fonte: Desenvolvido pelo autor.

4.2.5 Serviços Prestados

Esta tela, figura 4.13, lista todos os serviços prestados, cadastrados pelo usuário no banco de dados. Observa-se na figura 4.11 o atributo *providedServices* do usuário, que representa seus serviços prestados.

Além de listar, o usuário pode cadastrar um novo serviço que deseja prestar, ao clicar no botão "Adicionar Serviço"e realizar o fluxo de adição. Descrito na subseção a seguir, 4.2.6 .

4.2.6 Adicionar Serviço

A figura 4.13 representa a tela inicial do fluxo de adicionar serviço. Ao clicar no botão "Adicionar Serviço", a tela da figura 4.14 é inflada com a opção de adicionar um tipo de serviço ao clicar no ícone de adição '+'.

Na tela de selecionar um serviço, figura 4.15, são listados todos os serviços cadastrados no banco de dados. O usuário deve selecionar um desses serviços, ou adicionar um novo serviço, ao clicar no botão, vide figura 4.16.

Após selecionar e/ou criar um novo serviço, o usuário é redirecionado para a tela anterior com as opções escolhidas infladas logo abaixo de "Tipo de serviço", figura 4.17.

Clicando no botão de confirmar, *checkmark* amarelo, irá aparecer um modal para confirmar os serviços a serem adicionados, figura 4.18, em caso positivo, os mesmos serão gravados no banco de dados e atualizados na lista de serviços prestados da tela anterior, como vê-se na figura 4.19.

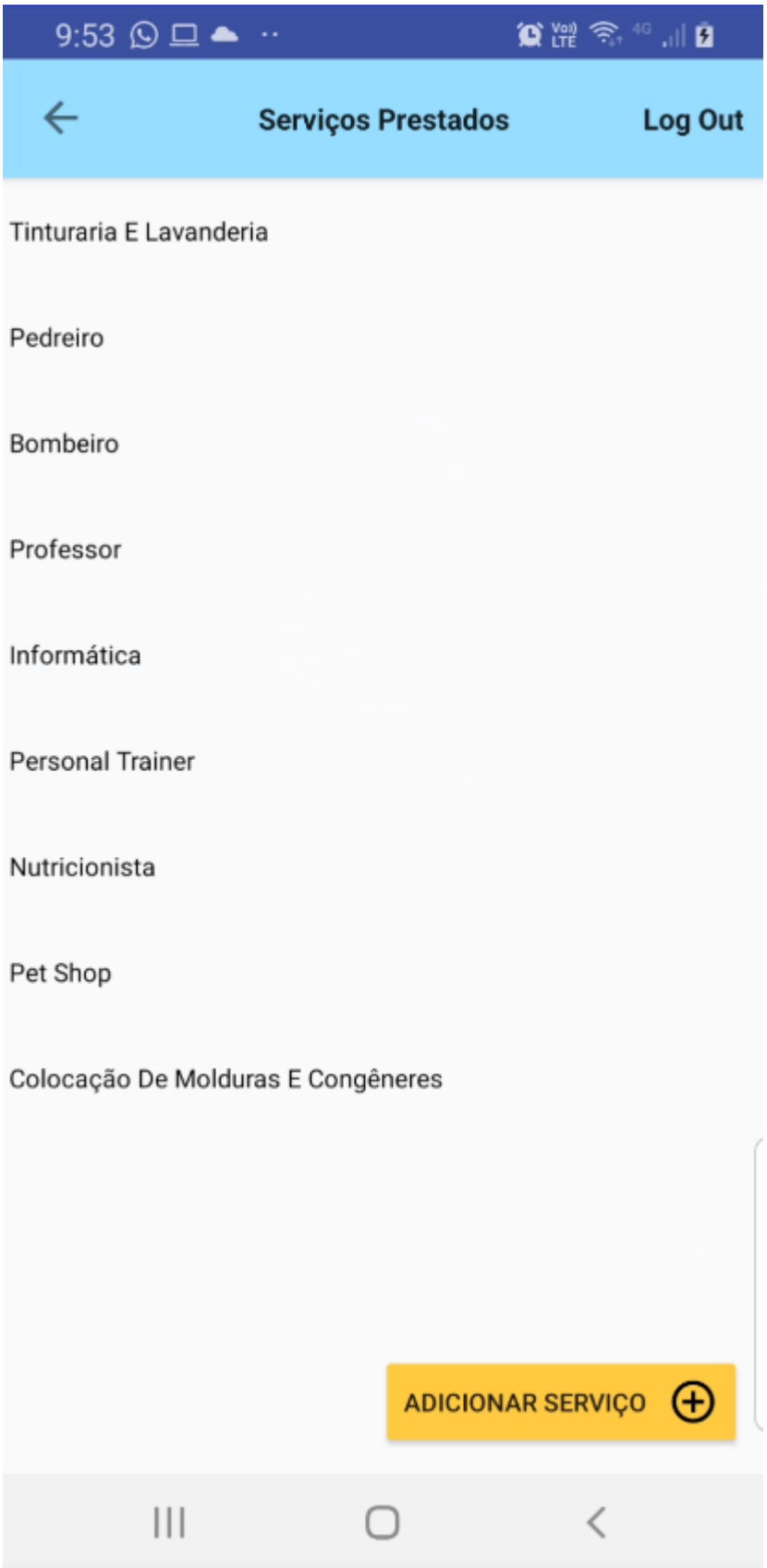

Figura 4.13: Listar Serviços Prestados e Iniciar Fluxo de Adicionar Serviço
Fonte: Desenvolvido pelo autor.

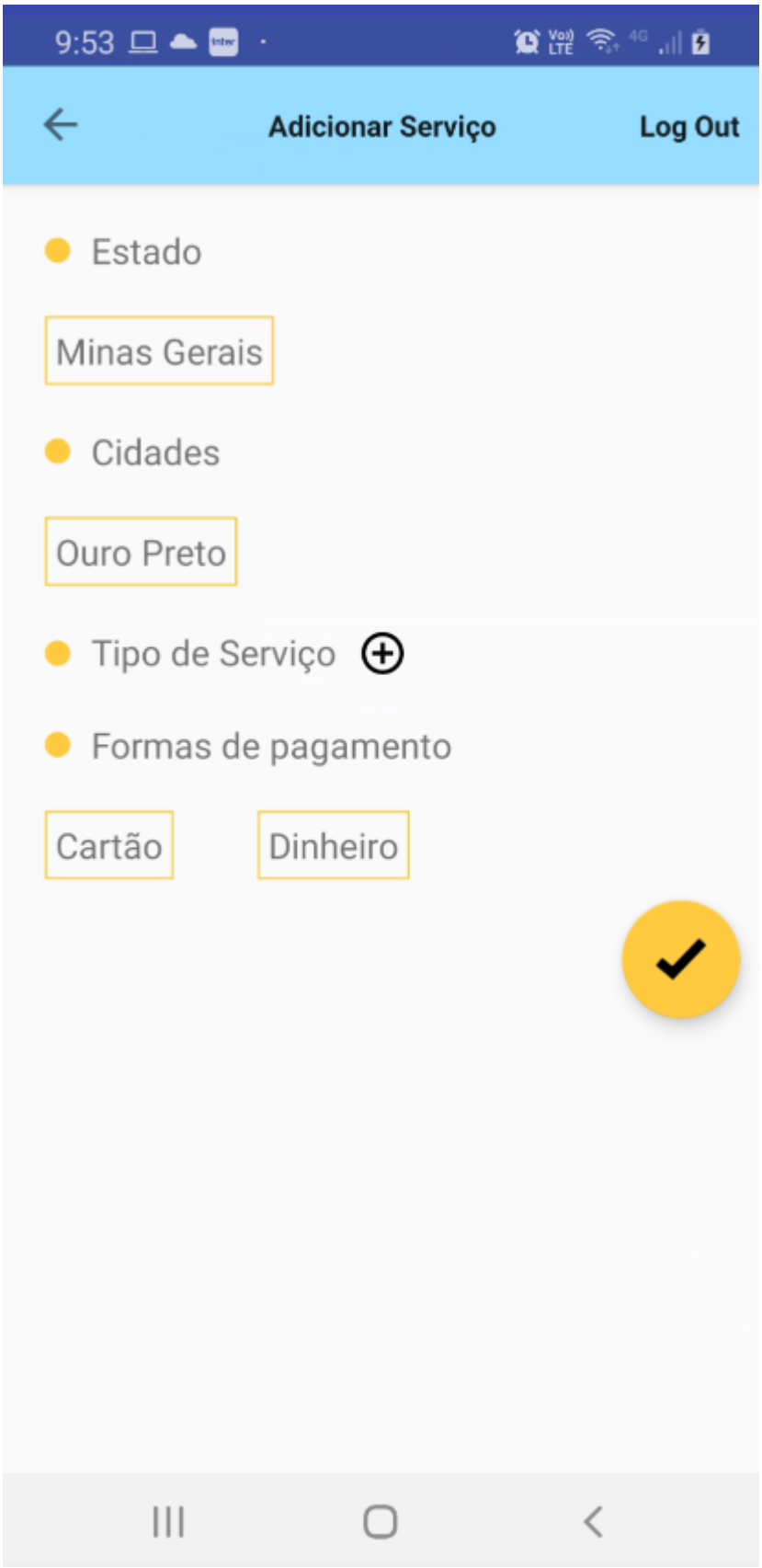

Figura 4.14: Segunda Etapa do Fluxo Adicionar Serviço.
Fonte: Desenvolvido pelo autor.

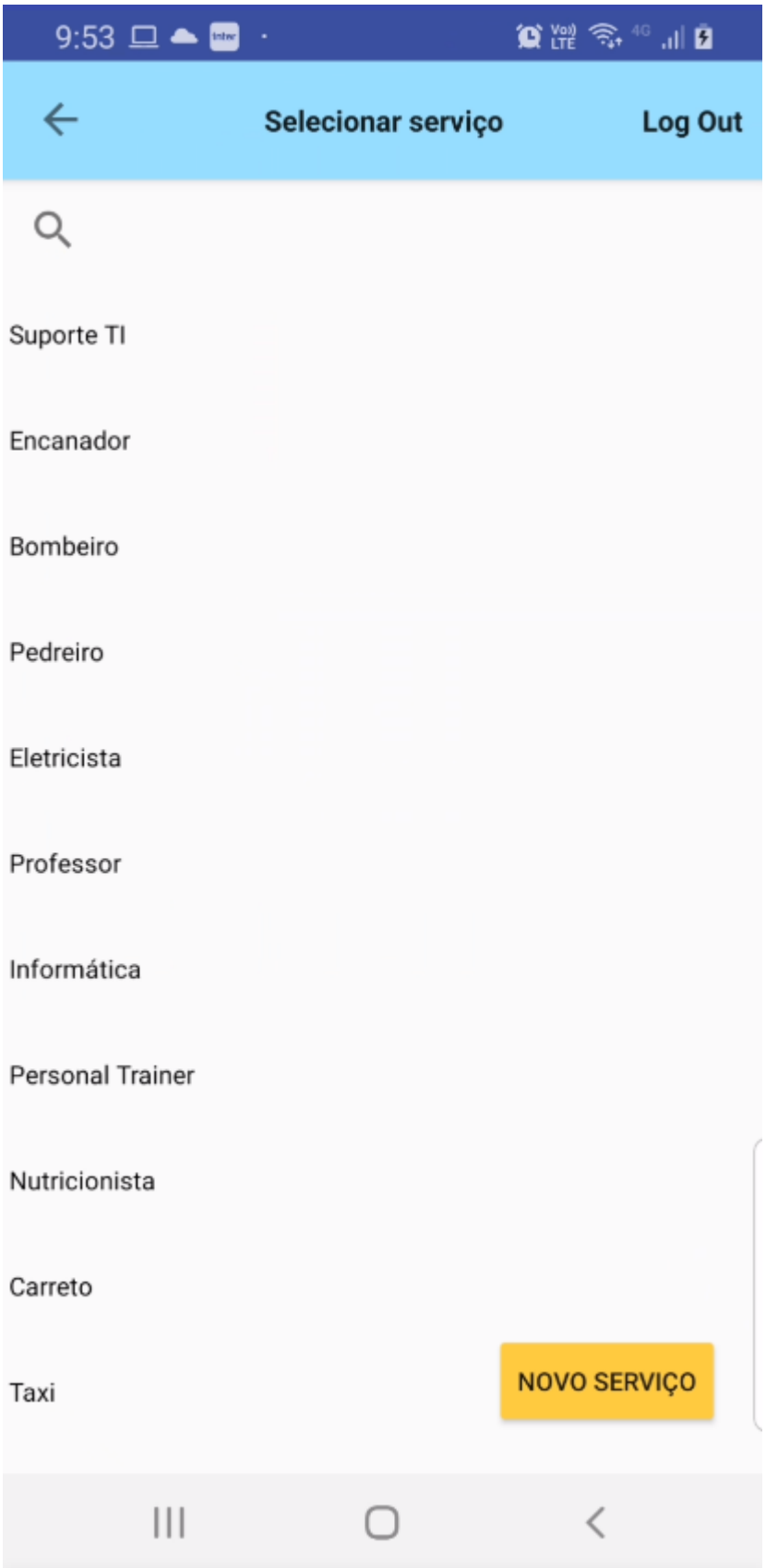

Figura 4.15: Selecionar/Criar serviço a ser prestado.
Fonte: Desenvolvido pelo autor.

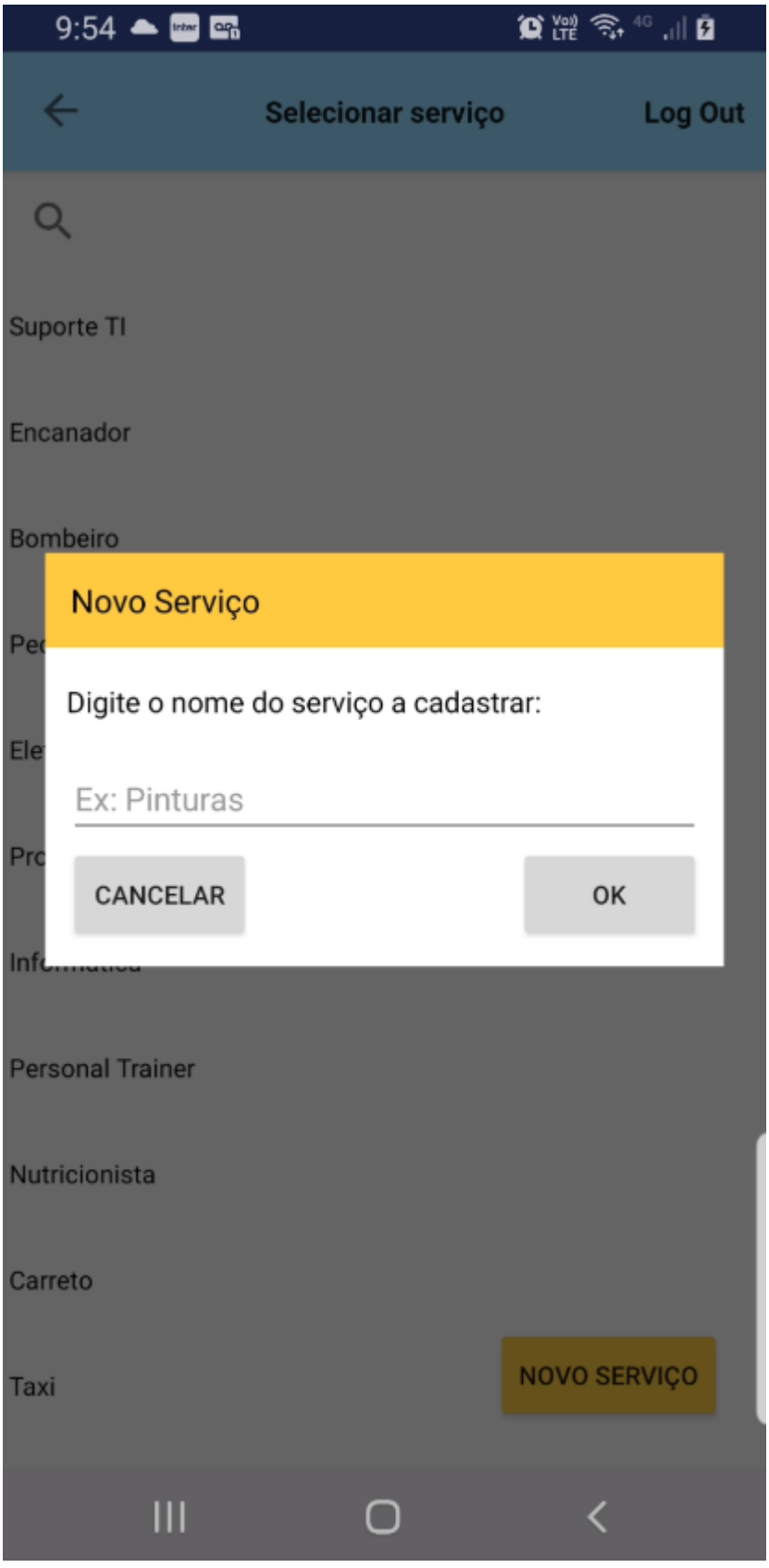

Figura 4.16: Modal para criar novo serviço a ser prestado.
Fonte: Desenvolvido pelo autor.

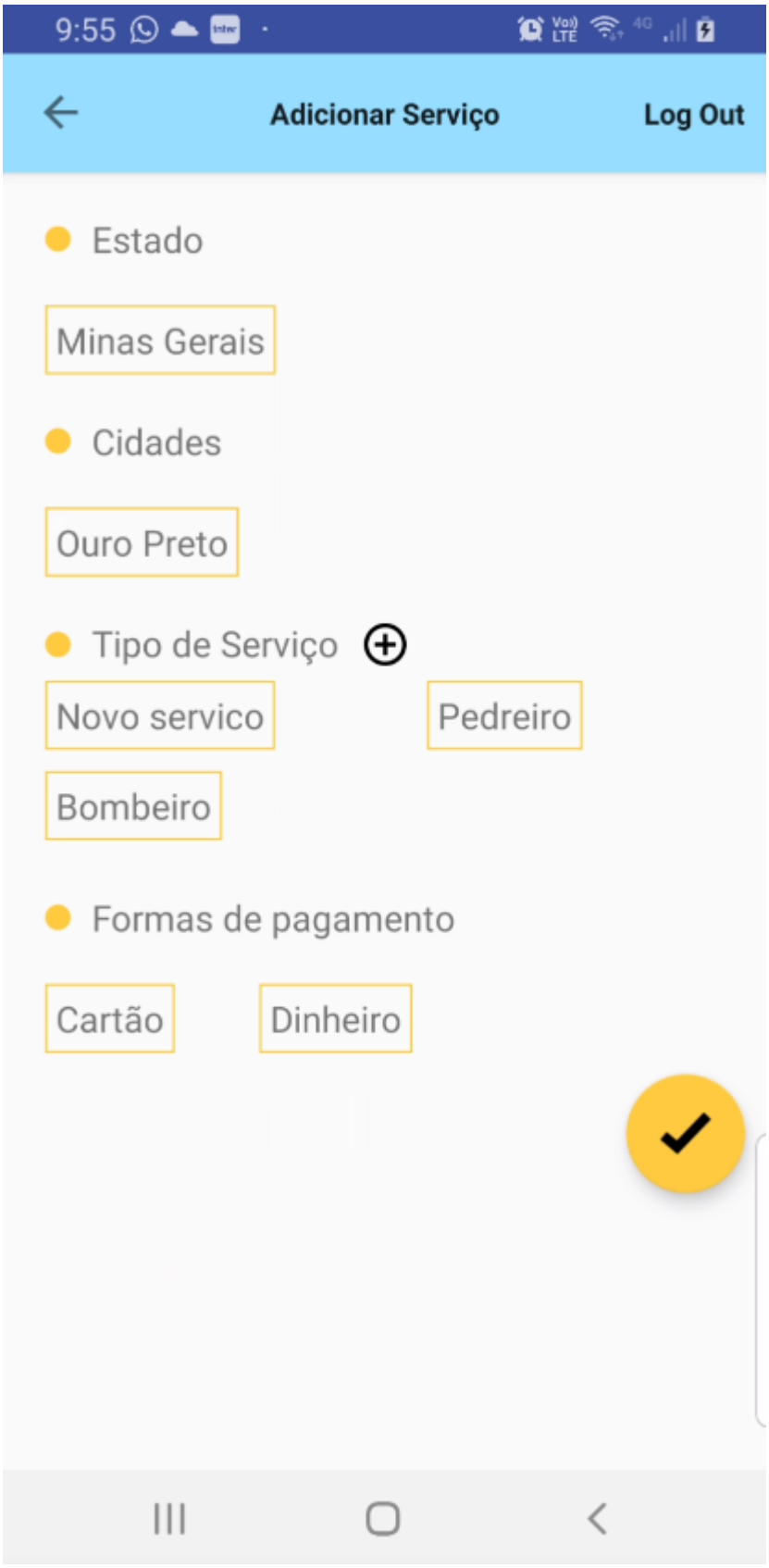

Figura 4.17: Serviços prestados adicionados.
Fonte: Desenvolvido pelo autor.

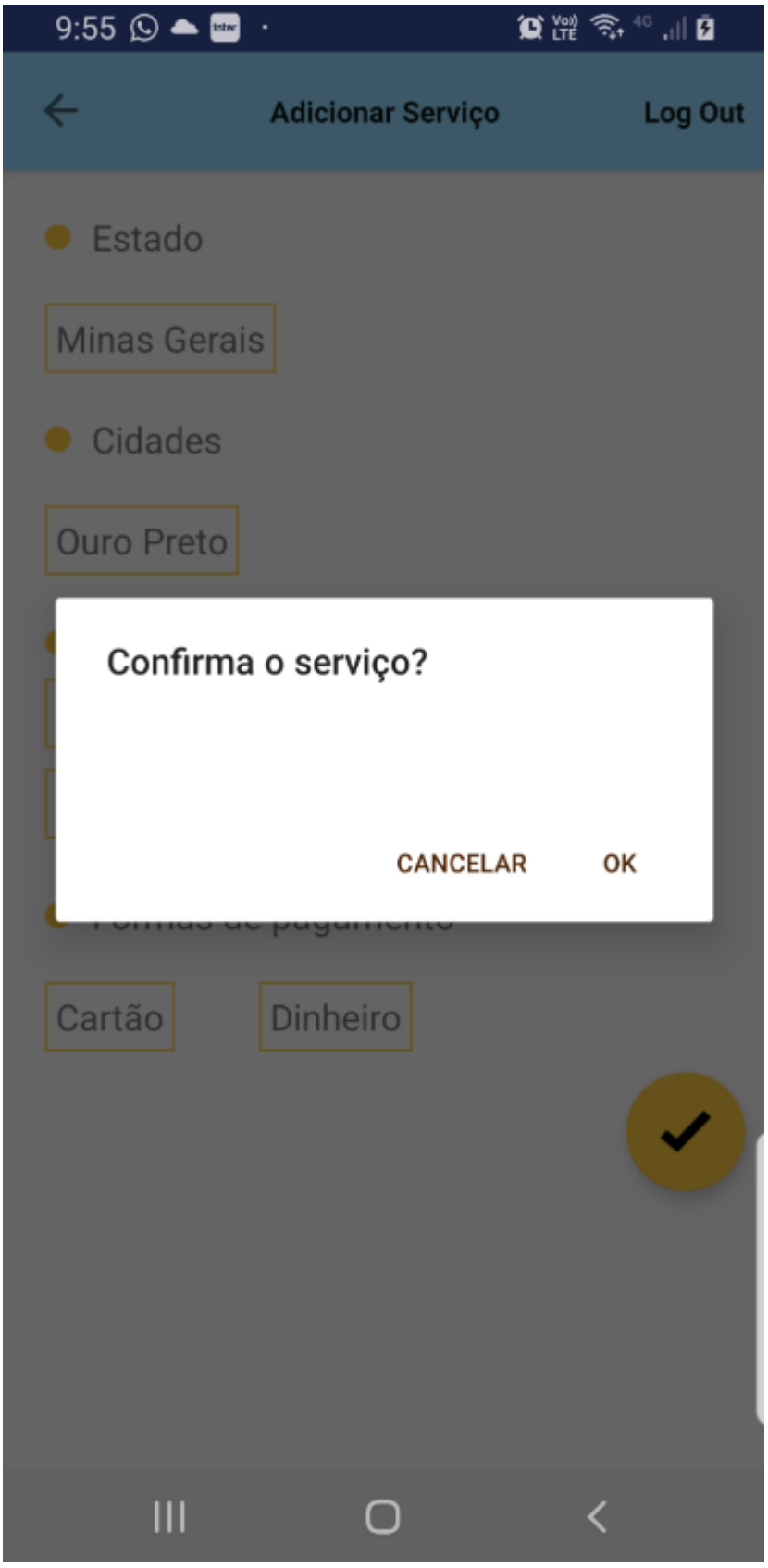

Figura 4.18: Modal de confirmação de criação dos serviços prestados.
Fonte: Desenvolvido pelo autor.

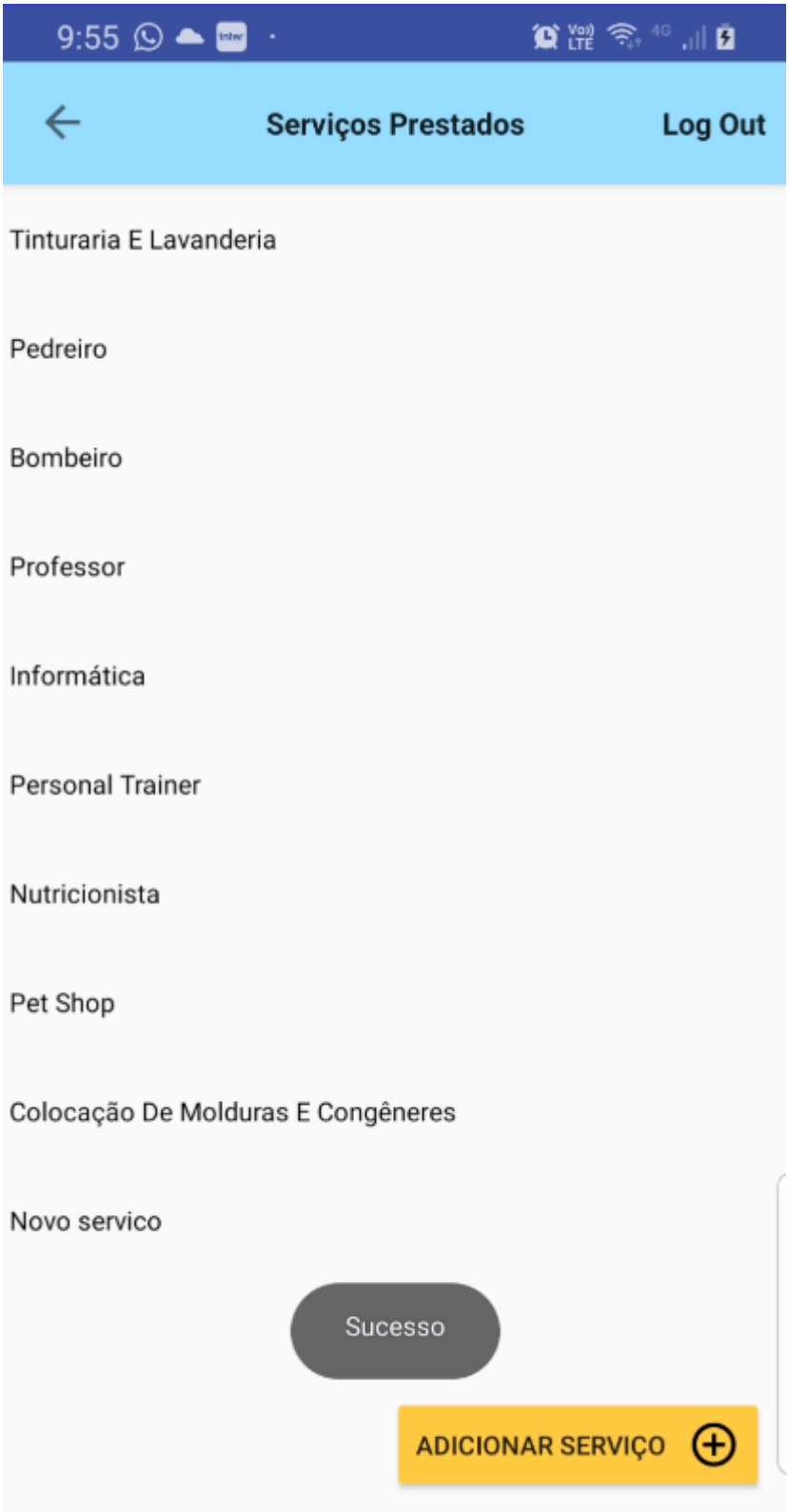

Figura 4.19: Novos serviços adicionados à lista de serviços prestados.
Fonte: Desenvolvido pelo autor.

4.2.7 Serviços Contratados

Esta tela, figura 4.20, lista todos os serviços contratados, cadastrados pelo usuário no banco de dados. Observa-se na figura 4.11 o atributo *contractedServices* do usuário, que representa seus serviços contratados.

Além de listar, o usuário pode contratar um novo serviço, ao clicar no botão "Contratar Serviço"e realizar o fluxo de contratação. Descrito na subseção a seguir, 4.2.8 .

4.2.8 Contratar Serviço

A figura 4.20 representa a tela inicial do fluxo de contratar serviço. Ao clicar no botão "Contratar Serviço", a tela da figura 4.21 é inflada com a lista de todos os usuários cadastrados no banco e seus respectivos serviços prestados, separados por nome e foto do prestador.

Ao selecionar um serviço da lista, o usuário é redirecionado para o perfil daquele prestador com todas as suas informações para contato. A seleção do serviço nesta etapa, é considerada como se o mesmo tivesse sido contratado pelo usuário, sendo assim, a lista de serviços contratados do usuário é atualizada nesse momento no banco de dados e um texto de "Sucesso"pode ser percebido na tela, indicando que a contratação foi salva, como observa-se na figura 4.22.

Após a realização desse fluxo descrito, é possível visualizar o novo serviço contratado ao retornar à tela de Serviços Contratados. Vide figura 4.23.

Figura 4.20: Listar Serviços Contratados e Iniciar Fluxo de Contratar Serviço
Fonte: Desenvolvido pelo autor.

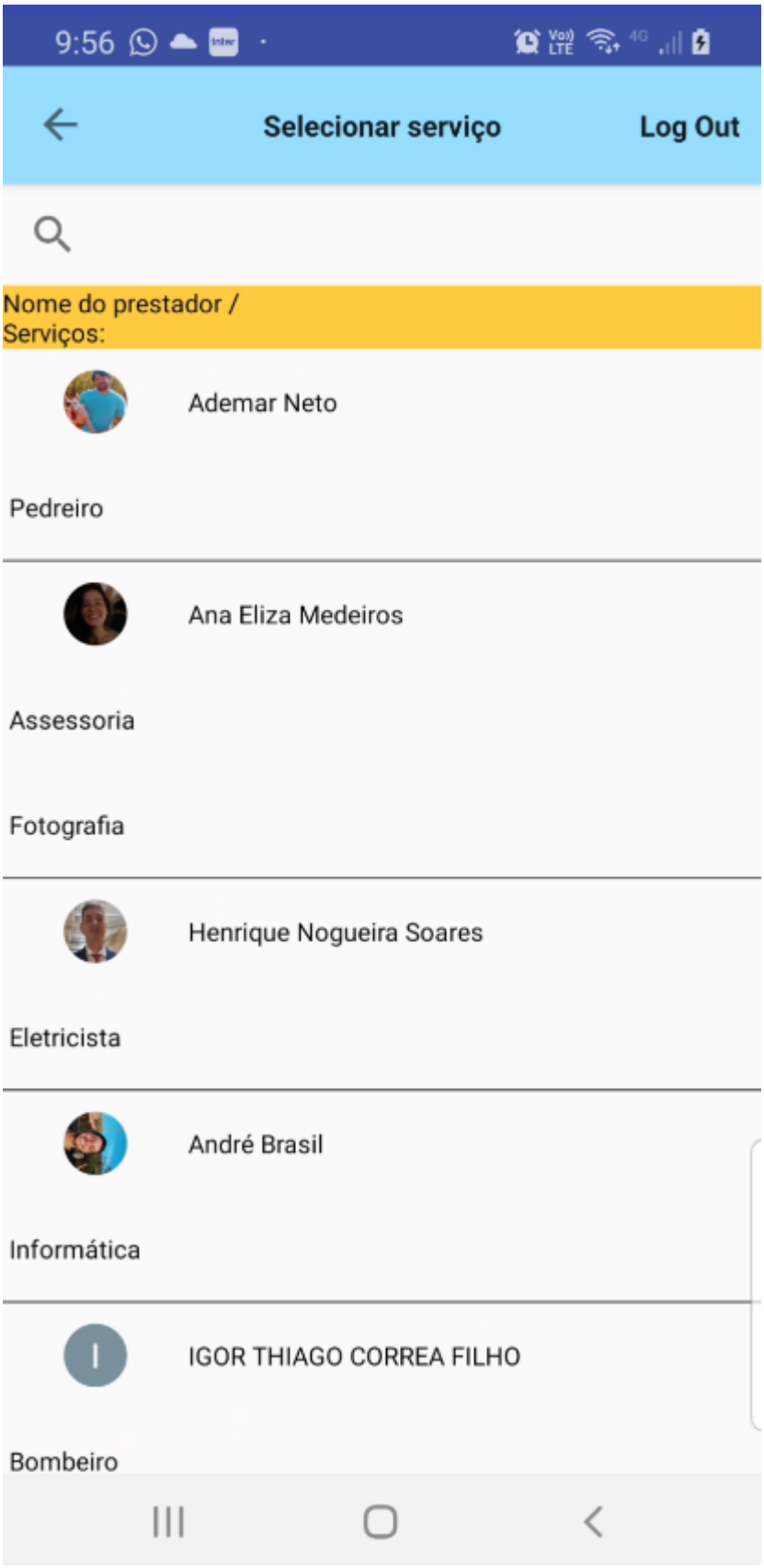

Figura 4.21: Listar prestadores e seus serviços.
Fonte: Desenvolvido pelo autor.

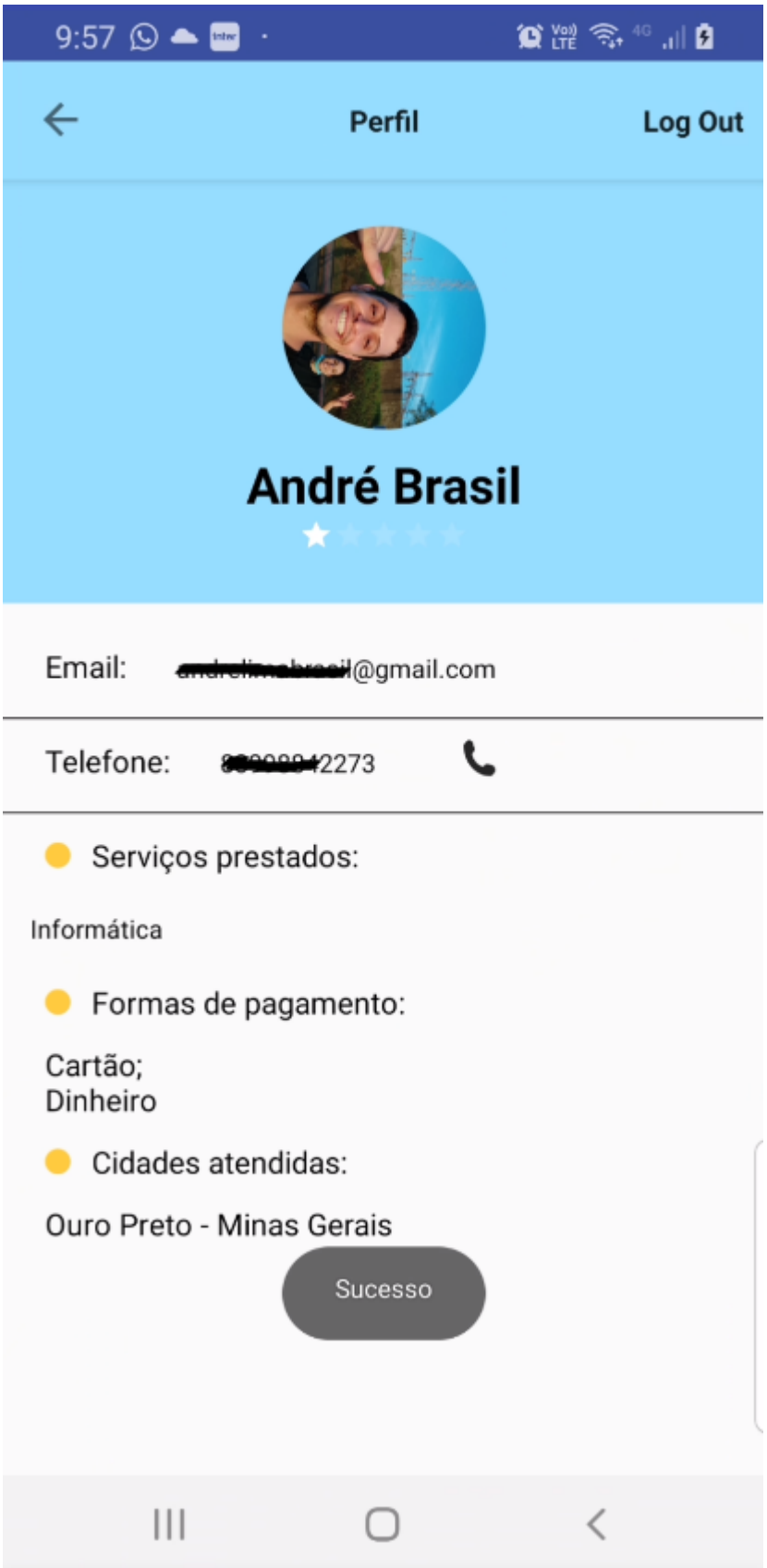

Figura 4.22: Sucesso ao contratar prestador e informações de seu perfil. **Informações censuradas para preservar a privacidade do usuário.** Fonte: Desenvolvido pelo autor.

Figura 4.23: Lista de serviços contratados com novo serviço contratado adicionado.
Fonte: Desenvolvido pelo autor.

Capítulo 5

Conclusão

Como conclusão deste trabalho percebe-se que a revisão bibliográfica foi fundamental para a execução do projeto. Apesar de algumas ferramentas estudadas não terem sido aplicadas, como o serviço *Firebase Realtime Database* apresentado na seção 2.6, toda a erudição adquirida nela serviu como guia de desenvolvimento.

Ao finalizar o desenvolvimento do projeto, foi notado que mesmo uma aplicação pequena e simples pode levar muitas horas de implementação e trazer vários desafios durante esse tempo. Realizando testes em vários dispositivos diferentes, foi possível perceber que ao se programar para o sistema operacional Android, deve-se sempre levar em conta as diversas marcas de dispositivos e antecipar possíveis problemas de hardware nos aparelhos que possa levar ao mal funcionamento do aplicativo, como ocorreu em alguns casos de teste em celulares diferentes.

Comparando com trabalhos relacionados, nota-se que há uma grande demanda em tipos de aplicativos que oferecem serviços sob demanda e que é necessário possuir um *back-end* robusto para a execução e infraestrutura desses aplicativos.

5.1 Trabalhos Futuros

Como perspectivas de trabalhos futuros podem ser consideradas:

- A publicação do aplicativo na loja da Google Play em fase de testes, alpha ou beta, para os estudantes da universidade testarem o aplicativo;
- Implementar funcionalidade de avaliar o serviço do prestador;
- Avaliar o engajamento e desempenho da aplicação, para sua escalabilidade;
- Monetizar o aplicativo, integrando um framework de pagamento in-app;
- Estruturar um *back-end* ideal para otimizar os serviços e acrescentar funcionalidades, como as avaliações dos usuários;

- Implementar testes instrumentais e automatizados do aplicativo;

Referências Bibliográficas

ADJUST, A. I. (2019). Adjust global app trends 2019. Disponível em: https://www.adjust.com/thank-you/ebooks/?download=https%3A%2F%2Fgo.adjust.com%2Fadjust-global-app-trends-2019. Acesso em: 18/11/2019 00:13.

AMAZON, A. (2021). O que é nosql? Disponível em: https://aws.amazon.com/pt/nosql/. Acesso em: 27/04/2021 21:32.

ARTY, D. (2014). Ux – user experience. descubra o que é user experience e como ele pode ajudar em seus projetos. Disponível em: https://www.chiefofdesign.com.br/user-experience/. Acesso em: 28/11/2019 23:35.

ARTY, D. (2018). Ui e ux design? o que significa cada coisa? quais as diferenças e semelhanças entre eles? Disponível em: https://www.chiefofdesign.com.br/ux-design-e-ui-design/. Acesso em: 28/11/2019 22:21.

BERGHER, R. (2020). Do 1g ao 5g: conheça a evolução da internet no celular. Disponível em: https://www.zoom.com.br/celular/deumzoom/do-1g-ao-5g-evolucao-internet-no-celular. Acesso em: 13/03/2021 18:32.

BRASIL, G. F. (2013). Agenda brasileira para a indústria 4.0. Disponível em: http://www.industria40.gov.br/. Acesso em: 06/09/2019 09:37.

COMPUTERWORLD, I. (2019). As 6 melhores linguagens de programação para desenvolver um app android. Disponível em: https://computerworld.com.br/carreira/as-6-melhores-linguagens-de-programacao-para-desenvolver-um-app-android/. Acesso em: 27/04/2021 19:23.

COSSETI, M. C. (2019). Oi enfrenta desafio de conectar rock in rio e faz teste 5g. Disponível em: https://tecnoblog.net/310075/oi-enfrenta-desafio-de-conectar-rock-in-rio-e-faz-teste-5g/. Acesso em: 18/11/2019 18:47.

CROCKFORD, D. (2021). Introdução ao json. Disponível em: https://www.json.org/json-pt.html. Acesso em: 27/04/2021 21:02.

FGV EASP, C. d. T. d. I. A. (2019). 30ª pesquisa anual do uso de ti nas empresas, 2019. Disponível em:https://eaesp.fgv.br/ensinoeconhecimento/centros/cia/pesquisa e https://eaesp.fgv.br/sites/eaesp.fgv.br/files/noticias2019fgvcia_2019.pdf. Acesso em: 18/11/2019 23:13.

FIGUEIREDO, C. M. S. e NAKAMURA, E. (2003). Computação móvel: Novas oportunidades e novos desafios. *T&C Amazônia*, 2:16–28.

FONSECA, M. (2017). Afinal, por que tantos negócios copiam o modelo do uber? Disponível em: https://exame.abril.com.br/pme/afinal-por-que-tantos-negocios-copiam-o-modelo-do-uber/. Acesso em: 28/11/2019 18:47.

FORBES, B. (2016). Uberização à brasileira: os apps que transformam pessoas em companhias. Disponível em: https://www.forbes.com.br/negocios/2016/09/uberizacao-a-brasileira-os-apps-que-transformam-pessoas-em-companhias/. Acesso em: 13/03/2021 19:33.

FRENKEN, K. e SCHOR, J. (2017). Putting the sharing economy into perspective. *Environmental Innovation and Societal Transitions*, 23:3–10.

GALANTE, V. R. (2018). Uberização: Entenda tudo sobre esse processo. Disponível em: https://usemobile.com.br/uberizacao-entenda-tudo-sobre-esse-processo/. Acesso em: 28/11/2019 19:20.

GATTAL, A. (2018). Understand android basics part 1: Application, activity and lifecycle. Disponível em: https://medium.com/@Abderraouf/understand-android-basics-part-1-application-activity-and-lifecycle-b559bb1e40e. Acesso em: 22/11/2019 20:12.

GETNINJAS, S. d. I. (2021). Getninjas. Disponível em: https://www.getninjas.com.br/. Acesso em: 13/03/2021 20:17.

GLOBALSTATS, S. (2018-2019). Mobile operating system market share worldwide. Disponível em: https://gs.statcounter.com/os-market-share/mobile/worldwide. Acesso em: 21/11/2019 17:13.

GOOGLE, G. I. (2019). Fundamentos de aplicativos. Disponível em: https://developer.android.com/guide/components/fundamentals?hl=pt-BR. Acesso em: 22/11/2019 17:52.

GOOGLE DEVELOPERS, G. (2018). Use o android jetpack para acelerar o desenvolvimento de aplicativos. Disponível em: https://developers-br.googleblog.com/2018/05/use-o-android-jetpack-para-acelerar-o.html. Acesso em: 23/11/2019 11:43.

GOOGLE DEVELOPERS, G. (2019a). Android jetpack. Disponível em: https://developer.android.com/jetpack?hl=pt-BR. Acesso em: 25/11/2019 22:06.

GOOGLE DEVELOPERS, G. (2019b). Armazene e sincronize dados em tempo real. Disponível em: https://firebase.google.com/products/realtime-database/?hl=pt-br. Acesso em: 26/11/2019 23:03.

GOOGLE DEVELOPERS, G. (2019c). Firebase realtime database. Disponível em: https://firebase.google.com/docs/database/?hl=pt-br. Acesso em: 26/11/2019 23:33.

GOOGLE DEVELOPERS, G. (2019d). Get started with google maps platform. Disponível em: https://developers.google.com/maps/gmp-get-started. Acesso em: 26/11/2019 22:28.

GOOGLE DEVELOPERS, G. (2019e). Maps sdk for android. Disponível em: https://developers.google.com/maps/documentation/android-sdk/intro. Acesso em: 26/11/2019 22:11.

GOOGLE DEVELOPERS, G. (2019f). O firebase ajuda as equipes de aplicativos para dispositivos móveis e da web a alcançar o sucesso. Disponível em: https://firebase.google.com/?hl=pt-br. Acesso em: 26/11/2019 22:48.

GOOGLE DEVELOPERS, G. (2020). Build.version_codes. Disponível em: https://developer.android.com/reference/android/os/Build.VERSION_CODES#LOLLIPOP. Acesso em: 01/05/2020 18:05.

GOOGLE DEVELOPERS, G. (2021a). Choose a database: Cloud firestore or realtime database | firebase. Disponível em: https://firebase.google.com/docs/database/rtdb-vs-firestore. Acesso em: 05/03/2021 22:40.

GOOGLE DEVELOPERS, G. (2021b). Cloud firestore | firebase. Disponível em: https://firebase.google.com/docs/firestore. Acesso em: 01/03/2021 19:23.

GOOGLE DEVELOPERS, G. (2021c). Firebase authentication | firebase. Disponível em: https://firebase.google.com/docs/auth. Acesso em: 05/03/2021 21:57.

GOOGLE DEVELOPERS, G. (2021d). Meet android studio. Disponível em: https://developer.android.com/studio/intro. Acesso em: 31/03/2021 18:19.

GRIN, T. (2019). Kotlin programming language. Disponível em: https://kotlinlang.org/assets/kotlin-media-kit.pdf. Acesso em: 23/11/2019 00:12.

GURU, R. (2021). O que é um padrão de projeto? Disponível em: https://refactoring.guru/pt-br/design-patterns/what-is-pattern. Acesso em: 27/04/2021 18:22.

HAMARI, J.; SJÖKLINT, M. e UKKONEN, A. (2015). The sharing economy: Why people participate in collaborative consumption. *Journal of the Association for Information Science and Technology*, 67:2047–2059.

HOWE, D. (2021). Application program interface. Disponível em: `http://foldoc.org/Application+Program+Interface`. Acesso em: 27/04/2021 21:23.

IBGE, I. B. d. G. e. E. (2019). Projeções e estimativas da população do brasil e das unidades da federação. Disponível em: `https://www.ibge.gov.br/apps/populacao/projecao/index.html/`. Acesso em: 18/11/2019 23:33.

KITPITAK, B. (2020). Reasons to use android single-activity architecture with navigation component. Disponível em: `https://oozou.com/blog/reasons-to-use-android-single-activity-architecture-with-navigation-component-36`. Acesso em: 30/03/2021 22:03.

KOTLIN ORG, J. (2019). Using kotlin for android development. Disponível em: `https://kotlinlang.org/docs/reference/android-overview.html`. Acesso em: 23/11/2019 00:12.

LEITE, A. F. (2005). Conheça os padrões de projeto. Disponível em: `https://www.devmedia.com.br/conheca-os-padroes-de-projeto/957`. Acesso em: 28/04/2021 17:21.

MARTIN, C. J. (2016). The sharing economy: A pathway to sustainability or a nightmarish form of neoliberal capitalism? *Ecological Economics*, 121:149–159.

MATSUMOTO, P. M. (2005). Computação móvel. Disponível em: `https://bcc.ime.usp.br/tccs/2005/patty/tecnica/computacaoMovel.html`. Acesso em: 13/03/2021 17:12.

MENDONÇA, F. (2015). 7 guerras que estão mudando o mundo para conquistar você. Disponível em: `https://administradores.com.br/noticias/7-guerras-que-estao-mudando-o-mundo-para-conquistar-voce`. Acesso em: 13/03/2021 19:40.

MINDS, L. (2017). O que é computação móvel e como você pode fazer uso dessa tecnologia? Disponível em: `http://logicalminds.com.br/o-que-e-computacao-movel-e-como-voce-pode-fazer-uso-dessa-tecnologia/`. Acesso em: 13/03/2021 17:21.

NGULELE, E. (2018). Introdução ao android jetpack. Disponível em: `https://medium.com/android-dev-moz/introdu%C3%A7%C3%A3o-ao-android-jetpack-68697c5fcff2`. Acesso em: 25/11/2019 22:10.

NORONHA, R. (2013). Funções de alta ordem – programação funcional "no mundo real". Disponível em: https://www.lambda3.com.br/2013/12/funcoes-de-alta-ordem-programacao-funcional-no-mundo-real/. Acesso em: 27/04/2021 20:04.

ORGAZ, C. J. (2019). 3 grandes vantagens do 5g que mudarão para sempre nossa experiência na internet - bbc news brasil. Disponível em: https://www.bbc.com/portuguese/geral-48499353. Acesso em: 03/10/2019 16:17.

PAUL, J. (2018). Top 5 kotlin programming courses for java and android programmers. Disponível em: https://hackernoon.com/top-5-kotlin-programming-courses-for-java-and-android-programmers-49e842b8af1a. Acesso em: 23/11/2019 09:36.

PORTO, A. M. d. S. e SOARES, A. B. (2017). Diferenças entre expectativas e adaptação acadêmica de universitários de diversas áreas do conhecimento. *Análise Psicológica*, 35 nº1 Lisboa - Disponível em: http://publicacoes.ispa.pt/index.php/ap/article/view/1170. Acesso em: 20/11/2019 17:43.

RICCIARDI, A. (2016). Uberização à brasileira: os apps que transformam pessoas em companhias. Disponível em: https://forbes.com.br/fotos/2016/09/uberizacao-a-brasileira-os-apps-que-transformam-pessoas-em-companhias/. Acesso em: 28/11/2019 18:47.

SAIYED, A. (2018). What is android jetpack? Disponível em: https://android.jlelse.eu/what-is-android-jetpack-737095e88161. Acesso em: 25/11/2019 22:19.

SILVEIRA, C. B. (2018). O que é indústria 4.0 e como ela vai impactar o mundo. Disponível em: https://www.citisystems.com.br/industria-4-0/. Acesso em: 06/09/2019 09:43.

SMIEH (2019). Anatomy physiology of an android. Disponível em: https://www.d.umn.edu/~tcolburn/cs2511/slides.new/android/android.xhtml. Acesso em: 22/11/2019 17:02.

SOUTO, T. (2018). Arquiteturas em android : Mvvm + kotlin + retrofit parte 1. Disponível em: https://othiagosouto.medium.com/arquiteturas-em-android-mvvm-kotlin-retrofit-parte-1-2ac77c8a26. Acesso em: 28/04/2021 17:42.

TRIPATHI, A. (2019). What's new in android @ i/o'19. Disponível em: https://medium.com/mindorks/whats-new-in-android-i-o-19-aabb0a59a3ae. Acesso em: 23/11/2019 00:17.

UBER TECHNOLOGIES, U. I. (2019). Quem somos. Disponível em: https://www.uber.com/br/pt-br/about/. Acesso em: 28/11/2019 20:26.