UNIVERSIDADE DO VALE DO SAPUCAÍ

DIEGO FRAGA DE OLIVEIRA

RAFAEL SANA MONTEVECHIO

TIPS: APLICATIVO DE DIVULGAÇÃO DE SERVIÇOS

POUSO ALEGRE, MG

2019

DIEGO FRAGA DE OLIVEIRA

RAFAEL SANA MONTEVECHIO

TIPS: APLICATIVO DE DIVULGAÇÃO DE SERVIÇOS

Projeto de conclusão de curso do curso de Sistemas de Informação da Universidade do Vale do Sapucaí como requisito parcial para a obtenção do título de bacharel em Sistemas de Informação.

Orientador: Prof. Rodrigo Luís de Faria.

POUSO ALEGRE, MG

2019

De Oliveira, Diego Fraga; Montevechio, Rafael Sana

SISTEMA DE ANÁLISE DE REDAÇÕES DE CANDIDATOS PARA MONITORIA / Denise Aparecida Gomes dos Santos; Luiz Henrique de Souza Borges – Pouso Alegre, MG: Univás, 2019.

73 f. il.

Trabalho de Conclusão de Curso (graduação) – universidade do Vale do Sapucaí, Univás, Sistemas de Informação.

Orientador: Prof. Rodrigo Luís de Faria

1 – (aplicativo). 2- (autônomos). 3 – (profissionais).

DIEGO FRAGA DE OLIVEIRA

RAFAEL SANA MONTEVECHIO

TIPS: APLICATIVO PARA A DIVULGAÇÃO DE SERVIÇOS

Trabalho de Conclusão de Curso apresentado à banca examinadora como requisito parcial para obtenção do título de Bacharel em Sistemas de Informação na Universidade do Vale do Sapucaí – Faculdade de Filosofia, Ciências e Letras Eugênio Pacelli, Pouso Alegre – MG. Orientado pelo Prof. Rodrigo Luís de Faria.

Aprovado em / / 2019

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. Rodrigo Luís de Faria

Orientador

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. (Titulação) (Nome)

Examinador

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Prof. (Titulação) (Nome)

Examinador

DEDICATÓRIA

Dedicamos este trabalho primeiramente a Deus por nos permitir esta conquista em nossas vidas, por nos ter permitido a conclusão de um curso superior. Dedicamos aos familiares e amigos, que sempre estiveram por perto para nos ajudar em nossas quedas, comemorar as nossas pequenas conquistas e não nos deixar desanimar, e, por fim, aos professores que nos passaram seus ensinamentos durante todo o curso e aos colegas de classe, pela amizade e companheirismo.

AGRADECIMENTOS

Agradecemos, a Deus por esta conquista, ao orientador deste projeto Prof. Rodrigo Luís de Faria pela sua experiência, competência e sempre disposto a nos ajudar, e à professora Denise Aparecida Gomes dos Santos pela ajuda e correção durante todo o processo de escrita da documentação.

Importante agradecer aos ex-colegas de curso, que com o seu TCCs ajudaram-nos servindo como base para desenvolvermos a parte de documental.

De (Diego Fraga de Oliveira)

De (Rafael Sana Montevechio)

*O trabalho vai preencher uma grande parte da sua vida, e a única maneira de ficar completamente satisfeito é fazer o que você acredita ser um bom trabalho. E a única forma de fazer um bom trabalho é amar aquilo que você faz. Se você ainda não descobriu o que é, continue procurando. Não se acomode. Da mesma forma que acontece com as coisas do coração, você vai saber quando encontrar.*

*(Steve Jobs)*

FRAGA, Diego; MONTEVECHIO, Rafael Sana. TIPS. 2019. Projeto de pesquisa para trabalho de conclusão de curso – Curso de SISTEMAS DE INFORMAÇÃO, Universidade do Vale do Sapucaí, Pouso Alegre – MG, 2019.

**RESUMO**

Muitas pessoas têm dificuldades em encontrar profissionais adequados para realizar algum tipo de serviço. Observando essa necessidade, este projeto tem como objetivo criar um aplicativo mobile no qual os mais variados profissionais autônomos poderão criar uma conta, um perfil e oferecer seus serviços e também realizar buscas. O TIPS foi proposto com o intuito de facilitar, tanto para os usuários quanto para os profissionais, localizarem de forma rápida, fácil e gratuita. Com isso, o usuário terá um aplicativo, por meio do qual poderá criar o seu perfil, buscar e analisar o melhor perfil profissional de acordo suas informações e avaliações. Depois de encontrado, o utilizador poderá realizar um contato com este profissional para fecharem a contratação do serviço. Após o término da execução deste serviço, o cliente e o prestador de serviço devem se avaliar no aplicativo, embora não haja vínculo algum entre as partes e o aplicativo. Para o seu desenvolvimento, serão utilizadas algumas ferramentas e plataformas que estão em alta no mercado atualmente, tais como, Ionic Framework, Angular e Firebase. A metodologia desenvolvida se pautou na aplicada, por permitir a aplicação dos conhecimentos adquiridos no curso de graduação de Sistema de Informação – Univás para produção de um produto. Por meio das tecnologias utilizadas podemos contar com um aplicativo. Este software tem por finalidade ajudar clientes e profissionais a ficarem mais próximos. Os resultados demonstram uma aplicação a qual facilite para as pessoas realizarem buscas por profissionais, aos profissionais, mais um canal para divulgação de seus serviços, e aos desenvolvedores o contato com tecnologias que estão em alta no mercado.

**Palavras-chaves:** Prestadores de serviços. Aplicação para Prestadores de Serviços. Avaliação de Serviços. Cadastro de Prestadores de Serviços.

FRAGA, Diego; MONTEVECHIO, Rafael Sana. TIPS. 2019. Projeto de pesquisa para trabalho de conclusão de curso – Curso de SISTEMAS DE INFORMAÇÃO, Universidade do Vale do Sapucaí, Pouso Alegre – MG, 2019.

**ABSTRACT**

Many people have issues finding adequate professionals to carry out a type of service. Noting this need, this project aims to create a *mobile* application in which various freelance professionals can create an account, a profile and offer their services, as well as performing searches. TIPS was proposed to make it easy for both users and professionals to locate quickly, easily and for free. With this, the user will have an application, through which they can create their profile, search and analyze the best professional one, according to their information and ratings. Once found, the user can make a contact with this professional to close the hiring service. Upon completion of this service, the customer and the service provider must evaluate each other in the application, although there is no link between the parties and the application. For its development, some tools and platforms that are currently on the market will be used, such as Ionic Framework, Angular and Firebase. The developed methodology was based on the applied one, because it allows the application of all knowledge acquired during the undergraduate course of Information System - Univás to set up a product.

**Palavras-chaves:** Service providers. Application for service providers. Service evaluation. Service provider registration.

**LISTA DE CÓDIGOS**

Código 1 – Configuração do Firebase ............................................................................ 36

Código 2 – Importando o *plugin* do Firebase .................................................................. 36

Código 3 – Criação do menu ........................................................................................... 38

Código 4 – Tela de Login ............................................................................................... 39

Código 5 – Método de Login .......................................................................................... 40

Código 6 – Autenticação no Firebase ............................................................................. 40

Código 7 – Tela de cadastro ........................................................................................... 41

Código 8 – Criação de nova conta ................................................................................... 42

Código 9 – Salvando os dados usuário ............................................................................ 42

Código 10 – Buscando perfis .......................................................................................... 43

**L****ISTA DE FIGURAS**

Figura 1 – Diagrama de casos de uso .............................................................................. 26

Figura 2 – Arquitetura do Projeto .................................................................................... 27

Figura 3 – Site Node.Js ................................................................................................... 28

Figura 4 – Configuração do Android Studio ................................................................... 30

Figura 5 – Configuração do Android Studio ................................................................... 31

Figura 6 – Aplicativo Ionic Básico ..................................................................................32

Figura 7 – Página inicial do aplicativo ............................................................................ 33

Figura 8 – Estrutura de diretórios .................................................................................... 35

Figura 9 – Página inicial do Firebase .............................................................................. 36

Figura 10 – Modelagem do banco de dados .................................................................... 37

Figura 11 – Tela de cadastro ........................................................................................... 44

Figura 12 – Tela de perfil ................................................................................................ 45

Figura 13 – Menu do aplicativo ...................................................................................... 46

Figura 14 – Tela de busca avançada ................................................................................ 47

Figura 15 – Tela de resultados da busca ......................................................................... 48

Figura 16 – Tela de solicitação de serviço ...................................................................... 49

Figura 17 – Tela de gerenciamento de serviços .............................................................. 50

Figura 18 – Tela de avaliação .......................................................................................... 51

**LISTA DE ABREVIATURAS E SIGLAS**

HTML *Hypertext Markup Language*

XML *Extensible Markup Language*

CSS *Cascading Style Sheets*

WWW *World Wide Web*

NPM *Node Package Maneger*

LTS *Long Term Support*

JDK *Java Development Kit*

SDK *Software Development Kit*

IDE *Integrated Development Environment*

APK *Android Application Package*

AVD *Android Virtual Device*

**SUMÁRIO**

[1 INTRODUÇÃO 14](#_Toc22038482)

[2 QUADRO TEÓRICO 17](#_Toc22038483)

[2.1 HTML 17](#_Toc22038484)

[2.2 CSS 18](#_Toc22038485)

[2.3 Node.Js 19](#_Toc22038486)

[2.4 TypeScript 19](#_Toc22038487)

[2.5 Angular 20](#_Toc22038488)

[2.6 Ionic 21](#_Toc22038489)

[2.7 Firebase 21](#_Toc22038490)

[3 QUADRO METODOLÓGICO 22](#_Toc22038491)

[3.1 Tipo de pesquisa 23](#_Toc22038492)

[3.2 Contexto da pesquisa 23](#_Toc22038493)

[3.3 Instrumentos 24](#_Toc22038494)

[3.4 Procedimentos 25](#_Toc22038495)

[3.4.1 Requisitos do sistema 25](#_Toc22038496)

[3.4.2 Diagrama de Casos de Uso 26](#_Toc22038497)

[3.4.3 Tecnologias utilizadas 27](#_Toc22038498)

[3.4.4 Configuração do Ambiente 29](#_Toc22038499)

[3.4.4.1 Instalação do Node.js 29](#_Toc22038500)

[3.4.4.2 Instalação do Ionic 30](#_Toc22038501)

[3.5 Desenvolvimento do sistema 34](#_Toc22038502)

[3.5.1 Criação do projeto 34](#_Toc22038503)

[3.5.2 Estrutura de diretórios 35](#_Toc22038504)

[3.5.3 Configuração do Firebase 36](#_Toc22038505)

[*3.5.3.1 Instalação do Firebase* 37](#_Toc22038506)

[3.5.4 Modelagem do banco de dados 38](#_Toc22038507)

[3.5.5 Desenvolvimento do aplicativo 39](#_Toc22038508)

[3.6 Testes 60](#_Toc22038509)

[3.7 Publicação do aplicativo 60](#_Toc22038510)

[4 RESULTADOS OBTIDOS 61](#_Toc22038511)

[4.1 Proporcionar cadastro de usuários 61](#_Toc22038512)

[4.2 Realizando buscas 65](#_Toc22038513)

[4.3 Solicitando um serviço 67](#_Toc22038514)

[4.4 Avaliando 69](#_Toc22038515)

[5 CONCLUSÃO 70](#_Toc22038516)

[REFERÊNCIAS 72](#_Toc22038517)

# 1 INTRODUÇÃO

Observa-se que para muitas pessoas, a propaganda de boca em boca é o melhor método para se encontrar e contratar uma prestação de serviço. Entretanto, com a mudança do contexto social, que não permite às pessoas uma disponibilidade maior de tempo para se interagir, a necessidade de se encontrar um profissional especializado, tornou-se um desafio. Em muitas ocasiões, as pessoas não conseguem localizar o profissional desejado, buscando em vários locais, como na internet.

Com as múltiplas opções que a internet oferece, observaram-se que algumas pessoas se manifestam, principalmente em redes sociais, solicitando a indicação de um profissional para desempenhar determinados serviços que exigem especialização, como mecânicos, encanador, pintor, pedreiros, médicos e outros. Entretanto, o *feedback* muitas vezes pode vir de fontes não confiáveis ou demorar para chegar ao interessado ou, quando encontrado, na maioria das vezes, não apresentam as qualificações necessárias e requeridas ou até mesmo boas recomendações.

Segundo Daniel Queiroz (2019), o número de pessoas que trabalham por conta própria (autônomos) ou em vagas sem carteira assinada superou os que têm um emprego formal em 2017, e esse número só aumenta, de acordo com os dados divulgados pelo Instituto Brasileiro de Geografia e Estatística (IBGE, 2017).

Ainda em 2019, o desemprego já subiu para 12,7% e atinge a marca de 13,4 milhões de brasileiros. É a maior taxa desde maio de 2018 e a saída para os brasileiros é tentar a sorte por conta própria. Dessa forma, o avanço do trabalho sem registro formal mostra o crescimento da informalidade na economia. O chamado “por conta própria” é uma categoria que abrange diversos profissionais autônomos.

Em 2017 a economia foi se recuperando e também o número de desempregados teve uma redução. Em março, o Brasil atingiu a marca de 14,176 milhões de desempregados e, em dezembro, este número caiu para 12,3 milhões segundo o IBGE, 2017. Nessa mesma época, o País já tinha cerca de 1,67 milhões de pessoas a mais em empregos formais e informais.

Em 2018, o número de pessoas trabalhando com carteira assinada foi de 33,32 milhões. Já o ápice do emprego formal foi em 2014 com 36,6 milhões de trabalhadores no regime CLT

Uma saída para estes profissionais que querem atuar por conta própria são os aplicativos de serviços, que hoje em dia existem diversos. Estes aplicativos se tornaram o maior empregador de profissionais autônomos, com cerca de 4 milhões de trabalhadores. Além de estes aplicativos representarem as mudanças na oferta de serviços, eles têm acompanhado mudanças significativas nas relações do trabalho. Para um autônomo o ganho gerado por estes aplicativos acaba se tornando a principal fonte de renda.

Este projeto tem como objetivo principal a criação de um aplicativo por meio do qual os profissionais autônomos poderão se cadastrar e divulgar seus serviços. Por esta razão, o objetivo do aplicativo TIPS é reunir, em um único lugar, os mais diversos profissionais das mais diversas áreas, facilitando assim serem encontrados, sendo os objetivos específicos:

1. Configurar um serviço de banco de dados online para prover contas e perfis.
2. Criar um ambiente virtual no qual os usuários possam encontrar o profissional com facilidade.
3. Gerenciar os perfis dos profissionais com identificação, habilidades e contato.

Este trabalho está dividido em 5 capítulos, os quais se apresentam a seguir. No segundo capitulo, Quadro Teórico, são discutidos os dados técnicos e as tecnologias que foram utilizadas, entre os quais estão: HTML5, ANGULAR, IONIC e CSS3. No terceiro, Quadro Metodológico, é descrito todo o procedimento de desenvolvimento até o resultado final. No quarto, Resultados obtidos, são discriminados e abordados todo o resultado obtido durante o desenvolvimento. E, por fim, a Conclusão, onde é apresentado o olhar dos pesquisadores com relação a criação do aplicativo e a sua utilização.

Este trabalho tem por finalidade contribuir com a sociedade que, por meio deste sistema, os usuários poderão escolher, baseados nos perfis cadastrados, os serviços ofertados e, com a ajuda de avaliações de outros consumidores do aplicativo, verificar se o perfil do profissional atenderá a sua necessidade.

Trata-se, portanto, de uma ferramenta que será alimentada pelos usuários, por meio das avaliações dos serviços prestados de cada profissional cadastrado e que fora recrutado. A aplicação trará, de um banco de dados online, os resultados da busca de acordo com os filtros aplicados pelo usuário. Isso resultará em benefícios reduzindo o tempo de procura e satisfação do usuário.

A abrangência deste aplicativo envolve, além de exibir um perfil do profissional com avaliações referentes aos trabalhos prestados a outros usuários do aplicativo, os dados de contato e localização que serão fornecidos pelo dono do perfil.

Esta proposta se pauta no desafio de melhorar a oferta de algumas plataformas já existentes, as quais não estão sendo bem aceitas pelo público, devido as suas limitações de áreas, software e acesso pago. Sendo um aplicativo de acesso fácil, rápido e gratuito. Além disso, justifica-se também por proporcionar aos autores do projeto um contato direto com novas tecnologias não estudadas na graduação e requeridas no mercado atual.

# 2 QUADRO TEÓRICO

Neste capítulo estão descritos os conceitos e tecnologias que serão utilizados no desenvolvimento do aplicativo proposto.

## 2.1 HTML

O HTML é a sigla em inglês que significa *Hyper Text Markup Language* e denomina-se, na língua portuguesa, Linguagem para Marcação de Hipertexto. Foi criado em 1991, pelo físico britânico, Tim Berners-Lee. Esta linguagem de marcação permite ao desenvolvedor utilizar marcações específicas para que o *agente de usuário*[[1]](#footnote-1) processe e apresente o conteúdo.

No início, o HTML tinha a finalidade de interligar as instituições de pesquisa próximas, e compartilhar documentos com facilidade. Após a criação do *World Wide Web* (WWW), rede de alcance mundial, em 1992, o HTML se tornou popular, pois ele é o responsável por estruturar todo o conteúdo de uma página. Desde a sua criação, ele passou por oito versões, na qual a última e mais avançada, o HTML5 (SILVA, 2015, p 19-25).

Neste projeto, o HTML5 é responsável por toda a estrutura de conteúdo Web criado pela plataforma Angular.

## 2.2 CSS

Segundo Silva (2011, p. 24-25), “CSS é uma abreviação de um termo em inglês *(Cascading Style Sheet)* que foi traduzido para o português como Folhas de Estilos em Cascata. Sua finalidade é devolver ao HTML/XML seu propósito inicial”.

O HTML foi criado para ser apenas a linguagem de estruturação de conteúdo. Isso significa que não cabe a ele fornecer ao agente de usuário como os elementos serão apresentados. Por exemplo: cores, tamanhos de fontes, textos, posicionamento e todo aspecto visual. Cabe ao CSS todo este trabalho de estilização de uma página *Web*, e esta é a sua função.

Como sua finalidade está voltada para a parte visual, a definição de uma regra[[2]](#footnote-2) de CSS se pauta em:

· Seletor: elementos de marcação do HTML onde será́ aplicada a regra CSS.

·  Declaração: parâmetros de estilização, propriedades e valores.

·  Propriedade: características do seletor.

·  Valor: quantificações ou qualificações do seletor a ser estilizado.

Para este projeto, foi utilizada a versão mais recente da linguagem, CSS3, a fim de desenvolver toda a interface visual do aplicativo, de maneira simples e intuitiva.

## 2.3 Node.Js

No final de 2009, Ryan Dahl criou o Node.Js com a ajuda de 14 programadores. Esta tecnologia possui um modelo inovador, pois sua arquitetura é totalmente *non-blocking-thread*[[3]](#footnote-3). Comparando esta ferramenta com as demais de mercado, percebe-se que ela tem funcionalidades extras. Por exemplo, caso uma aplicação trabalhe com processamentos de arquivos e/ou realiza muitas operações de entrada e saída de informações, adotar esta arquitetura vai resultar em uma boa performance, além de trabalhar apenas em *single-thread*[[4]](#footnote-4).

O Node.Js é uma plataforma altamente escalável e de baixo nível, pois permite programar diretamente com diversos protocolos de rede e internet e também utilizar bibliotecas que acessam recursos do sistema operacional, principalmente os de sistemas baseados em Unix (PEREIRA, 2014).

O Node.Js foi usado neste projeto para configuração, instalação e gerenciamento das dependências do Ionic e alguns *plug-ins*, como o Firebase, por meio do NPM, *Node Package Maneger*, que em português significa Gerenciador de Pacotes do Node.

## 2.4 TypeScript

O *TypeScript* é um superset[[5]](#footnote-5) de *JavaScript*[[6]](#footnote-6), desenvolvido pela Microsoft em 2012. Hoje é mundialmente conhecido pelo seu poder e facilidade de desenvolvimento em larga escala.

Possui a mesma semântica e sintaxe do *JavaScript*, porém adicionando novas funções e possibilidades como, por exemplo, tipagem[[7]](#footnote-7) de variáveis e interfaces. Isso possibilita ao desenvolvedor codificar de maneira mais simples, organizada e de fácil manutenção.

De acordo com seu site, o *TypeScript* é compilado para um código JavaScript limpo e simples que é executado em qualquer navegador, no *Node.js* ou em qualquer mecanismo JavaScript que suporte o *ECMAScript 3*[[8]](#footnote-8).

O *TypeScript* é usado pelo Angular e ficou responsável por toda a parte lógica e funcional, ou seja, a criação dos métodos e rotinas do aplicativo.

## 2.5 Angular

Segundo o site do Angular, ele é uma plataforma que facilita a criação de aplicativos Web e Mobile. O Angular combina modelos declarativos, injeção de dependência, ferramentas de ponta a ponta e práticas recomendadas integradas para resolver desafios de desenvolvimento. O Angular capacita os desenvolvedores a criar aplicativos que estejam na Web, em dispositivos móveis ou para Desktops.

O Angular surgiu em meados de 2014, sendo uma evolução do AngularJS[[9]](#footnote-9), com novas funcionalidades e propriedades. A plataforma que é *open-source[[10]](#footnote-10),* é mantida por uma equipe de desenvolvimento do Google e por uma extensa comunidade de desenvolvedores ao redor do mundo, mantendo a mesma atualizada, estável e com melhorias constantes. Usando como base o *TypeScript*, este framework foi usado para a criação de um conteúdo Web, do qual foi responsável por toda parte visual e lógica do aplicativo (ANGULAR, 2019).

Ele é um dos itens principais do Ionic e sua finalidade é construir todo conteúdo Web que será exibido para o usuário. Ele utiliza o *TypeScript* para as partes lógicas, o HTML e o CSS para estruturar o conteúdo e a interface da maneira mais agradável e fácil para o usuário.

## 2.6 Ionic

O *Ionic* é uma plataforma de desenvolvimento *mobile* *open-source*, criado em 2012. Tem o intuito de facilitar para os desenvolvedores Web, a criação de aplicativos multiplataforma, ou seja, aplicações que podem ser executadas em sistemas operacionais distintos, como o Android[[11]](#footnote-11)e o iOS[[12]](#footnote-12), na qual com apenas um código, é possível criar aplicativos para dispositivos movíeis, Web e Desktop.

Aplicativos desenvolvidos com o *Ionic* consistem em um navegador web, sem barra de endereços ou configurações, com a finalidade de reproduzir um conteúdo Web específico, como um aplicativo *mobile*.

Neste projeto, utilizou-se a versão 3.2.0 do Ionic para desenvolver o aplicativo, que é a versão mais usada no momento. Neste projeto, o desenvolvimento foi feito apenas para o Sistema Operacional Android (IONIC, 2019).

## 2.7 Firebase

Segundo Smyth (2017, p. 1), o Google concluiu a aquisição de uma empresa sediada em San Francisco chamada Firebase. Esta fornece uma variedade de soluções para desenvolvedores, projetadas para acelerar a integração de recursos baseados em nuvem e aplicativos móveis.

Após a compra da empresa, o Google combinou os serviços fornecidos pelo Firebase com diversos recursos complementares anteriormente incluídos como parte do *Google Cloud Platform*[[13]](#footnote-13)*.* Por teradicionado novas funcionalidades, tornou o Firebase uma ferramenta poderosa e cheia de recursos práticos para o desenvolvedor.

Para este projeto, foram usadas algumas das ferramentas do seu plano gratuito que tem um número de requisições limitadas, como o sistema de autenticação, banco de dados em tempo real, armazenamento de arquivos e *cloud functions*[[14]](#footnote-14), itens necessários para o funcionamento do aplicativo.

# 3 QUADRO METODOLÓGICO

Neste capítulo, o objetivo é apresentar o tipo de pesquisa, contexto e os procedimentos realizados no desenvolvimento do projeto.

## 3.1 Tipo de pesquisa

Segundo Gil (2007, p 17), pesquisa é um procedimento racional e sistemático com o objetivo de proporcionar respostas para problemas propostos. A pesquisa se desenvolve por inúmeras fases, desde a adequada formulação do problema até a satisfatória apresentação e discussão de seus resultados.

Para este projeto, foi usado o método de pesquisa aplicada, que possui o objetivo de resolver problemas específicos. Segundo Thiollent (2009, p 36, apud FLEURY et al, 2016, p.11), “a pesquisa aplicada concentra-se em torno dos problemas presentes nas atividades das instituições, organizações, grupos ou atores sociais. Ela está empenhada na elaboração de diagnósticos, identificação de problemas e busca de soluções.”

## 3.2 Contexto da pesquisa

Este trabalho foi desenvolvido devido à dificuldade que algumas pessoas têm de encontrar determinados profissionais, tencionando à criação de um aplicativo por meio do qual autônomos e empresas poderão se cadastrar e divulgar seus serviços. Muitas pessoas têm dificuldades em localizar um profissional desejado. Mesmo buscando em redes sociais, familiares e amigos, o profissional encontrado nem sempre é o melhor indicado e, na maioria dos casos, sabe-se pouco sobre ele.

O objetivo desta aplicação é reunir, em um único lugar, os mais diversos profissionais das mais diversas áreas, facilitando assim serem encontrados. Por meio deste sistema, os usuários poderão escolher profissionais, baseando em suas informações de perfil e avaliações dos demais usuários.

Para os profissionais, o aplicativo se torna um novo meio de divulgação de serviços, além de verificar o perfil do seu possível cliente, por meio de avaliações de outros profissionais.

## 3.3 Instrumentos

O instrumento utilizado foram as reuniões. Ele foi escolhido por facilitar a organização e desenvolvimento de todo o projeto.

No começo, foram abordadas várias reuniões entre os desenvolvedores para discutir, sobre o problema a ser solucionado com este projeto, possíveis ideias para sua solução, requisitos do sistema e tecnologias a serem usadas durante todo o desenvolvimento.

Foram necessárias reuniões sobre o projeto com o orientador para um melhor acompanhamento e supervisão de todo o desenvolvimento.

## 3.4 Procedimentos

A seguir estão todos os procedimentos usados no desenvolvimento do projeto:

* Definição dos requisitos do Sistema.
* Criação do diagrama de Caso de Uso.
* Definição das tecnologias utilizadas.
* Configuração de ambiente.
* Desenvolvimento do aplicativo.
* Realização dos testes.
* Publicação do aplicativo.

Estes procedimentos estão descritos a seguir dando uma visão do desenvolvimento do projeto.

### 3.4.1 Requisitos do sistema

Para o desenvolvimento deste projeto, foram levantados alguns requisitos necessários para seu funcionamento. Durante as primeiras reuniões, foram evidenciados os seguintes requisitos.

* Cadastro de profissionais autônomos e usuários.
* Busca por profissionais.
* Visualização de profissionais.
* Avaliação de profissionais e usuários.
* Acesso gratuito.

Foi realizada uma pesquisa por aplicativos parecidos no mercado e poucas opções foram encontradas. Algumas delas não possuem a mesma finalidade do projeto, as demais possuem recursos limitados nas versões gratuitas e uma avaliação de nota média dos usuários.

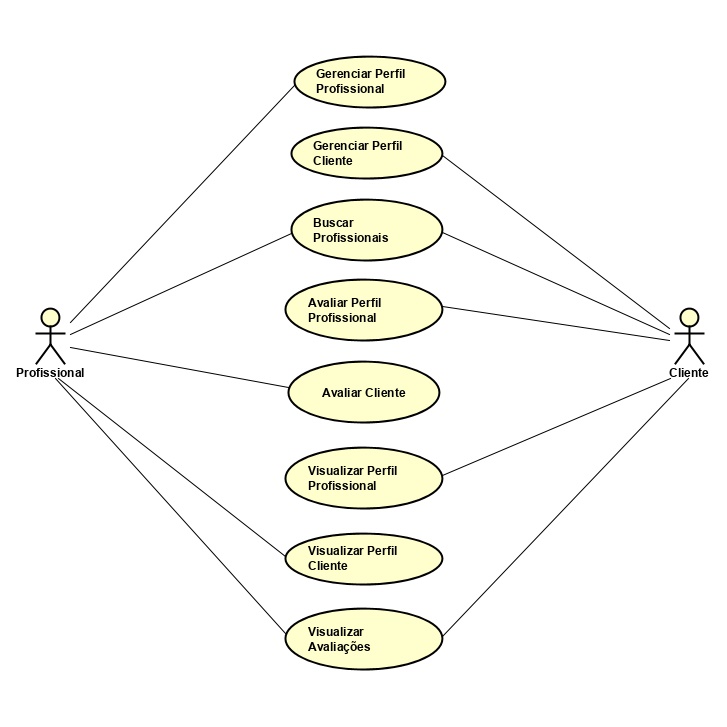
### 3.4.2 Diagrama de Casos de Uso

O diagrama de casos de uso possibilita a compreensão de comportamento de um sistema por qualquer pessoa que tenha o conhecimento sobre o problema em questão. Ele tem, por objetivo, apresentar uma visão externa geral das funcionalidades que o sistema oferecerá, sem se preocupar com a profundidade e sua implementação (GUEDES, 2018).

O diagrama é de grande auxílio na identificação e compreensão dos requisitos funcionais ofertados pelo sistema, ajudando a especificar, visualizar e documentar suas funções (GUEDES, 2018).

Na Figura 1 estão representadas as funcionalidades básicas do aplicativo através de um diagrama de casos de uso.

**Figura 1** – Diagrama de casos de uso



**Fonte:** Elaborado pelos autores (2019)

Na Figura 1, verificam-se as ações de cada tipo de usuário no sistema. O Profissional pode gerenciar seu perfil com informações de seu serviço, visualizar suas avaliações, visualizar o perfil de um cliente e avaliá-lo ou então realizar buscas por outros profissionais.

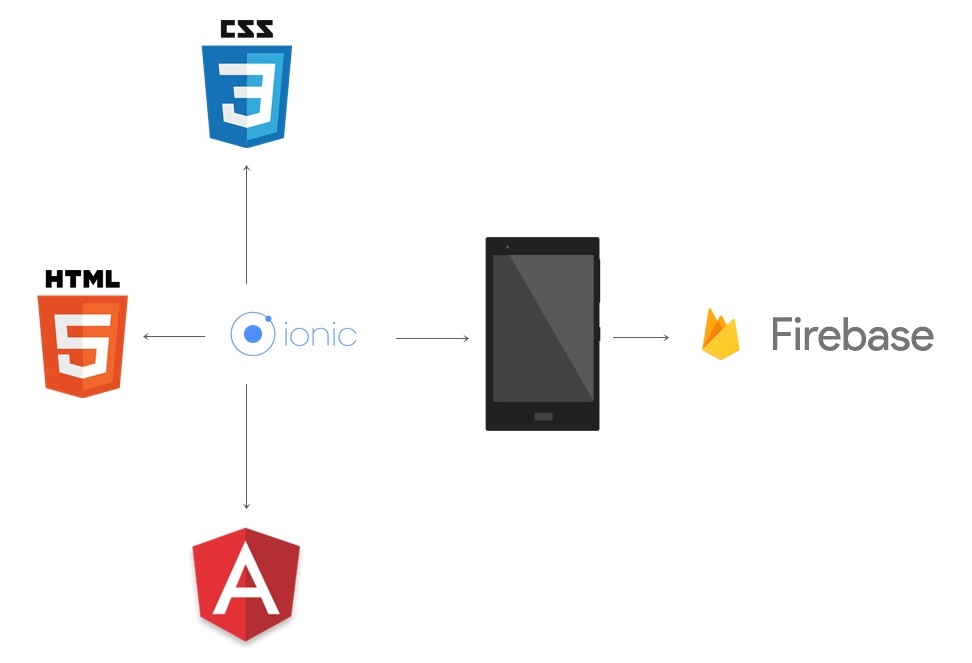
Assim também o cliente pode gerenciar seu perfil e verificar avaliações recebidas além de buscar profissionais, visualizá-los e avaliá-los.

### 3.4.3 Tecnologias utilizadas

Foi realizada uma pesquisa para escolher as melhores e mais práticas tecnologias para desenvolvimento do aplicativo. Selecionamos tecnologias que estão em alta no mercado, com funcionalidades que facilitam o desenvolvimento e proporcionam uma nova experiência aos desenvolvedores.

Na Figura 2 a seguir, é demonstrada a arquitetura do projeto.

**Figura 2** – Arquitetura do Projeto



**Fonte:** Elaborado pelos autores (2019)

Decidiu-se que o aplicativo seria desenvolvido em Ionic, que utiliza tecnologias muito usadas no mercado, como HTML 5, CSS 3, TypeScript e Angular. Devido ao fato de o Ionic proporcionar o desenvolvimento de um aplicativo para múltiplas plataformas, torna fácil para os desenvolvedores expandirem o aplicativo para outras plataformas, como o iOS. O Angular foi o responsável por estruturar todo o código com o HTML 5 e o CSS 3, para gerenciar de maneira mais intuitiva a interface do aplicativo.

O *Firebase*, por sua vez, é um *BaaS*[[15]](#footnote-15) que será responsável por fornecer e gerenciar o banco de dados, autenticação de usuários, armazenamento de arquivos, como as fotos dos perfis, entre outras funcionalidades. Tudo isso sem escrever uma linha de código. Com ele é fácil fazer um cadastro de contas, controlar acessos, permissões entre outras.

### 3.4.4 Configuração do Ambiente

Para realizar o desenvolvimento do projeto, foi necessário instalar alguns plug-ins e ferramentas.

#### 3.4.4.1 Instalação do Node.js

No Node.js existe uma ferramenta a qual será muito utilizada na preparação do ambiente, o NPM. Para instalar o *Node.js*, basta acessar o seu site https://nodejs.org/en/ e fazer o download da versão LTS, que é recomendada para maioria dos usuários por ser a mais estável e realizar a instalação.

**Figura 3:** Site Node.Js



**Fonte:** Node.Js (2019)

Após sua instalação, a ferramenta NPM estará disponível. Sua funcionalidade é gerenciar os pacotes do node. Ela acessa um repositório online com vários *plug-ins* prontos para serem instalados. Para instalar algum *plug-in*, é necessário usar o comando *npm install* mais o nome do pacote do *plug-in* no terminal do sistema (NodeBR, 2016).

#### 3.4.4.2 Instalação do Ionic

Para instalar o Ionic, deve-se executar dois comandos no terminal do sistema, a saber:

* **npm install –g cordova**: Este comando é necessário para instalar um *plug-in* essencial para o Ionic, no qual com ele temos acesso a muitas funcionalidades do sistema e do hardware no qual o aplicativo será instalado.
* **npm install –g ionic**: Comando responsável pela instalação do Ionic. Após sua instalação é possível executar comandos para a criação ou execução de uma aplicação em Ionic.

Após a execução desses comandos, é possível criar um aplicativo em Ionic simples, para isso são necessários os seguintes comandos:

* **ionic start tips-mobile:** ele é responsável por criar a estrutura de um aplicativo em Ionic. Quando finalizado, será criada uma pasta com todos os arquivos necessários para executar o aplicativo.
* **ionic serve:** a finalidade deste comando é executar o aplicativo. Após sua execução será possível visualizar o aplicativo por meio do seu navegador, usando o endereço [*http://localhost:8100/*](http://localhost:8100/).

Após a criação de um projeto em *Ionic*, para executá-lo em um dispositivo, é necessária a instalação de mais dois programas, são eles o Java JDK e o Android Studio. O Java JDK, (*Java Development Kit)* é necessário para instalação do Android Studio que, por sua vez, é uma IDE (*Integrated Development Environment*), um ambiente que reúne ferramentas e configurações para desenvolvimento de aplicações, neste caso para Android.

O Android Studio é usado para configurar um dispositivo virtual, que é necessário para executar o aplicativo em sistema Android sem usar um smartphone. Para isso, é necessário instalar o Android Studio, que pode ser baixado no site [*https://developer.android.com/studio*](https://developer.android.com/studio).

Concluída a instalação, seleciona-se a opção SDK Manager[[16]](#footnote-16). Na sessão de nome *Android SDK*[[17]](#footnote-17), o desenvolvedor deve selecionar uma das versões listadas.

Na Figura 4 estão listadas as plataformas a serem baixadas e instaladas, para que o aplicativo possa ser criado e configurado corretamente. Não é necessária a instalação de todas as plataformas, uma vez que o aplicativo exige apenas a versão da plataforma no qual será executado durante o seu desenvolvimento.

**Figura 4** – Configuração do Android Studio



**Fonte:** Elaborado pelos autores (2019)

Feito isso, deve-se selecionar a aba SDK Tools[[18]](#footnote-18) e instalar as opções selecionadas na Figura 5. Esta é uma lista de *plug-ins* que são necessários para execução e desenvolvimento de qualquer aplicativo para Android.

Na Figura 5, é mostrado a tela de configuração, download e instalação das ferramentas do Android Studio. Neste cenário, recomendamos que sejam instalados todos os itens selecionados na figura, para obter um melhor funcionamento de todo o ambiente de desenvolvimento.

**Figura 5** – Configuração do Android Studio



**Fonte:** Elaborado pelos autores (2019)

O Android Studio é uma ferramenta de desenvolvimento muito poderosa e possui recursos que auxiliam muito o desenvolvedor, reduzindo tempo e melhorando a qualidade de toda aplicação feita com ele, mas tem seu preço. A instalação de plataformas desnecessárias pode ocupar um espaço em disco considerável, além de exigir um bom processamento computacional.

Após a instalação da plataforma e ferramentas necessárias para o Android Studio, a opção AVD Manager[[19]](#footnote-19) deve ser clicada e depois e criar um dispositivo virtual. Após a criação, basta executar o comando no terminal do sistema, a partir do diretório do seu projeto Ionic: ***ionic cordova run android***. Feito isso, o Ionic instalará alguns pacotes e *plug-in*s e criará um APK[[20]](#footnote-20) do aplicativo. Em seguida realizará a instalação no dispositivo virtual configurado no Android Studio.

Caso o desenvolvedor opte por executar em um smartphone Android, basta conectá-lo no computador, ativar o modo desenvolvedor e executar o comando acima. O resultado será igual à Figura 6.

**Figura 6**- Aplicativo Ionic Básico.



**Fonte:** Elaborado pelos autores (2019)

Em relação a Figura 6, e mostrada a tela inicial de uma aplicação básica Ionic.

## 3.5 Desenvolvimento do sistema

Durante o desenvolvimento do sistema, foi necessário criar a interface de todas as telas, a lógica para gerenciar todo o conteúdo a ser exibido e salvo, estrutura de todo o banco de dados, tipos de autenticações de usuários, regras e algoritmos para busca de profissionais. Além de fazer várias baterias de testes durante todo o processo.

### 3.5.1 Criação do projeto

Feita a configuração de todo o ambiente de desenvolvimento, acessa-se o terminal e dá-se início à criação do projeto, inserindo o seguinte comando:

**ionic start tips-mobile**

Após se iniciar o projeto, entra-se no diretório, para poder subir o servidor e visualizar a aplicação com o seguinte comando:

**cd tips**

Uma vez dentro do diretório, inicia-se o servidor com o seguinte comando:

**ionic serve**

Dado o comando, consegue-se o seguinte resultado, conforme Figura 7.

**Figura 7** – Página inicial

****

**Fonte:** Elaborado pelos autores (2019)

No exemplo da Figura 7 é possível acessar a aplicação pelo navegador com o seguinte endereço [*http://localhost:8100/*](http://localhost:8100/).

### 3.5.2 Estrutura de diretórios

O Angular possui uma estrutura de diretórios muito completa e bem organizada. A pasta que mais utilizamos, durante o desenvolvimento, foi *src/app*; são nestas pastas que ficam armazenados todos os arquivos do projeto, conforme Figura 8.

**Figura 8** – Estruturas de diretórios



**Fonte:** Elaborado pelos autores (2019)

A figura 8 demonstra a estrutura de diretórios empregada na aplicação.

Além dos diretórios acimas apresentados, ainda foram utilizados mais 4 na composição deste projeto, a saber:

1) *e2e*:diretório responsável por conter os arquivos de teste.

2) *node\_modules*: este diretório e responsável por todas as bibliotecas quando as adicionamos ao projeto, e também gerenciar os pacotes e suas versões.

3) *assets*:diretório responsável pelos recursos externos ao trabalho, como imagens e ícones, entre outros.

4) *environments*: diretório responsável pela configuração dos ambientes de desenvolvimento e produção.

### 3.5.3 Configuração do Firebase

A configuração do *Firebase* é simples. Basta acessar o site *https://firebase.google.com*, clicar no botão *Get Started* e seguir o tutorial. Para criar projetos neste serviço de banco de dados, é necessária a criação de uma conta Google. Caso o desenvolvedor tenha uma conta no Gmail, poderá utilizá-la sem problemas.

Após a criação de um projeto dento do *Firebase*, foi necessária a instalação de seu *plug-in* para *Ionic* e a configuração dos dados de acesso ao projeto pelo *plug-in*.

#### *3.5.3.1 Instalação do Firebase*

Para instalar o Firebase é necessário também realizar a instalação do AngularFire, um modulo que irá nos prover uma serie de facilidades para trabalhar com o Firebase. E para tal deve-se executar o seguinte comando: ***npm install angularfire2 firebase --save***

Feito isso, agora se adiciona a configuração do *firebase* ao projeto. Para isso, deve-se abrir o *src/app/app.modules.ts* e adicionar as seguintes linhas de código, como mostra o Código 1.

**Código 1**: Configuração do firebase

1 **const** config = {

2 apiKey: "YOUR\_API\_KEY",

3 authDomain: "YOUR\_AUTHENTICATION\_DOMAIN",

4 databaseURL: "YOUR\_DATABASE\_URL",

5 projectId: "YOUR\_PROJECT\_ID",

6 storageBucket: "YOUR\_STORAGE\_BUCKET\_DOMAIN",

7 messagingSenderId: "YOUR\_MESSAGE\_SENDER\_ID"

8 };

**Fonte:** Elaborado pelos autores (2019)

Prosseguindo com a configuração, injeta-se os provedores do Firebase e especifica-se sua configuração. Para isso, foi necessário abrir o arquivo ***src/app/app.mudules.ts*** e adicionar a seguinte configuração, conforme Código 2:

**Código 2:** Importando os plugins do Firebase

1 imports: [

2 BrowserModule,

3 ComponentsModule,

4 IonicModule.forRoot(MyApp),

5 IonicStorageModule.forRoot({

6 name: '\_userData',

7 driverOrder: ['indexeddb']

8 }),

9 AngularFireModule.initializeApp(config),

10 AngularFirestoreModule,

11 AngularFireAuthModule,

17 ],

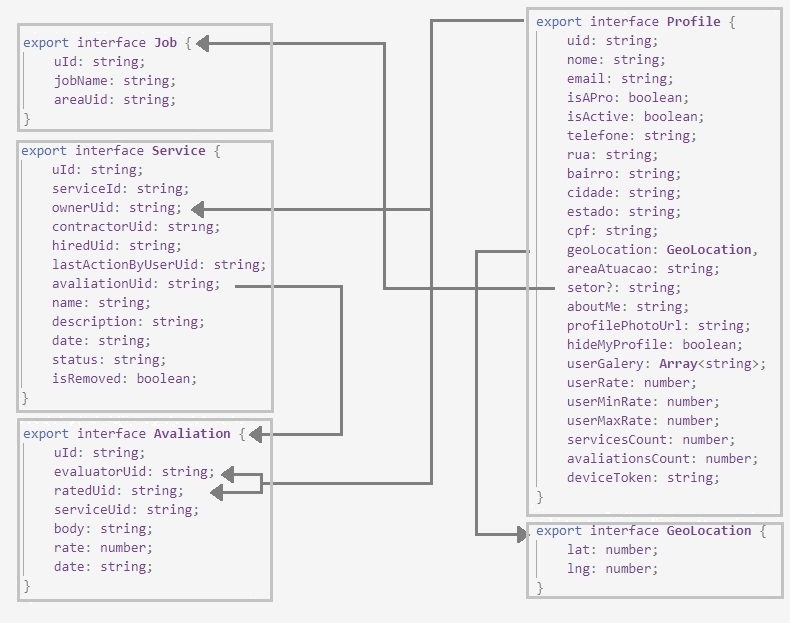
**Fonte:** Elaborado pelos autores (2019)

O Código 2 mostra as importações dos provedores necessários para o correto funcionamento do Firebase.

### 3.5.4 Modelagem do banco de dados

Para o desenvolvimento do trabalho, fez-se necessário a criação de um modelo de banco de dados, que possa explicar as características de funcionamento e comportamento de um sistema, facilitando seu entendimento para evitar assim erros de programação futuramente. Conforme mostra Figura 10.

**Figura 10**: Modelagem do banco de dados.



**Fonte:** Elaborado pelos autores (2019)

Na Figura 10 é apresentada a modelagem do banco de dados utilizada pelos desenvolvedores no trabalho.

### 3.5.5 Desenvolvimento do aplicativo

O aplicativo foi desenvolvido com o *framework Ionic*. Após a criação do projeto, começa-se a desenvolver o componente de menu, conforme mostra Código 3.

**Código 3:** Criação do menu

1 **<div** class="menu"**>**

2 **<div** class="menuHeader"**>**

3 **<div** class="profilePhoto” ...

4 **</div>**

5 **<div** class="profileData"**>**

6 **<h4>**{{profile.name.firstName}}**</h4>**

7 **<p>**{{profile.email}}**</p>**

8 **</div>**

9 **</div>**

10 **<div** class="menuProfileOptions"**>**

11 **<button** icon-star (click)= "editProfile()"**>**

12 **<ion-icon** name="person"**></ion-icon>**

13 Conta

14 **</button>**

15 **<button** icon-start (click)= "rating()"**>**

16 **<ion-icon** name="star"**></ion-icon>**

17 Avaliações

18 **</button>**

19 **<button** icon-start (click)="services()"**>**

20 **<ion-icon** name="briefcase"**></ion-icon>**

21 Solicitações de Serviços

22 **</button>**

23 **</div>**

24 **<div** class="menuAppOptions"**>**

25 **<button** icon-start (click)="configs()"**>**

26 **<ion-icon** name="settings"**></ion-icon>**

27 Configurações

28 **</button>**

29 **<button** icon-start**>**

30 **<ion-icon** name="information-circle"**></ion-icon>**

31 Sobre

32 **</button>**

33 **<button** icon-start (click)="logout()"**>**

34 **<ion-icon** name="exit"**></ion-icon>**

35 Sair

36 **</button>**

37 **</div>**

38 **</div>**

**Fonte:** Elaborado pelos autores (2019)

No Código 3, cada *tag*[[21]](#footnote-21) *button* [[22]](#footnote-22) representa um botão do menu. A *tag* *button* nos permite colocar um ícone que representa sua funcionalidade; isso para deixar o menu mais intuitivo.

Nas linhas 6 e 7 no Código 3 é mostrado o nome e e-mail do usuário logado. Para cadastro e utilização do aplicativo, foi desenvolvida uma tela para *login* e uma para cadastro, conforme mostra Código 4.

**Código 4:** Tela de login

1 **<ion-content** padding**>**

2 **<div** class="card login"**>**

3 **<div** class="cardTitle"**>**

4 **<img** src="../../../assets/imgs/logo.png" **/>**

5 **<h2>**Tips**</h2>**

6 **</div>**

7 **<div** class="cardBody"**>**

8 **<form** #form="ngForm" (submit)="login(form)"**>**

9 **<ion-item**

10 **<ion-label** floating**>**E-mail**</ion-label>**

11 **<ion-input** type="email" name="email"

12 ngModel maxLength="60"**>**

13 **</ion-input>**

14 **</ion-item>**

15 **<ion-item>**

16 **<ion-label** floating**>**Senha**</ion-label>**

17 **<ion-input** type="password" name="password"

18 ngModel maxLength="10"**>**

19 **</ion-input>**

20 **</ion-item>**

21 **<button** ion-button**>**Login**</button>**

22 **</form>**

23 **</div>**

24 **<div** class="cardFooter"**>**

25 **<h5** (click)="newAccount()"**>**Nova Conta**</h5>**

26 **<h5** (click)="forgotPassword()"**>**Esqueci minha senha**</h5>**

27 **</div>**

28 **</div>**

29 **<div** class="devBy"**>**

30 **<p>**Develop by Tips **<sup>**®**</sup></p>**

31 **</div>**

32 **</ion-content>**

**Fonte:** Elaborado pelos autores (2019)

No Código 4 está representada uma tela de *login,* onde tem-se um formulário para preenchimento do e-mail e senha para efetuar a autenticação do usuário. Após o usuário clicar sobre o botão de login, os dados digitados são capturados e acionado o método de login, como mostra o Código 5.

**Código 5:** Método de login

1 login(form: NgForm): **void** {

2 **if** (**this**.validateAccount(form)) {

3 **this**.loading.showLoading('Entrando em sua conta...')

4 .then(async () => {

5 **this**.afAuth.login(form)

6 .then(async (result) => {

7 **this**.successLogin(result);

8 })

9 .**catch**((error) => {

10 console.log('Erro ao fazer login: ', error);

11 **this**.errorLogin();

12 });

13 })

14 }

15 }

**Fonte:** Elaborado pelos autores (2019)

O Código 5 é responsável em acionar o método *validateAccount*, linha 2, que tem por finalidade a validação dos campos. Após esse procedimento, é acionado o método de login, linha 5. A linha 3 é responsável em acionar o loading do sistema, que exibe uma mensagem para o usuário enquanto o processo de acesso ao sistema e executado.

No Código 6 é demonstrado o procedimento de *login,* que éacionado na linha 5 pelo Código 5.

**Código 6:** Autenticação no Firebase.

1 login(form: NgForm): Promise<any> {

2 **return** firebase.auth()

3 .signInWithEmailAndPassword(form.value.email,form.value.password)

4 }

**Fonte:** Elaborado pelos autores (2019)

De acordo com o código 6, na linha 1, o método recebe os dados necessários para realizar o acesso ao sistema, usuário e senha, por parâmetro; e na linha 2 em seguida, aciona-se o método de *login* com e-mail e senha do *firebase*, *signInWithEmailAndPassword,* informando os valores recebidos anteriormente. Após o termino da execução deste método, o *successLogin,* na linha 7 no Código 5, é acionado, redirecionando o usuário para a tela inicial. Caso ocorra algum erro, o método *errorLogin* será acionado exibindo um erro ao usuário.

Para se inscrever no aplicativo, deve ser desenvolvida uma tela para cadastro, em que se tem um formulário para preenchimento de algumas informações. Após clicar sobre o botão de criar nova conta, os dados são capturados e acionado o método *newAccount*. Conforme Código 7.

**Código 7:** Tela de cadastro

1 **<ion-row>**

2 **<ion-col** col-6**>**

3 **<ion-item>**

4 **<ion-label** stacked**>**Nome:**</ion-label>**

5 **<ion-input** type="text"

6 name="firstName" ngModel maxLength="30"**>**

7 **</ion-input>**

8 **</ion-item>**

9 **</ion-col>**

10 **<ion-col** col-6**>**

11 **<ion-item>**

12 **<ion-label** stacked**>**Sobrenome:**</ion-label>**

13 **<ion-input** type="text"

14 name="lastName" ngModel maxLength="30"**>**

15 **</ion-input>**

16 **</ion-item>**

17 **</ion-col>**

18 **</ion-row>**

19 **<ion-item>**

20 **<ion-label** stacked**>**E-mail**</ion-label>**

21 **<ion-input** type="email"

22 name="email" ngModel maxLength="60"**>**

23 **</ion-input>**

24 **</ion-item>**

25 **<ion-item>**

26 **<ion-label** stacked**>**Nova Senha:**</ion-label>**

27 **<ion-input** type="password"

28 name="password" ngModel maxLength="10"**>**

29 **</ion-input>**

30 **</ion-item>**

31 **<ion-item>**

32 **<ion-label** stacked**>**Confirmar Senha:**</ion-label>**

33 **<ion-input** type="password"

34 name="confirmPass" ngModel maxLength="10"**>**

35 **</ion-input>**

36 **</ion-item>**

37 **<ion-item>**

38 **<ion-label>**Sou um profissional?**</ion-label>**

39 **<ion-toggle** name="isAPro"

40 [(ngModel)]="isAPro"

41 (ionChange)="alertInformation()">

42 **</ion-toggle>**

43 **</ion-item>**

**Fonte:** Elaborado pelos autores (2019)

O Código 8, na linha 2 é responsável por acionar o método de validação dos campos, *validadteAccount.* Após esse procedimento, na linha 5, é acionado o método *createNewAccount* informando os dados de *login* do novo usuário. Na linha 9, deve ser criado um objeto de novo usuário com os demais dados e acionado o método *saveUser,* conforme Código 9.

**Código 8:** Criação de nova conta

1 newAccount(form: NgForm): **void** {

2 **if** (**this**.validateAccount(form)) {

3 **this**.loading.showLoading('Estamos criando a sua conta...')

4 .then(() => {

5 **this**.authProvider.createNewAccount(

6 form.value.email,

7 form.value.password)

8 .then((result) => {

9 **let** newUser = {

10 uid: result.user.uid,

11 firstName: form.value.firstName,

12 lastName: form.value.lastName,

13 email: form.value.email,

14 isAPro: form.value.isAPro,

15 accountType: 'APPLICATION'

16 }

17 **return** **this**.saveUser(newUser);

18 })

19 .**catch**((e) => {

20 **this**.loading.hideLoading();

21 console.log(e);

22 **this**.alert.simpleAlert('Opps!',

23 'Houve um erro ao criar conta!');

24 });

25 })

26 }

27 }

**Fonte:** Elaborado pelos autores (2019)

O Código 9 é responsável em criar um novo perfil de usuário com as configurações básicas e autenticação.

**Código 9:** Salvando os dados do usuário

1 **private** async saveUser(newUser: any) {

2 **return** **this**.userProvider.saveNewUser(newUser)

3 .then(async () => {

4 **return** **this**.userProvider.saveUserAuth(newUser.uid)

5 .then(() => {

6 **this**.setProfileConfigurations();

7 })

8 })

9 }

**Fonte:** Elaborado pelos autores (2019)

Após salvar as informações básicas e de autenticação nas linhas 2 e 4, o usuário é redirecionado para a tela de configuração de conta, para preencher com seus demais dados.

O Código 10 é responsável por realizar a busca dos perfis profissionais no banco de dados *firebase* de acordo com os filtros aplicados pelo usuário.

**Código 10:** Buscando perfis.

1 async getProfiles(filter: FilterOptions, limit: number) {

2 console.log("ProfileProvider | Get Profiles by search!");

3 **return** **this**.db.collection(Constants.PROFILES\_COLLECTION, ref => {

4 **let** query: firebase.firestore.CollectionReference |

5 firebase.firestore.Query = ref;

6 **if** (filter.profileName != "")

7 { query = query.where('nome', '==', filter.profileName) };

8 **if** (filter.profileState != "")

9 { query = query.where('estado', '==', filter.profileState) };

10 **if** (filter.profileCity != "")

11 { query = query.where('cidade', '==', filter.profileCity) };

12 **if** (filter.profileSector != "")

13 { query = query.where('setor', '==', filter.profileSector) };

14 **if** (filter.profileRate != 0)

15 { query = query.where('userRate', '==', filter.profileRate) }

16 **else** { query = query.orderBy('userRate', 'desc') };

17 **return** query.where('hideMyProfile', '==', **false**)

18 .where('isActive', '==', **true**)

19 .where('isAPro', '==', **true**)

20 .limit(limit)

21 }).valueChanges()

22 }

**Fonte:** Elaborado pelos autores (2019)

A partir da linha 3, do Código 10, é criada a query[[23]](#footnote-23) para um dos tipos de filtro selecionado pelo usuário, que são armazenados no objeto filter e então requisitar ao banco por profissionais específicos.

O Código 11 é responsável por criar a tela onde é exibo o resultado da busca, com todos os perfis que foram encontrados no Firebase.

**Código 11:** Tela de resultado da busca.

1 **<div** class="card profile"

2 \*ngFor="let profile of profiles"

3 (click)="goToDetails(profile)"**>**

4 **<div** class="profileTitle"**>**

5 **<div** class="profilePhotoResult"

6 [ngStyle]= "{'background-image':

7 'url(' + profile.profilePhotoUrl + ')'}"**>**

8 **<img** \*ngIf="profile.profilePhotoUrl == ''"

9 src="../../../assets/imgs/149071.png" **/>**

10 **</div>**

11 **<div** class="profileName"**>**

12 **<h3>**{{profile.name.firstName + "

13 " + profile.name.lastName}}**</h3>**

14 **</div>**

15 **<div** class="profileRating rating"

16 \*ngFor="let star of starsRate(profile.userRate);

17 let i of index"**>**

18 **<ion-icon** name="{{star}}"

19 [ngClass]= "starsRateColor(profile.userRate) "

20 \*ngIf="i != 5"**></ion-icon>**

21 **</div>**

22 **</div>**

23 **<div** class="clearfix"**></div>**

24 **<div** class="cardBody"**>**

25 **<p** \*ngIf="profile.job"**>**{{profile.job}}**</p>**

26 **</div>**

27 **</div>**

**Fonte:** Elaborado pelos autores (2019)

Após analisar os perfis que foram encontrados, o usuário pode então realizar a solicitação de serviço, acessando a tela de solicitação de serviço.

O Código 12 é responsável por exibir a tela onde o usuário poderá realizar a solicitação de serviço.

**Código 12:** Tela de solicitação de serviço

1 **<div** class="card service" \*ngIf="!solicitationDone"**>**

2 **<div** class="cardBody" \*ngIf="!enableDescription"**>**

3 **<p>**Deseja enviar uma solicitação de serviço para

4 **<br/>**

5 **<b>**{{ hiredPf.name.firstName }}**</b>**?

6 **</p>**

7 **<button** ion-button (click)="setServiceDescription()"

8 \*ngIf="!enableDescription"**>**

9 Confirmar**</button>**

10 **</div>**

11 **<div** class="serviceDesciption" \*ngIf="enableDescription"**>**

12 **<div** class="cardBody"**>**

13 **<p>**

14 Descreva abaixo um pouco sobre o serviço que você precisa:

15 **</p>**

16 **<ion-item>**

17 **<ion-label** floating**>**Descrição do serviço:**</ion-label>**

18 **<ion-textarea** [(ngModel)]="solicitationDescription"

19 type="text" name="solicitationDescription"

20 maxLength="600" rows="8"**>**

21 **</ion-textarea>**

22 **</ion-item>**

23 **<button** ion-button (click)="makeService()"**>**Enviar**</button>**

24 **</div>**

25 **</div>**

26 **</div>**

**Fonte:** Elaborado pelos autores (2019)

Dentro da tela de solicitação de serviço, na linha 5, e exibido o nome do profissional que receberá a solicitação, na linha 18 temos a *tag* *ion-textarea* que nos permite criar um espaço onde o solicitante pode inserir algumas informações referentes ao serviço que ele precisa, na linha 23 temos um botão chamado enviar, assim que o mesmo for pressionado, o método *makeService* e acionado.

O Código 13 mostra o método *makeService* que dá início ao evento de solicitação de serviço. A linha 2 aciona o método *formValidation* que é encarregado de verificar se a solicitação está preenchia corretamente, caso o preenchimento não atenda aos requisitos, é exibido uma mensagem ao usuário solicitando que faça o preenchimento correto. Conforme Código 14.

**Código 13:** Inicia solicitação de serviço

1 makeService() {

2 **if** (**this**.formValidation()) {

3 **this**.enableDescription = **false**;

4 **this**.solicitationUser();

5 }

6 }

**Fonte:** Elaborado pelos autores (2019)

**Código 14:** Método *formValidation*

1 formValidation() {

2 **if** (!**this**.solicitationDescription) {

3 **this**.toast.showToast

4 ("Preencha a descrição desta solicitação!");

5 **return** **false**;

6 }

7 **return** **true**;

8 }

**Fonte:** Elaborado pelos autores (2019)

Após o preenchimento correto da solicitação, a linha 4 do método *makeService* fica encarregada de acionar o método *solicitationUser.*

O método *solicitationUser* é responsável em capturar todos os dados necessários e construir a solicitação de serviço, e por fim, acionar o método *saveSolicitation*, na linha 29. Conforme Código 15.

**Código 15:** Construindo solicitação de serviço

1 solicitationUser() {

2 **let** solicitation: Solicitation = {

3 uId: UUID.UUID(),

4 solicitationId: UUID.UUID(),

5 contractorUid: this.contractorPf.uid,

6 hiredUid: this.hiredPf.uid,

7 lastActionByUserUid: this.contractorPf.uid,

8 description: this.solicitationDescription,

9 observations: null,

10 date: parseInt(Date.now().toString()),

11 status: Constants.SOLICITATION\_IS\_OPEN,

12 name: "",

13 profileNames: {

14 contractorName:

15 this.contractorPf.name.firstName

16 + " " + **this**.contractorPf.name.lastName,

17 hiredName: this.hiredPf.name.firstName

18 + " " + **this**.hiredPf.name.lastName,

19 },

20 removedTo: {

21 contractorUid: null,

22 hiredUid: null,

23 },

24 avaliatedTo: {

25 contractorAvaliation: null,

26 hiredAvaliation: null,

27 }

28 }

29 **this**.saveSolicitation(solicitation);

30 }

**Fonte:** Elaborado pelos autores (2019)

O método *saveSolicitation* é responsável por acionar o a rotina denominada *createSolicitation,* na linha 4, e também informar ao usuário se houve algum erro ou não durante o envio da solicitação. Se obter êxito, a linha 12 exibe uma mensagem dizendo que a solicitação foi enviada com sucesso, caso contrário, a linha 18 exibirá uma mensagem de erro. Conforme Código 16.

**Código 16:** Método *saveSolicitation*

1 **private** saveSolicitation(solicitation: Solicitation) {

2 **this**.loading.showLoading("Solicitando serviço...")

3 .then(() => {

4 **this**.solicitationProvider.createSolicitation(solicitation)

5 .then(async () => {

6 **this**.solicitation = solicitation;

7 **this**.solicitationDate = **new** Date

8 (**this**.solicitation.date).toLocaleDateString();

9 **this**.loading.hideLoading();

10 **this**.solicitationDone = **true**;

11 **this**.toast.showToast

12 ("Solicitação enviada com sucesso!");

13 })

14 .**catch**((err) => {

15 console.log(err);

16 **this**.solicitationDone = **false**;

17 **this**.loading.hideLoading();

18 **this**.toast.showToast("Erro ao solicitar serviço!");

19 });

20 })

21 }

**Fonte:** Elaborado pelos autores (2019)

A rotina *createSolicitation* fica encarregada em salvar a solicitação no banco de dados, em uma *collection[[24]](#footnote-24)* chamada *solicitation\_collection.* Conforme Código 17.

**Código 17:** Método *saveSolicitation*

1 async createSolicitation(service: any): Promise<**void**> {

2 **return** await **this**.db.collection

3 (Constants.SOLICITATION\_COLLECTION)

4 .doc(service.uId)

5 .set(service)

6 }

**Fonte:** Elaborado pelos autores (2019)

O usuário tem acesso a uma tela onde será capaz de visualizar todas as suas solicitações de serviços de acordo com seu status, como mostra Código 18.

**Código 18:** Tela de solicitações de serviços

1 **<div** class="filter"**>**

2 **<ion-item>**

3 **<ion-select** [(ngModel)]="filterType"

4 (ionChange)="onFilterChange()"**>**

5 **<ion-option** value="ALL\_SOLICITATIONS"**>**

6 Todos os serviços**</ion-option>**

7 **<ion-option** value="SOLICITATIONS\_DONE"

8 \*ngIf="profile.isAPro"**>**Serviços feitos**</ion-option>**

9 **<ion-option** value="SOLICITATIONS\_RECEIVED"

10 \*ngIf="profile.isAPro"**>**Serviços recebidos**</ion-option>**

11 **<ion-option** value="SOLICITATION\_IS\_OPEN"**>**

12 Serviços abertos**</ion-option>**

13 **<ion-option** value="SOLICITATION\_IS\_RUNNING"**>**

14 Serviços em andamento**</ion-option>**

15 **<ion-option** value="SOLICITATION\_IS\_FINISHED"**>**

16 Serviços terminados**</ion-option>**

17 **<ion-option** value="SOLICITATION\_IS\_CANCELED"**>**

18 Serviços cancelados**</ion-option>**

19 **<ion-option** value="SOLICITATION\_IS\_REMOVED"**>**

20 Serviços removidos**</ion-option>**

21 **<ion-option** value="SOLICITATION\_IS\_AWAIT\_TO\_FINISH"**>**

22 Serviços para terminar**</ion-option>**

23 **<ion-option** value="SOLICITATION\_IS\_AWAIT\_TO\_CANCEL"**>**

24 Serviços para cancelar**</ion-option>**

25 **</ion-select>**

26 **</ion-item>**

27 **</div>**

**Fonte:** Elaborado pelos autores (2019)

Ainda na tela de gerenciamento de solicitações de serviço, o usuário tem a possibilidade buscar serviços escolhendo um dos status mencionados no Código 18, para acessar os detalhes da solicitação, caso o mesmo não possua nenhuma solicitação, ele verá uma mensagem informando que não possui serviços, como mostra Código 19.

**Código 19:** Solicitação de serviço

1 **<div** class="list"**>**

2 **<div** \*ngIf="solicitations.length > 0"**>**

3 **<div** class="card serviceItemList"

4 [ngClass]="setSolicitationStatusClass(service.status)"

5 \*ngFor="let service of solicitations; let i of index;"

6 [click)="goToDetails(service)"**>**

7 **<div** class="cardTitle"**>**

8 **<h3>**{{ service.name }}**</h3>**

9 **</div>**

10 **<div** class="cardBody"**>**

11 **<p>**{{ service.description }}**</p>**

12 **</div>**

13 **<div** class="cardFooter"**>**

14 **<h3>**{{ setStatusValueToShow(service.status) }}**</h3>**

15 **</div>**

16 **</div>**

17 **</div>**

18 **<div** class="notFound" \*ngIf="solicitations.length == 0"**>**

19 **<div** class="body"**>**

20 **<ion-icon** name="information-circle"**></ion-icon>**

21 **<div** \*ngIf="filterType == 'ALL\_SERVICES'"**>**

22 **<p>**Você não tem serviços!**</p>**

23 **</div>**

24 **<div** \*ngIf="filterType != 'ALL\_SERVICES'"**>**

25 **<p>**Não encontramos serviços para esse tipo de filtro!**</p>**

26 **</div>**

27 **</div>**

28 **</div>**

29 **</div>**

**Fonte:** Elaborado pelos autores (2019)

A linha 6 é responsável em acionar o método *goToDetails* passando o serviço selecionado anteriormente por parâmetro para que possa ser exibido os detalhes da solicitação.

O Código 20 mostra o método *goToDetails* o qual é responsável em direcionar o usuário para a tela de detalhes da solicitação.

**Código 20:** Método *goToDetails*

1 goToDetails(solicitation: any) {

2 **if** (solicitation.contractorUid == **this**.profile.uid) {

3 **this**.navCtrl.push('SolicitationDetailsPage',

4 { 'solicitation': solicitation })

5 } **else** {

6 **this**.navCtrl.push('SolicitationManagerPage',

7 { 'solicitation': solicitation })

8 }

9 }

**Fonte:** Elaborado pelos autores (2019)

O Código 21 fica encarregado de mostrar os detalhes da solicitação, onde é exibido o nome de quem requisitou, os dados de contato, uma descrição do serviço e a data em que a solicitação foi realizada.

**Código 21:** Detalhes da solicitação

1 **<div** class="serviceDescription"**>**

2 **<h5>**Serviço solicitado por:**</h5>**

3 **<p>**{{contractorPf.name.firstName}}

4 {{contractorPf.name.lastName}}**</p>**

5 **</div>**

6 **<div** class="serviceDescription"**>**

7 **<h5>**Formas de contato:**</h5>**

8 **<div** class="contact-field"**>**

9 **<div>**

10 **<ion-icon** name="call"**></ion-icon>**

11 **</div>**

12 **<span>**{{contractorPf.phone}}**</span>**

13 **</div>**

14 **<div** class="contact-field"**>**

15 **<div>**

16 **<ion-icon** name="mail"**></ion-icon>**

17 **</div>**

18 **<span>**{{contractorPf.email}}**</span>**

19 **</div>**

20 **<div** class="contact-field"**>**

21 **<div>**

22 **<ion-icon** name="pin"**></ion-icon>**

23 **</div>**

24 **<span>**{{contractorPf.street}}, {{contractorPf.houseNumber}} -

25 {{contractorPf.district}},

26 **<br** **/>**{{contractorPf.city}} - {{contractorPf.state}}**</span>**

27 **</div>**

28 **</div>**

29 **<div** class="serviceDescription"**>**

30 **<h5>**Descrição do serviço: **</h5>**

31 **<p>**{{solicitation.description}}**</p>**

32 **</div>**

33 **<div** class="serviceDescription observations"

34 \*ngIf="solicitation.observations"**>**

35 **<h5>**Observações: **</h5>**

36 **<div** \*ngFor="let obs of solicitation.observations"**>**

37 **<span** class="by"**>**Por {{ obs.userName }}**</span>**

38 **<p>**{{obs.body}}**</p>**

39 **<span** class="when"**>**Feito em: {{obs.date}}**</span>**

40 **</div>**

41 **</div>**

42 **<div** class="cardFooter solicitationDate"**>**

43 **<p>**Solicitado em: {{solicitationDate}}**</p>**

44 **</div>**

**Fonte:** Elaborado pelos autores (2019)

Após a finalização do serviço, o usuário tem acesso a tela avaliação, onde pode dar início a sua avaliação, selecionado uma nota de 1 a 5. Conforme Código 22.

**Código 22:** Selecionando nota

1 **<div** class="cardTitle" \*ngIf="asContractor"**>**

2 **<h2>**Avaliação de serviços!**</h2>**

3 **</div>**

4 **<div** class="cardTitle" \*ngIf="!asContractor"**>**

5 **<h2>**Avaliação de cliente!**</h2>**

6 **</div>**

7 **<div** class="cardBody"**>**

8 **<div** \*ngIf="asContractor"**>**

9 **<p>**Avalie o serviço em uma nota de 1 a 5!**</p>**

10 **</div>**

11 **<div** \*ngIf="!asContractor"**>**

12 **<p>**Avalie o cliente em uma nota de 1 a 5!**</p>**

13 **</div>**

14 **<div>**

15 **<div** class="rating"**>**

16 **<ionic3-star-rating** #rating activeIcon="ios-star"

17 defaultIcon="ios-star-outline"

18 activeColor="{{starActiveColor}}"

19 defaultColor="{{starActiveColor}}"

20 readonly="false"

21 rating="{{avaliationRate}}"

22 (ratingChanged)="ratingEvent($event)"**>**

23 **</ionic3-star-rating>**

24 **<p** class="notLike"**>**Não gostei!**</p>**

25 **<p** class="like"**>**Gostei!**</p>**

26 **</div>**

27 **<div** class="clearfix"**></div>**

28 **</div>**

29 **</div>**

**Fonte:** Elaborado pelos autores (2019)

Continuando a avaliação, o usuário tem acesso ao um campo de texto onde pode descrever sua opinião em relação ao serviço prestado ou no caso de o profissional estar avaliando, ele pode deixar sua opinião em relação ao cliente. Conforme Código 23.

**Código 23:** Escrevendo avaliação

1 **<div** class="cardBody"**>**

2 **<div** \*ngIf="asContractor"**>**

3 **<p>**Dê sua opinião sobre os serviços de

4 **<br** **/>**

5 **<b>**{{ hiredProfile.name.firstName }}

6 {{ hiredProfile.name.lastName }}**</b>** abaixo.

7 **</p>**

8 **</div>**

9 **<div** \*ngIf="!asContractor"**>**

10 **<p>**Dê sua opinião sobre seu cliente,

11 **<br** **/>**

12 **<b>**{{ contractorProfile.name.firstName }}

13 {{ contractorProfile.name.lastName }}.**</b>** abaixo.

14 **</p>**

15 **</div>**

16 **</div>**

17 **<div** class="cardFooter"**>**

18 **<ion-item>**

19 **<ion-textarea** [(ngModel)]="avaliationBody"

20 type="text" name="avaliationDescription"

21 placeholder="Ex: Gostei muito do serviço prestado.

22 Excelente profissional!"

23 maxLength="350"**>**

24 **</ion-textarea>**

25 **</ion-item>**

26 **</div>**

**Fonte:** Elaborado pelos autores (2019)

Ao termino da avaliação o usuário tem um botão confirmar, o qual, quando pressionado, aciona o método *finish*, na linha 5. Conforme Código 24.

**Código 24:** Enviando avaliação

1 **<ion-footer>**

2 **<ion-toolbar>**

3 **<ion-title>**

4 **<button** ion-button full

5 (click)="finish()"**>**Confirmar**</button>**

6 **</ion-title>**

7 **</ion-toolbar>**

8 **</ion-footer>**

**Fonte:** Elaborado pelos autores (2019)

Após finalizado todo o preenchimento das informações da avaliação, o método denominado *finish,* captura todos os dados informados referente a avaliação em questão, faz todo o tratamento necessário através do método *formValidation,* linha 2. Como mostra Código 25.

**Código 25:** Finalizando avaliação

1 async finish() {

2 **if** (**this**.formValidation()) {

3 **this**.avaliation.date = parseInt(Date.now().toString());

4 **this**.avaliation.rate = **this**.avaliationRate;

5 **this**.avaliation.body = **this**.avaliationBody;

6 await **this**.loading.showLoading("Salvando avaliação...")

7 .then(async () => {

8 **return** await **this**.avaliationProvider

9 .saveAvaliation(**this**.avaliation)

10 .then(async () => {

11 await **this**.updateSolicitation(**this**.avaliation.uId);

12 })

13 .**catch**(() => {

14 **this**.onError();

15 })

16 })

17 .**catch**(() => {

18 **this**.loading.showLoading("Erro ao Salvar avaliação...")

19 });

20 }

21 }

**Fonte:** Elaborado pelos autores (2019)

O Código 26 mostra o método *formValidation* que verifica se uma nota foi escolhida e a avaliação foi inserida, se os dados não forem preenchidos corretamente, uma mensagem será exibida para o usuário solicitando o correto preenchimento das informações, por outro lado, se tudo estiver correto a avaliação será salva e o serviço atualizado, como mostra as linhas 9 e 11 do Código 25.

**Código 26:** Validação da avaliação

1 formValidation() {

2 **if** (**this**.avaliationRate == 0) {

3 **this**.toast.showToast("Escolha uma nota de 1 a 5 estrelas!");

4 **return** **false**;

5 }

6 **if** (!**this**.avaliationBody) {

7 **if** (**this**.asContractor) {

8 **this**.toast.showToast

9 ("Escreva uma avaliação sobre este profissional!");

10 } **else** {

11 **this**.toast.showToast

12 ("Escreva uma avaliação sobre este cliente!");

13 }

14 **return** **false**;

15 }

16 **return** **true**;

17 }

18 }

**Fonte:** Elaborado pelos autores (2019)

Após finalizado todos os recessos mencionados interiormente, o procedimento de avaliação e finalizado.

## 3.6 Testes

Durante todo o desenvolvimento do aplicativo, teste foram realizados em todas as rotinas que estavam sendo desenvolvidas. E, após finalizar o processo de desenvolvimento, foram realizados testes em dispositivos reais afim de garantir que todos os recursos do aplicativo estivem funcionando corretamente.

## 3.7 Publicação do aplicativo

Depois de finalizada toda a parte de desenvolvimento e testes do aplicativo, foi necessário publicá-lo na *Play Store*, loja de aplicativos do Android, para que fique disponível para todos.

# 4 RESULTADOS OBTIDOS

Neste capítulo são discutidos os resultados obtidos durante o processo de desenvolvimento do projeto, cujo objetivo principal consiste na criação de um aplicativo para dar mais uma opção aos profissionais autônomos divulgarem seus serviços e conquistarem mais clientes. Para a sociedade, um novo meio de encontrar prestadores de serviços avaliados por outros usuários, de forma rápida e simples. Com esta pesquisa, é esperado também obter conhecimento nos métodos utilizados no desenvolvimento do projeto, tais como:

* Desenvolvimento de aplicativos com Ionic.
* Desenvolvimento de aplicações Web com Angular.
* Gerenciamento de um banco de dados com Firebase.
* Publicação de um aplicativo na *Play Store*.

## 4.1 Proporcionar cadastro de usuários

Para que os usuários tenham acesso ao sistema TIPS, antes de tudo eles devem realizar um cadastro tanto para realizarem buscas ou para disponibilizarem seus serviços. Conforme ilustra a Figura 11, que mostra a tela de cadastro, na qual os usuários devem fornecer os dados que são solicitados para efetuarem o registro, ao finalizar, os usuários serão redirecionados para outras telas onde serão solicitados os demais dados para configuração do perfil.

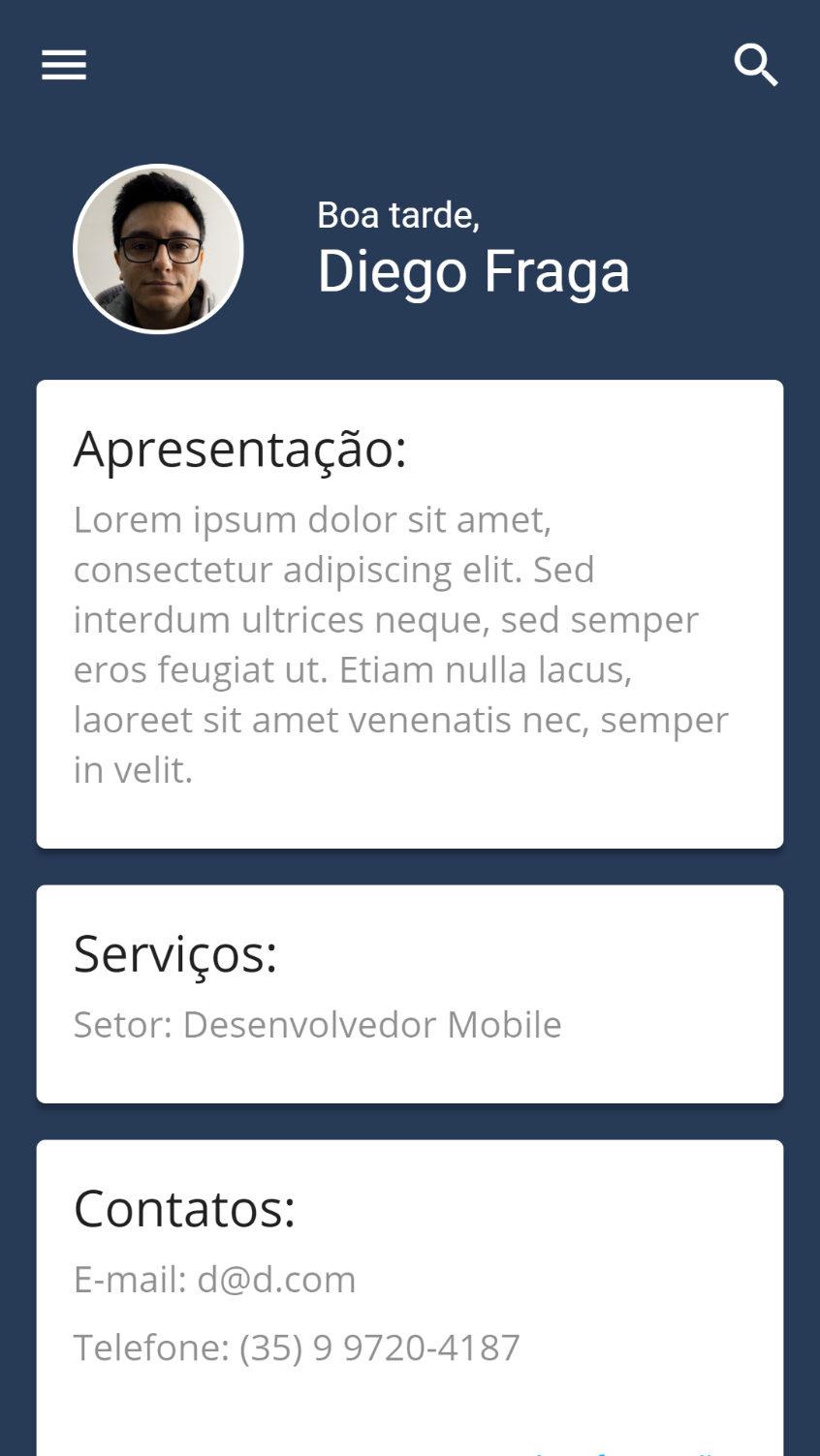
**Figura 11** – Tela de cadastro



**Fonte:** Elaborado pelos autores (2019)

No término da configuração de perfil o usuário deverá ver a tela inicial do aplicativo, conforme Figura 12.

**Figura 12** – Tela inicial

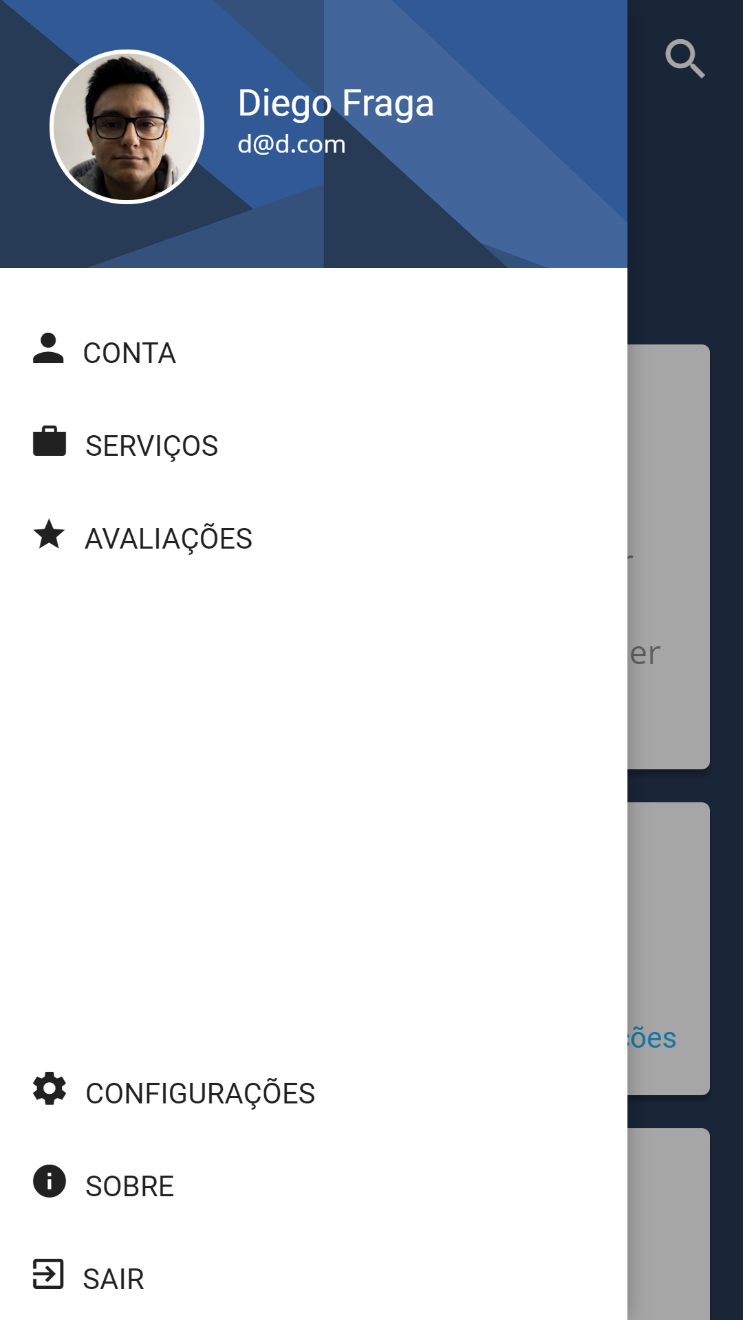


**Fonte:** Elaborado pelos autores (2019)

Uma vez na tela inicial, o usuário visualizará todas as suas informações que foram preenchidas anteriormente e terá acesso ao ícone de menu e ao de busca.

Acessando o menu, a partir da tela inicial, os usuários poderão ter acesso às configurações do aplicativo, conforme Figura 13.

**Figura 13** – Tela do menu do aplicativo



**Fonte:** Elaborado pelos autores (2019)

Dentro do menu, o usuário encontrará as seguintes opções: conta, local onde poderá realizar modificações dos dados do seu perfil, serviços, onde serão mostrados os serviços feitos e contratados e avaliações. Ele poderá visualizar suas avaliações, configurações, o usuário poderá alterar o tipo da sua conta, entre simples ou profissional, deixando seu perfil visível para ser encontrado, alterar a senha e excluir a conta, sobre, visualizará um breve resumo do aplicativo e por fim, sair do aplicativo.

## 4.2 Realizando buscas

Para um usuário realizar buscas, obrigatoriamente ele deve estar logado no sistema. Acessando a página de busca, o usuário deverá escolher os filtros de sua pesquisa para que seja localizado o profissional desejado, conforme ilustra Figura 14.

**Figura 14** – Tela de busca avançada



**Fonte:** Elaborado pelos autores (2019)

Na página de busca, o usuário tem a opção avançada, na qual deve fornecer alguns dados, como por exemplo, o nome do profissional, a profissão, estado e a cidade que ele deseja encontrar, ou apenas escolher que tipo de profissional deseja procurar, em uma busca mais simples, assim, ocultando as opções avançadas e o sistema se encarregara de trazer os perfis dos profissionais e exibira em uma lista, conforme Figura 15.

**Figura 15** – Tela de resultados da busca



**Fonte:** Elaborado pelos autores (2019)

## 4.3 Solicitando um serviço

Ao término da busca com a lista de perfis sendo exibida, o usuário deve selecionar um dos perfis, ver as informações do mesmo e se este atender as suas necessidades, enviar uma solicitação de serviço e entrar em contato através dos meios disponibilizados, conforme mostra Figura 16.

**Figura 16** – Tela de solicitando um serviço



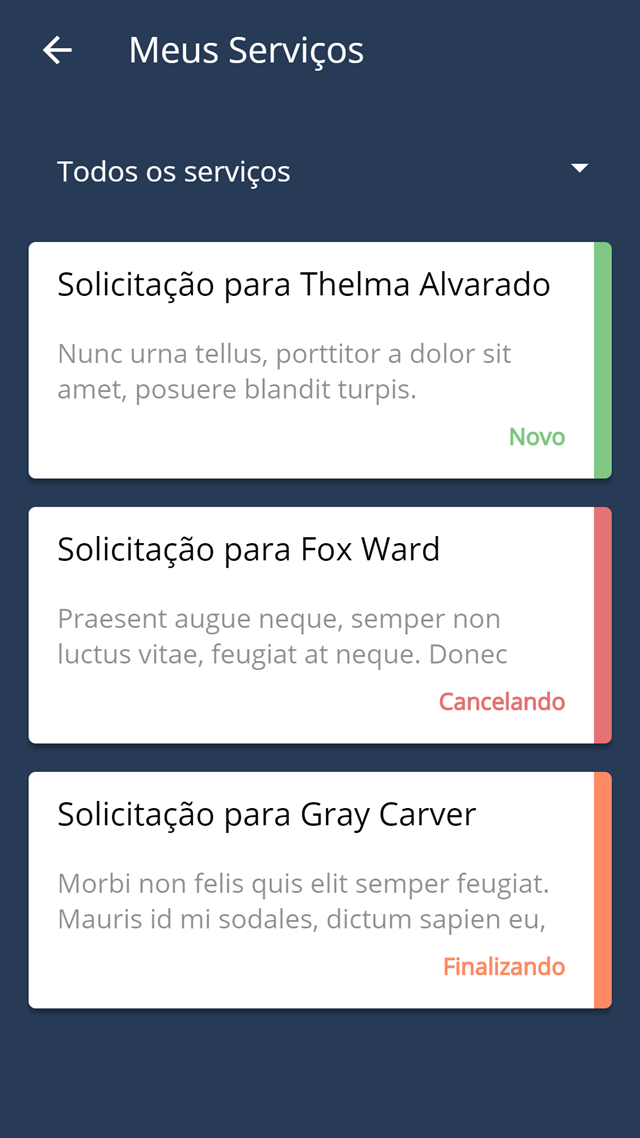
**Fonte:** Elaborado pelos autores (2019)

O proprietário do perfil será notificado com a solicitação do serviço, poderá analisar o perfil da pessoa que enviou a solicitação, e diante desta análise tomar a sua decisão, aceitar ou ignorar.

Assim que o profissional recebe a solicitação, o usuário já pode conferir na tela de gerenciamento de serviços.

A Figura ilustra 17 lista as solicitações de serviço e seus status, que são: Novo, Finalizado e Cancelado.

**Figura 17** – Tela de gerenciamento de serviços

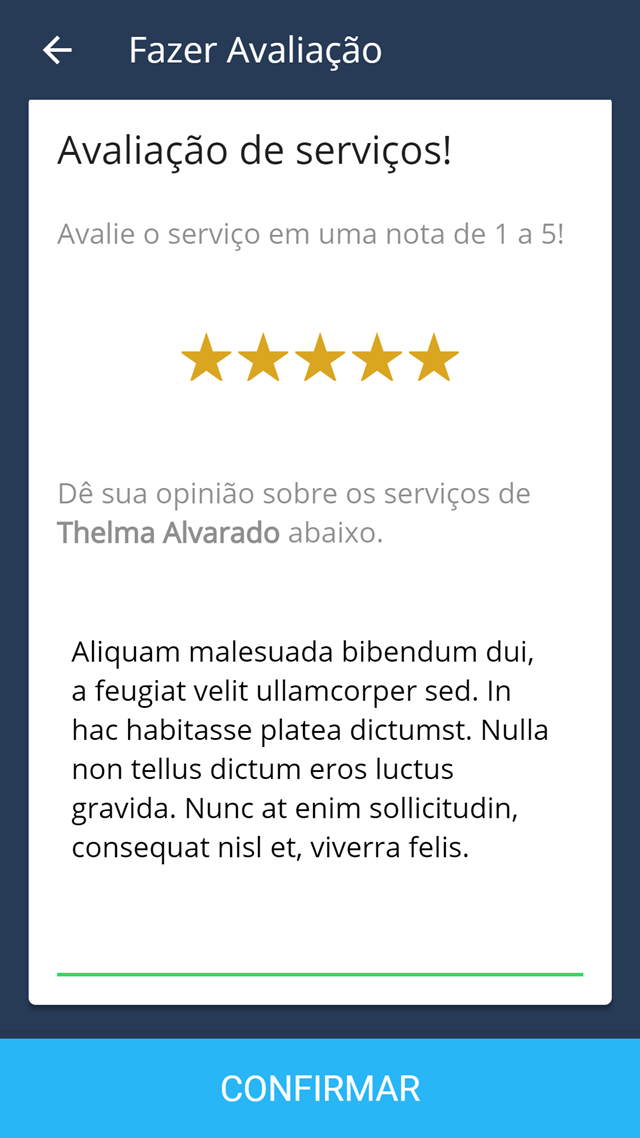


**Fonte:** Elaborado pelos autores (2019)

## 4.4 Avaliando

Quando o serviço estiver concluído ambas as partes, contratante e contratado, deve-se avaliar no aplicativo, conforme Figura 18, que mostra a tela de avaliação, por meio da qual as partes devem se avaliar, selecionado uma nota de 1 a 5 e um comentário. Estas avaliações são de extrema importância para os outros usuários, por que mediante elas, outros usuários do aplicativo poderão julgar se os perfis em questão lhes atendem ou não.

**Figura 18** – Tela de avaliação



**Fonte:** Elaborado pelos autores (2019)

# 5 CONCLUSÃO

Observando a dificuldade que as pessoas tem de localizar determinado profissional, vimos uma oportunidade da criação de um aplicativo, que por meio do qual, possibilita que profissionais autônomos se cadastrem, podendo criar um perfil, inserir seus dados de contato e informações sobre os serviços que prestam, de forma que os usuários, que realizarem um rápido cadastro na aplicação, conseguirão buscar, visualizar estes perfis e escolher o profissional mais adequado as suas necessidades.

Conclui-se que este trabalho auxiliará as pessoas na busca por diversos profissionais, das mais variadas áreas, a fim de suprir esta dificuldade que temos hoje em dia, além da falta de tempo para interagir com outras pessoas pedindo por recomendações, e em alguns casos, das más recomendações que algumas vezes recebemos. Este trabalho também contribuirá para a área de desenvolvimento *mobile,* servindo de referência para outros alunos que queiram utilizar de tecnologias que atualmente estão em alta no mercado. Concluímos também que este projeto serviu para fornecer mais conhecimento aos desenvolvedores, pois além de aplicado tudo que foi visto no curso ao longo destes anos, tivemos contato com tecnologias que não foram vistas na graduação, e que estão em alta no mercado.

O aplicativo foi projetado pensando na usabilidade dos usuários, de forma que mesmos não terão dificuldades em utilizar nenhuma das funcionalidades.

As ideias que foram pensadas e não foram adicionas na versão atual do aplicativo, serão aplicadas como melhorias futuras.

* Descrever ...

# 

# REFERÊNCIAS

ANGULAR. **Angular -** what is Angular**.** Disponível em: https://angular.io/docs Acesso em 17 mar. 2019.

QUEIROZ, Daniel. **Trabalho por conta própria já supera o trabalho por carteira assinada** – Disponível em: Acesso em 29 ago. 2019.

FIREBASE. **Firebase.** Disponível em: https://firebase.google.com/ Acesso em 12 abr. 2019.

FLEURY, Maria Tereza L.; WERLANG, Sergio R.C. **Pesquisa aplicada:** conceitos e abordagens. Disponível em: file:///E:/72796-150874-1-PB.pdf Acesso em 12 abr. 2019.

GIL, Antônio Carlos. **Como elaborar projetos de pesquisa.** 4. ed. São Paulo: Atlas S.A, 2002. Disponível em: http://www.urca.br/itec/images/pdfs/modulo%20v%20-%20como\_elaborar\_projeto\_de\_pesquisa\_-\_antonio\_carlos\_gil.pdf Acesso em 12 abr. 2019.

GUEDES, Gilleanes T. A. **UML 2:** uma abordagem prática. 3. ed. São Paulo: Novatec, 2018. Disponível em: https://books.google.com.br/books?id=RUdLDwAAQBAJ&printsec=frontcover&hl=pt-BR&source=gbs\_ViewAPI&output=embed&redir\_esc=y#%257B%257D Acesso em 12 abr. 2019.

IONIC FRAMEWORK. **About ionic cross-platform mobile development technologies.** Disponível em: https://ionicframework.com/about Acesso em 17 mar. 2019.

NODEBR. **O que é a NPM do Node.JS.** 2016. Disponível em: http://nodebr.com/o-que-e-a-npm-do-nodejs/ Acesso em 12 abr. 2019.

PACIEVITCH, Yuri. **HTML – informática.** Disponível em: https://www.infoescola.com/informatica/html/ Acesso em 17 mar. 2019.

PEREIRA, Caio R. **Aplicações web real-time com Node.Js.** 2014. Disponível em: https://books.google.com.br/books?id=Wm-CCwAAQBAJ&printsec=frontcover&dq=Aplica%C3%A7%C3%B5es+web+real-time+com+Node.js&hl=pt-BR&sa=X&ved=0ahUKEwi60ojU--vgAhVEK7kGHcUZBU0Q6wEIKjAA#v=onepage&q&f=false Acesso em 17 mar. 2019.

RABELO, Eduardo. **TypeScript:** o guia definitivo. 2018. Disponível em: https://medium.com/@oieduardorabelo/typescript-o-guia-definitivo-1a63b04259cc Acesso em 17 mar. 2019.

SILVA, Maurício S. **Desenvolva aplicações web profissionais com uso dos poderosos recursos de estilização das CSS3.** 2011. Disponível em: https://books.google.com.br/books?id=EEOZAwAAQBAJ&printsec=frontcover&hl=pt-BR&source=gbs\_ViewAPI&output=embed&redir\_esc=y#%257B%257D Acesso em 17 mar. 2019.

\_\_\_\_\_\_. **Fundamentos de HTML5 e CSS3.** 2015**.** Disponível em: https://books.google.com.br/books?id=2iPYCQAAQBAJ&printsec=frontcover&hl=pt-BR&source=gbs\_ViewAPI&output=embed&redir\_esc=y#%257B%257D Acesso em 21 mar. 2019.

SMYTH, Neil. **Firebase essentials.** Disponível em: https://www.ebookfrenzy.com/pdf\_previews/FirebaseEssentialsAndroidPreview.pdf 2017. Acesso em 17 mar. 2019.

TYPESCRIPT. **TypeScript -** JavaScript that scales. Disponível em: https://www.typescriptlang.org/index.html Acesso em 17 mar. 2019.

UNIVERSIDADE FEDERAL DO RIO GRANDE DO SUL. Secretaria de Educação a Distância. **Métodos de pesquisa.** 1. ed. 2009. Disponível em: http://www.ufrgs.br/cursopgdr/downloadsSerie/derad005.pdf Acesso em 12 abr. 2019.

1. Agente de usuário é todo e qualquer dispositivo capaz de exibir conteúdo Web. [↑](#footnote-ref-1)
2. Regra CSS é a unidade básica de uma folha de estilo. Ela significa a menor porção de código capaz de produzir um efeito de estilização. Ela é composta por duas partes: seletor e a declaração. [↑](#footnote-ref-2)
3. Uma nova requisição poderá ser processada sem ser bloqueada por outra. [↑](#footnote-ref-3)
4. Sistema que trabalha com apenas um processo em execução. [↑](#footnote-ref-4)
5. Superset, que em português, significa superconjunto. [↑](#footnote-ref-5)
6. Linguagem de programação de alto nível, muito usada na Web. [↑](#footnote-ref-6)
7. É o ato de definir o tipo de dado de uma variável. [↑](#footnote-ref-7)
8. Linguagem de programação baseada em scripts, padronizada pela empresa Ecma International. [↑](#footnote-ref-8)
9. Versão inicial do Angular. [↑](#footnote-ref-9)
10. Código aberto, para alteração, publicação de qualquer desenvolvedor. [↑](#footnote-ref-10)
11. Sistema Operacional mantido pela Google. [↑](#footnote-ref-11)
12. Sistema Operacional desenvolvido e mantido pela Apple. [↑](#footnote-ref-12)
13. Plataforma de serviços em nuvem para desenvolvedores. [↑](#footnote-ref-13)
14. Serviço para execução de códigos em nuvem. [↑](#footnote-ref-14)
15. Back-end como serviço [↑](#footnote-ref-15)
16. Gerenciador do Kit de Desenvolvimento de Software [↑](#footnote-ref-16)
17. Kit de Desenvolvimento Android [↑](#footnote-ref-17)
18. Ferramentas do SDK [↑](#footnote-ref-18)
19. Android Virtual Device é um dispositivo virtual Android. [↑](#footnote-ref-19)
20. Android Application Package é o pacote que contém uma aplicação Android. [↑](#footnote-ref-20)
21. *Tag* é uma estrutura de linguagem de marcação contendo instruções, tendo uma marca de início e outra de fim para que o navegador possa exibir uma página *HTML*. [↑](#footnote-ref-21)
22. Nome da *tag* que representa um componente visual do tipo botão. [↑](#footnote-ref-22)
23. É um Código que tem a função de buscar ou executar alguma ação em um banco de dados [↑](#footnote-ref-23)
24. Coleção de documentos dentro do banco de dados. [↑](#footnote-ref-24)