Aplicativo Para Gerenciamento de *Home-care*

Eduardo de Paula Xavier
Graduando em Eng. de Software – Uni-FACEF
edpxavier@gmail.com

Orientador: Prof. Me. Ely Fernando do Prado
Mestre em ciência da computação – Uni-FACEF
elyfprado@gmail.com

Resumo:
Com a democratização da tecnologia, aumentou-se exponencialmente o número de brasileiros que hoje podem se beneficiar das funcionalidades e da praticidade de um aparelho *smartphone.* Este projeto buscou analisar necessidades do dia-dia de profissionais de *home-care*, que precisam se locomover com certa frequência, para propor uma solução considerando que dispositivos móveis se adequam satisfatoriamente a essa rotina. Através de entrevistas com funcionários e empreendedores da área, foram levantadas diversas dificuldades que causam desorganização e retrabalho. O objetivo deste projeto foi desenvolver uma aplicação para promover uma melhor organização e gerenciamento das informações cotidianas desses funcionários, possibilitando melhor administração e padronização da emissão de guias de atendimento dos pacientes. Para o desenvolvimento desta aplicação foram utilizados aparelhos móveis e ferramentas de desenvolvimento de software tais como o *framework Flutter* e o Sistema de Gerenciamento de Banco de Dados do *Firebase,* um serviço em nuvem. Como resultado, foi desenvolvida uma aplicação com um *layout* intuitivo e fluido, que possibilita a coleta e o armazenamento de dados em nuvem bem como a exportação de resultados em arquivos PDF reproduzindo as guias de atendimento dos pacientes.

Palavras-chave: Aplicativo *mobile.* Firebase. Flutter. *Home-care*. Saúde.

Abstract:
With the democratization of technology, the number of Brazilians who today can benefit from the features and practicality of a smartphone device has increased exponentially. This project sought to analyze the day-to-day needs of home-care professionals, who need to move around with a certain frequency, to propose a solution considering that mobile devices satisfactorily adapt to this routine. Through interviews with employees and entrepreneurs in the area, several difficulties that cause disorganization and rework were raised. The objective of this project was to develop an application to promote better organization and management of the daily information of these employees, enabling better administration and standardization of the issuance of patient care guides. For the development of this application, mobile devices and software development tools such as the Flutter framework and the Firebase Database Management System, a cloud service, were used. As a result, an application was developed with an intuitive and fluid layout, which enables the collection and storage of data in the cloud, as well as the export of results in PDF files reproducing the patient care guides.

Keywords: *Firebase. Flutter. Health. Home-care. Mobile application.*

1 INTRODUÇÃO

Um tipo de empresa muito comum na área da saúde são aquelas que atendem seus pacientes de forma domiciliar, conhecidas como *Home-care*. Neste tipo de empresa os profissionais de saúde conseguem cuidar de seus pacientes de uma maneira mais próxima, eficiente e principalmente com atenção àqueles que têm dificuldades de locomoção. Porém, muitas empresas de *home-care* relatam dificuldades na padronização e emissão de Recibos de Atendimento obrigatórios pelo SUS, principalmente por parte de seus funcionários, que muitas vezes não seguem as orientações de forma adequada gerando problemas na hora de emitir um comprovante válido.

Este projeto busca responder a questões sobre como as técnicas e metodologias de Engenharia de Software, podem apoiar o desenvolvimento de sistemas digitais para profissionais de *home-care* registrarem os atendimentos, para a geração de recibos de modo que se tenha um padrão válido definido pelo SUS.

A ideia do trabalho se deu após detectar um problema em uma empresa de fonoaudiologia, em que as funcionárias têm que tirar fotos do recibo escrito de próprio punho e enviar para a proprietária em formato PDF. Porém, muitas vezes, essas funcionárias encaminhavam uma única foto contendo todos os recibos, acarretando em transtornos na hora de identificar as informações, sendo necessárias conversões e edições no arquivo PDF dos recibos das consultas, gerando assim um serviço desnecessário. Tendo isto em vista, este trabalho é motivado por encontrar uma solução computacional para minimizar o devido problema.

O objetivo do projeto é desenvolver um aplicativo de interface simples e intuitiva que gere os recibos de forma padronizada, no formato PDF, e que possa ser facilmente enviado aos devidos responsáveis ou impressos, além de implementar outros recursos como um sistema de geolocalização por GPS, uma vez que o aplicativo é focado para pessoas que realizam o atendimento na casa de seus clientes, o que facilita a localização de endereços.

O projeto utilizou métodos de pesquisa exploratória, em que foram realizados estudos teóricos e práticos para validar a solução proposta no contexto estudado.

Foram utilizadas técnicas aprendidas durante o curso de Engenharia de Software, como Gestão de Projetos, Arquitetura de Software, Interface Humano-Computador (IHC) e Qualidade de Software, além da linguagem de programação Dart utilizada no *framework* Flutter.

Foi desenvolvida também, a estrutura do banco de dados para armazenamento em nuvem, onde os dados ficarão sempre disponíveis na aplicação desde que se tenha acesso à internet.

O motivo do projeto ser produzido em Flutter se deve ao fato de ser um *framework* multiplataforma, permitindo assim, que seus usuários possam acessar a aplicação em diferentes dispositivos, sendo eles Android ou IOS.

2 REFERENCIAL TEÓRICO

Nesta seção são apresentadas as tecnologias e as referências que contribuem para a compreensão da solução proposta e o seu desenvolvimento.

2.1 Dispositivos Móveis

A humanidade sempre se diferenciou dos demais seres vivos pela sua habilidade de comunicação, que a ajudou em sua evolução e no desenvolvimento de novas tecnologias, que por sua vez, acabou gerando novas formas de comunicação sejam elas orais, escritas ou até mesmo visuais.

O telefone, uma das invenções mais revolucionárias neste aspecto, sendo o predecessor dos dispositivos móveis, talvez não tivesse alcançado a importância que tem hoje não fosse o interesse do Brasil, especificamente de Dom Pedro II, por tal dispositivo. Segundo Cribelli (2009, p.251), o interesse do imperador pelo aparato na *Centennial Exhibition*, foi o responsável por motivar os juízes da exposição a considerar a invenção de Alexander Graham Bell, que até então estava sendo considerada um fracasso.

Após o sucesso do telefone, e com o avanço da área, novas propostas foram desenvolvidas, dentre elas o conceito de telefone móvel. Porém estes dispositivos não chamaram tanta atenção quanto seu ancestral. De acordo com Farley (2005, p.22) os *mobiles* (dispositivos móveis) foram negligenciados até o término da Segunda Guerra Mundial, ainda que houvesse tecnologias primitivas antes do início da guerra. Porém, ao final do conflito, a capacidade de comunicação foi severamente comprometida, com edificações e linhas telefônicas destruídas, gerando a necessidade de tecnologias mais flexíveis.

Com a ideia audaciosa do *mobile* ganhando espaço no mercado, e sendo a tecnologia cada vez mais aprimorada, tornou-se grande sucesso na década de 80 com a chegada dos telefones celulares, como descrito por Farley (2005, p.29, tradução nossa) “A popularidade do celular nos Estados Unidos foi inesperadamente forte. Estimativas dizem que foram 340.213 clientes em 1985; 681.825 em 1986, e 1.300.855 em 1987”[1].

Os aparelhos celulares possuem uma enorme quantidade de tecnologia embarcada, sendo oferecidos em diversos tipos de modelos, tamanhos e sistemas operacionais, transformando-se em uma ferramenta do cotidiano das pessoas, onde muitas já não conseguem viver sem ela.

2.2 Flutter

Flutter é um *framework* que se utiliza da linguagem de programação Dart, definido como

> um kit de ferramentas de interface multiplataforma que foi projetada para permitir reutilizar o código entre sistemas operacionais como IOS e Android, ao mesmo tempo que permite que os aplicativos façam interface diretamente com os serviços da plataforma subjacente (FLUTTER, 2021,*online*).

[1] No Original: *Cellular’s popularity in the United States was unexpectedly strong. Estimates say there were 340,213 customers in 1985; 681,825 by 1986, and 1,300,855 by 1987.2.*

Além de ser multiplataforma o *framework* também permite que aplicativos sejam desenvolvidos com alta qualidade suportando “uma vasta variedade de hardware (como câmera, GPS, *network* e *storage*) e serviços (como pagamentos, nuvem, autenticação e propagandas)” (FLUTTER, 2021, *online*).

Flutter utiliza a linguagem Dart que segundo o seu site oficial (DART, 2021, *online*), tem como objetivo ser

> uma linguagem otimizada ao cliente para um desenvolvimento de aplicativos rápidos em qualquer plataforma. Seu objetivo é oferecer a linguagem de programação mais produtiva para o desenvolvimento multiplataforma, emparelhada com uma execução flexível da plataforma para frameworks de aplicativos.

A linguagem foi desenvolvida e recebe total suporte da Google, sendo “uma linguagem de programação fácil de se aprender, com uma sintaxe bastante familiar” (DART, 2021, *online*). Dessa maneira, Flutter consegue ser uma ferramenta muito útil que atende às demandas deste projeto, ao fornecer uma curta curva de aprendizagem, materiais e ferramentas para desenvolver um aplicativo de forma rápida, com interfaces de nível profissional e intuitivas.

2.3 Firebase

Para que se entenda o Firebase, é importante conhecer sobre o conceito de *real time database*, segundo Lindström (2007, p.1) um *real time database system* (RTDBS)

> é um sistema de banco de dados projetado para lidar com cargas de trabalho cujo estado está em constante mudança. Este sistema difere de bancos de dados tradicionais que contêm dados persistentes, em sua maioria não afetados pelo tempo.

O Firebase é utilizado nesse projeto, uma vez que oferece ferramentas de gerenciamento de banco de dados que utilizam o conceito de *real time database*, que funciona da seguinte forma:

> O Firebase Realtime Database é um banco de dados hospedado na nuvem. Os dados são armazenados como JSON e sincronizados em tempo real com todos os clientes conectados. Quando você cria apps multiplataforma com nossos SDKs para iOS, Android e JavaScript, todos os clientes compartilham uma instância do Realtime Database e recebem automaticamente atualizações com os dados mais recentes (FIREBASE, 2021, online).

Como o intuito deste projeto é desenvolver um aplicativo multiplataforma, com diferentes dispositivos interagindo com o banco de dados, a escolha do Firebase se torna uma ótima opção que oferece um serviço de baixo custo, rápido e seguro, além de outras ferramentas, como opções de testes e Google Analytics, que podem possibilitar um futuro aprimoramento do projeto.

2.4 Assistência Domiciliar

Como o projeto tem o intuito de desenvolver para o público alvo específico da área de assistência domiciliar, é importante que se entenda seus conceitos para que seja possível traçar um perfil de usuário.

Segundo Giacomozzi (2006, p. 646) assistência domiciliar (*home-care*) refere-se a atividades que:

> constitui a modalidade geral da atenção à saúde prestada no domicílio, sendo uma categoria genérica que engloba e representa o atendimento, a visita e a internação domiciliares, cada qual com seus objetivos e características.

Na área de fonoaudiologia, segundo Barbosa (2018), o serviço de *home-care* iniciou-se em 1990, apesar do conceito já ter sido introduzido em 1968 no Hospital do Servidor Público Estadual de São Paulo, porém era restrito à vigilância epidemiológica materno-infantil, sendo difundida além de médicos e enfermagem apenas em 1990, onde profissionais de outras áreas começaram a aplicar o tratamento em suas determinadas áreas, dentre elas a fonoaudiologia.

Ainda segundo Barbosa (2018) a atuação da fonoaudiologia em *home-care* possibilitou “abrir as portas para a reabilitação e assistência domiciliar[...] nos casos de paciente com traqueostomia, comunicação e desmame”, e ainda afirma:

> A atuação da fonoaudiologia em assistência domiciliar se expandiu para cuidados paliativos, gerenciamento de doenças crônicas, Office Care, programas de prevenção de riscos, etc (BARBOSA, 2018, online).

Tendo em vista a importância da área e o seu constante serviço em *home-care*, ainda há dificuldades com relações burocráticas e de comunicação com funcionários, como fora apontado em entrevistas com pessoas da área. Dessa maneira, com a ajuda da tecnologia, esse projeto tem o intuito de auxiliar e aliviar dificuldades do dia-dia de uma empresa de fonoaudiologia, a FonoCare.

3 DESENVOLVIMENTO

O capítulo de desenvolvimento é responsável por apresentar o processo realizado durante a criação do aplicativo. Nele são abordados temas desde seu planejamento estrutural e econômico, até a codificação de um protótipo funcional que resultará na aplicação final.

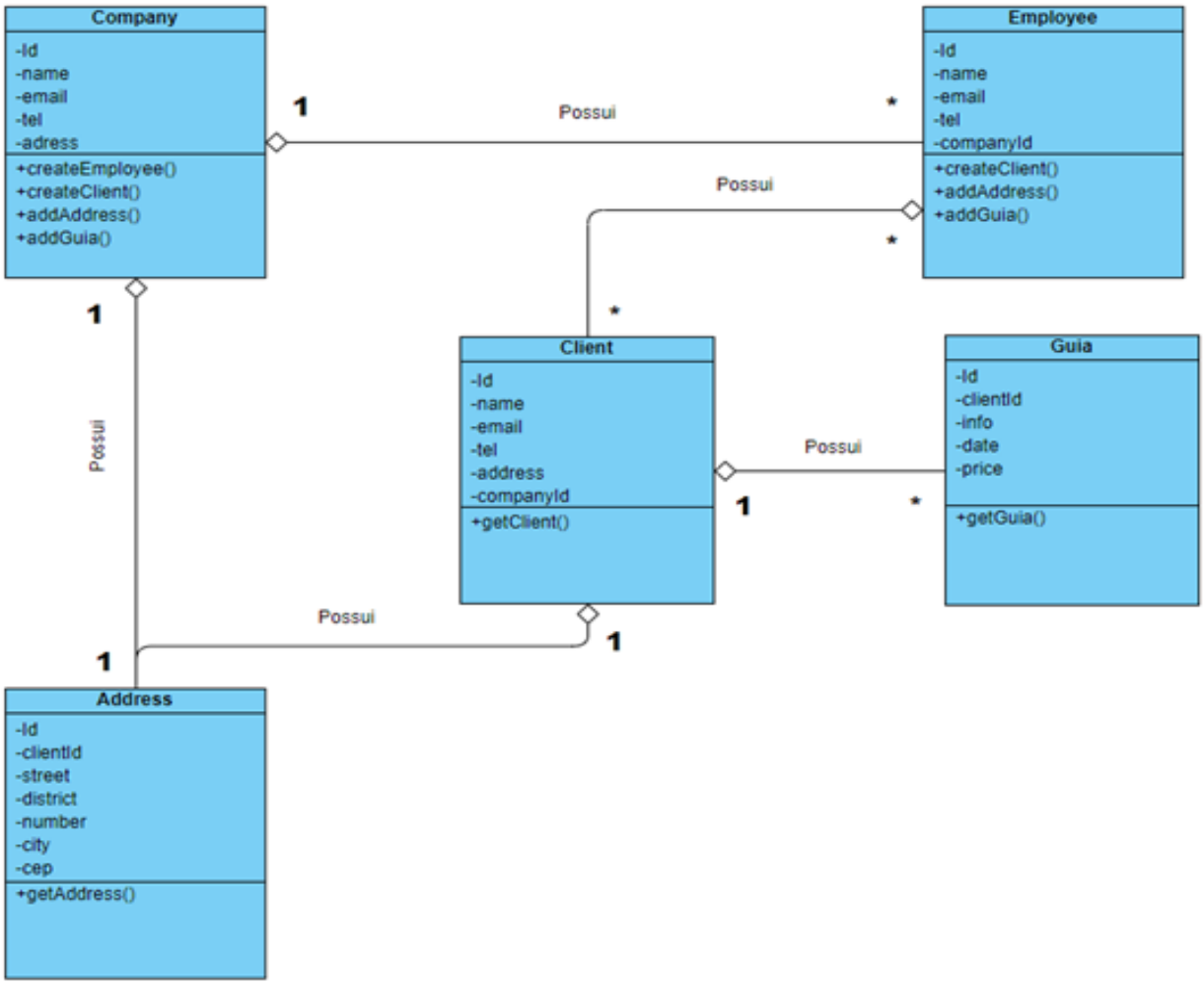
3.1 Planejamento

Neste item aborda-se a fase de planejamento do projeto, na qual serão apresentados tópicos como, Levantamento e Documentação de Requisitos, BPMN, Diagrama de Classe e de Casos de Uso, EAP, Plano 5W1H, dentre outros. Estes assuntos tratam das técnicas e ferramentas que permitem a organização e construção do projeto e facilitam futuras manutenções.

3.1.1 Diagrama de Classes

A Figura 1 apresenta um Diagrama de Classes que apresenta as principais classes que compõem este projeto, bem como, cada atributo pertencente a elas, juntamente com os métodos que cada uma pode executar, além da maneira como se relacionam.

Figura 1: Diagrama de classes entre empresa, funcionário e cliente



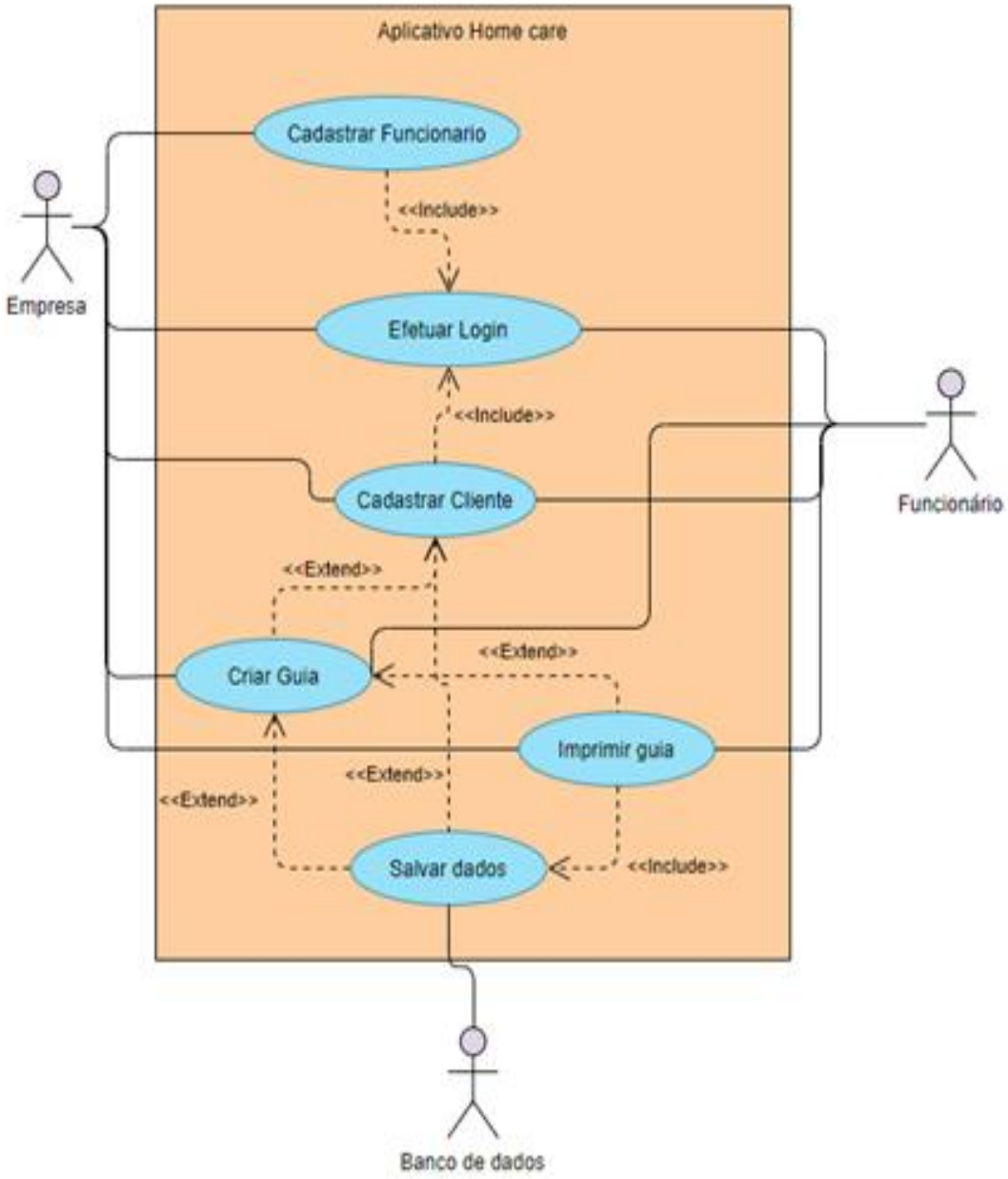
Fonte: os autores

Cada retângulo representa uma classe diferente, onde a primeira divisão representa o nome, a segunda divisão são os atributos e a última divisão são os métodos da classe. Já as linhas que conectam cada uma das classes, são como elas se relacionam, mais em específico, qual classe pertence a qual a cardinalidade dos relacionamentos.

3.1.2 Diagrama de Casos de Uso

O Diagrama de Casos de Uso, serve para demonstrar os possíveis caminhos que um usuário do sistema poderá percorrer para acessar suas funcionalidades, além de mostrar quais casos de uso irão depender de outros, como por exemplo, para a empresa cadastrar um cliente ou funcionário, é necessário que o *login* seja efetuado primeiro, porém para criar uma guia não há necessidade de criar um cliente caso já exista um na lista de clientes (Figura 2).

Figura 2: Diagrama de caso de uso entre empresa, funcionário e cliente

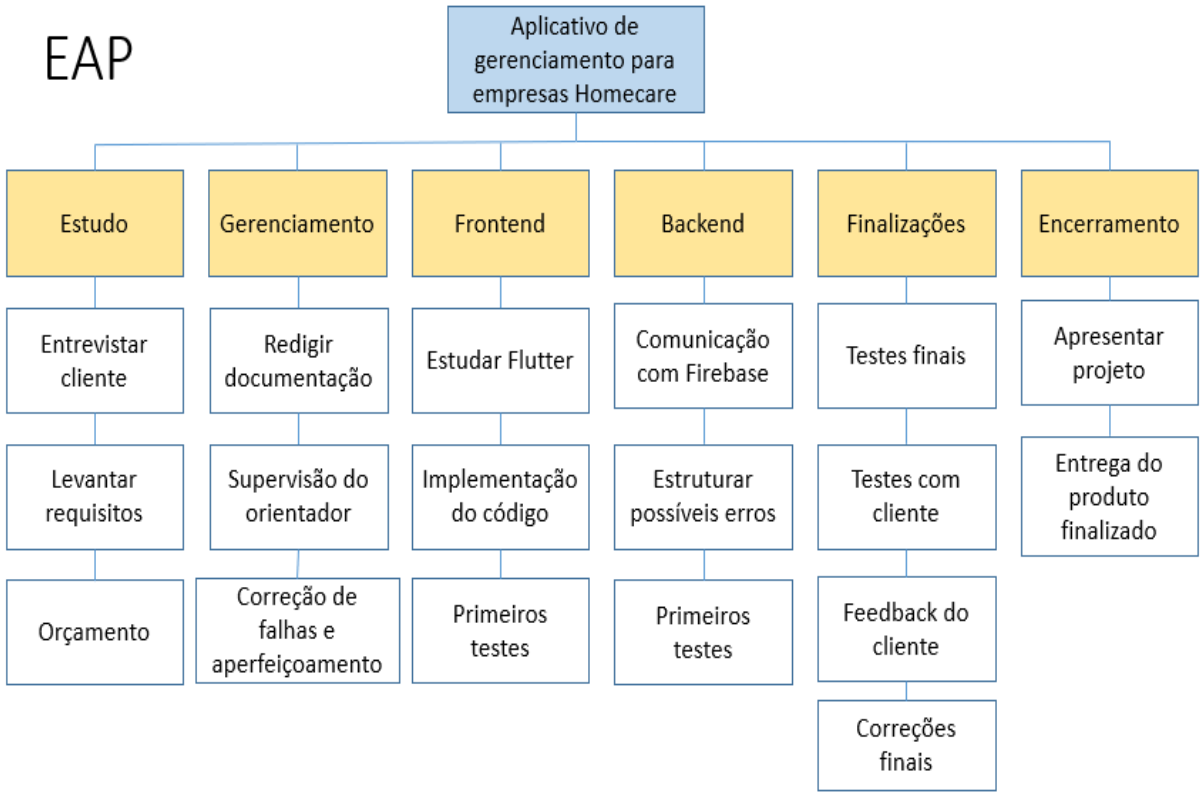


Fonte: os autores

3.1.3 EAP

A EAP (Estrutura analítica do projeto), define o escopo do projeto em que as tarefas relacionadas são separadas em sub-tarefas a serem cumpridas até o prazo de entrega. Para esse projeto, foram divididas 6 principais tarefas (Figura 3), onde as 4 primeiras representam a parte de planejamento e desenvolvimento do produto e as 2 últimas são destinadas a revisão e entrega do produto.

Figura 3: EAP do projeto



Fonte: os autores

3.1.4 Plano 5W1H

O Plano 5W1H é uma técnica composta por 6 perguntas a serem feitas sobre o projeto antes do desenvolvimento começar. As perguntas são, *What?*, *Why?*, *Where?*, *Who?*, *When?* e *How?*, do inglês respectivamente, O Quê?, Porquê?, Onde?, Quem?, Quando? e Como?.

O objetivo dessas perguntas é definir a principal linha de desenvolvimento do projeto, expondo um problema a ser solucionado, bem como a maneira e os recursos que serão utilizados para chegar a tal solução. Dependendo da escala do projeto um segundo H de *How Much?*(Quanto custa?) poderá ser feito para que haja um planejamento do impacto financeiro que a produção e pós-produção causará. A Figura 4 mostra o Plano 5W1H deste projeto.

Figura 4: Plano 5W1H do projeto

5W1H

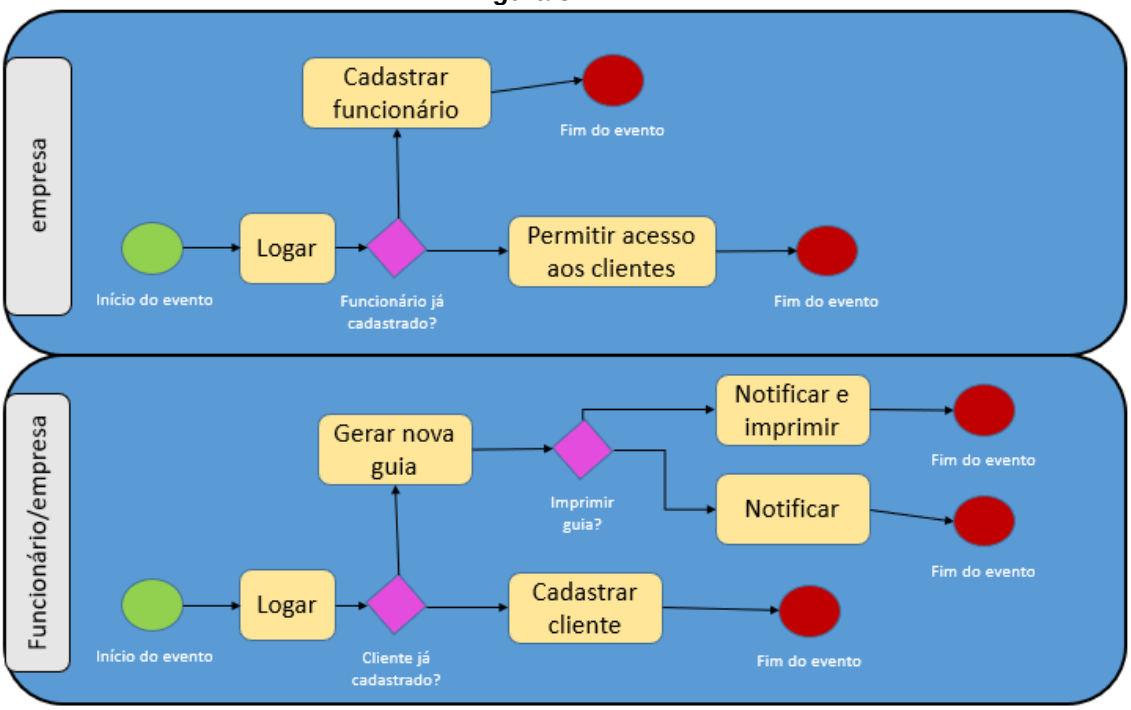
O que?	Porque?	Onde?	Como?	Quem?	Quando?
Desenvolver uma aplicação para gerar guias de pacientes de empresas homecare, além de controle de consultas e funcionários	Dificuldade de empresas homecare em organização e padronização no controle de atividades burocráticas realizadas no dia-dia das mesmas.	Empresas de Homecare e semelhantes	Através de um aplicativo de celular feito em flutter, utilizando firebase como backend para armazenamento de dados de forma rápida e segura	Funcionários e suas respectivas Empresas	Final de 2021

Fonte: os autores

3.1.5 BPMN:

O BPMN é um Modelo de Notação de Processos de Negócio, utilizado para organizar e facilitar o entendimento do usuário em como utilizar o sistema criado, através do mapeamento dos acessos às telas conforme as suas tomadas de decisões para executar uma funcionalidade. A Figura 5 mostra os processos modelados a partir do levantamento de requisitos.

Figura 5: BPMN



Fonte: os autores

3.1.6 Levantamento de Requisitos

O Levantamento de Requisitos é uma tarefa que permite identificar e compreender as necessidades do cliente ao longo do uso da aplicação, para que o

desenvolvimento do sistema seja o mais fiel possível à visão do cliente sobre o cenário em que será utilizado.

Os requisitos são divididos em duas categorias, os funcionais e os não funcionais. Os primeiros relatam todas as funcionalidades da aplicação que serão essenciais para que o sistema cumpra com seu objetivo. Sem eles, as funções ficarão inoperantes. Já os não funcionais, são funcionalidades que não são essenciais para que o sistema atenda o cliente, porém irão aprimorar as experiências do usuário com relação ao produto criado.

A seguir são descritos os requisitos definidos, a partir de estudos e entrevistas sobre as necessidades do cliente, para o MVP (*Minimum Viable Product*) a ser implementado bem como os que serão desenvolvidos posteriormente:

3.1.7 Requisitos Funcionais:

RF001 - Cadastro de clientes.
RF002 - Consultar clientes.
RF003 - Cadastro de planos de saúde.
RF004 - Consultar planos de saúde.
RF005 - Gerar guia.
RF006 - Imprimir guia.
RF007 - Consultar guias.
RF008 - Cadastrar dados básicos da guia (assinatura, preço de consulta).
RF009 - Deleção de guias e funcionários pelo usuário empresa
RF010 - Cadastro de empresa.
RF011 - Cadastro de funcionários.
RF012 - Consultar funcionários.
RF013 - Agenda de horários dos clientes e funcionários.
RF014 - Cadastro de batimento de ponto dos funcionários de acordo com a distância de seu cliente.
RF015 - Gerar fotos de progresso do cliente.
RF016 - Consultar fotos.
RF017 - Mostrar endereço do cliente no mapa.

3.1.8 Requisitos não Funcionais:

RNF001 - O usuário poderá recuperar sua senha através de um link enviado por e-mail.
RNF002 - Clientes só podem ser vistos por suas respectivas empresas e funcionários.
RNF003 - Validação de pagamento pelo uso do aplicativo.
RNF004 - Ícones de *loading* enquanto o sistema busca os dados.
RNF005 - A criação de horários dos clientes e funcionários só poderá ser feita por uma conta empresa
RNF006 - O funcionário só poderá bater o ponto caso esteja próximo ao local de consulta.
RNF007 - O cadastro de funcionários só poderá ser feito por uma conta de empresa.

3.1.9 Canvas

As Regras de Negócio permitem que sejam organizadas as formas de conduta de uma empresa, para que esta consiga estabelecer como será produzido o seu produto e quem é o público alvo a ser atingido.

O Canvas, de acordo com Qastharin (2016, tradução nossa), “é uma ferramenta que possibilita criar um modelo de negócio simples que permite a fácil compreensão enquanto captura a complexidade de como a empresa funciona”[2], sendo assim, se torna uma ótima maneira para determinar quais são os objetivos a serem atingidos, bem como, a forma na qual serão trabalhados ao criar um aplicativo como um produto.

O Canvas do projeto, como pode ser visualizado na Figura 6, expressa as principais propostas da aplicação, além de como serão realizadas, operadas e os principais pontos a serem considerados nos quesitos de custos e fontes de renda.

Figura 6: Canvas

Fonte: os autores

3.2 Codificação

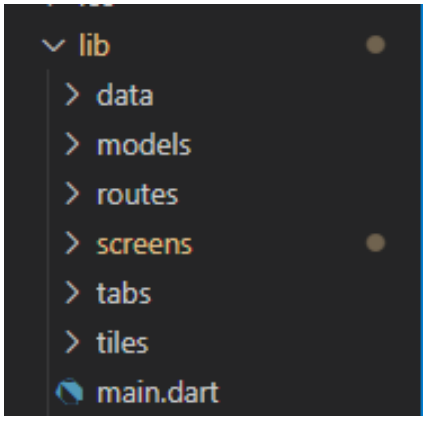
Nessa seção do artigo são apresentados e explicados o código e a estrutura utilizada para o desenvolvimento da aplicação, que está disponível no repositório do Github[3]. O projeto foi desenvolvido utilizando o *framework* Flutter, conforme comentado em seções anteriores, e o editor de código fonte VS Code, por

[2] No Original: *it makes a useful tool to understand the business model of an enterprise and to conduct business model innovation.*
[3] https://github.com/Eduardo-XavierPaula/meu-tcc

ser um editor gratuito, que permite a utilização de várias extensões que facilitam a criação e a manutenção do código durante a programação.

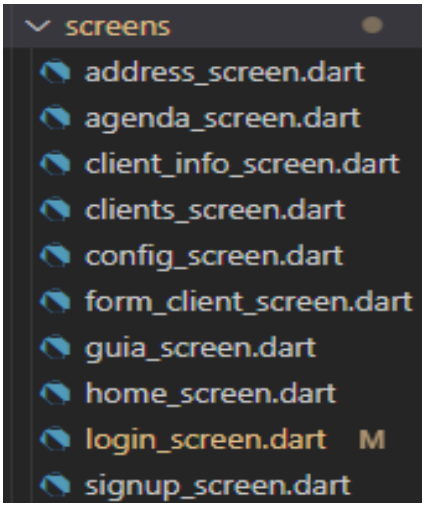
A estrutura do projeto é realizada em divisões de pastas e arquivos conforme mostrado nas Figura 7 e 8.:

Figura 7: Estrutura de pastas do projeto



Fonte: os autores

Figura 8: arquivos do projeto



Fonte: os autores

Conforme apresentado na Figura 8, os arquivos de código da aplicação se localizam dentro de suas respectivas pastas; na Figura 7 é demonstrado que todas as pastas e arquivos que contêm o código do projeto, se localizam por padrão do Flutter, dentro da pasta *lib,* onde há subpastas para organização e padronização dos arquivos.

A pasta *data* contém arquivos que definem os atributos dos objetos que serão manipulados pelo banco de dados, definindo os tipos de dados que esses atributos irão receber (*String*, *Number*, etc.), a forma como eles são representados dentro do banco, e quais os dados serão manipulados pelos demais arquivos. Na Figura 9 pode-se observar um exemplo de classe modelo, em que são realizados os mapeamento dos atributos da classe *ClienteData*.

Figura 9: Código do client_data.dart.

```dart
class ClientData{
  String userId;
  String cid;
  String name;
  String email;
  num tel;


  ClientData();
  ClientData.fromDocument(DocumentSnapshot snapshot){
    cid = snapshot.documentID;
    userId = snapshot.data["userId"];
    name = snapshot.data["name"];
    email = snapshot.data["email"];
    tel = snapshot.data["tel"] + 0.0;
  }

  Map<String,dynamic> toMap(){
    return{
      "name":name,
      "email":email,
      "tel":tel,
      "userId":userId
    };
```

Fonte: os autores

A pasta *models* contém arquivos com funções modelos para a realização de métodos CRUD (*Create, Read, Update* e *Delete)*, de cada um dos objetos representados como as classes do Diagrama de Classes na Figura 1. Essas funções podem ser chamadas em todos os demais métodos do projeto, para que não haja a necessidade de criar uma função em cada arquivo que precise de uma manipulação dos dados, conforme mostrado na Figura 10.

Figura 10: Exemplo da função addClient do arquivo client_model.dart.

```dart
void addClient(ClientData clientData) {
  products.add(clientData);

  Firestore.instance
      .collection("clients")
      .add(clientData.toMap())
      .then((doc) {
    clientData.cid = doc.documentID;
    clientData.userId=user.firebaseUser.uid;
  });

  notifyListeners();
}
```

Fonte: os autores

Em *screens* é onde estão localizados todos os arquivos de telas da aplicação, em que cada tela é responsável por manipular seus respectivos dados ou navegar para outras telas. No Flutter uma tela é composta por *Widgets*, que geralmente possuem um *appBar* para definir a barra da aplicação e o *body* que define

o corpo, sendo que, dentro desses *Widgets*, é possível retornar outros arquivos *Widgets*, para manter o código organizado e fácil de criar estruturas que se repetem muito dentro de uma tela (*Tiles)*, como por exemplo, os *cards* que exibem os dados de cada cliente, ou até mesmo o corpo inteiro da tela (*Tabs*). A Figura 11 apresenta um exemplo de um *Widget* de tela (Screen).

Figura 11: Exemplo de Widget para tela do cliente do arquivo client_screen.dart.

```
Widget build(BuildContext context) {
  return Scaffold(
    appBar: AppBar(
        title: Text("Clientes"),
        centerTitle: true,
        elevation: 0,
        ), // AppBar
    body:ScopedModelDescendant<ClientModel>(builder: (context, child, model){
        return ClientsTab();
    },), // ScopedModelDescendant


    floatingActionButton: FloatingActionButton(
    onPressed: () {
      Navigator.of(context).pushNamed("/form").then((value) => setState(() {}));
    },
    child: const Icon(Icons.person_add_alt_1_rounded),
    backgroundColor: Theme.of(context).primaryColor,
  ),        // FloatingActionButton
  ); // Scaffold
}
```

Fonte: os autores

O único arquivo que não fica dentro de pastas é o *main.dart,* responsável por inicializar a aplicação e chamar a primeira tela do aplicativo, que no caso desse projeto, há duas telas diferentes dependendo da condição em que o aplicativo se encontra: se o usuário está desconectado da aplicação, a tela inicial será a de *login*, caso contrário, a tela exibida é a tela de Menu da conta do usuário. A Figura 12 apresenta um exemplo de um *Widget* no arquivo *main.dart*.

Figura 12: Exemplo de Widget do arquivo main.dart.

```
@override
Widget build(BuildContext context) {
  return ScopedModel<UserModel>(
      model: UserModel(),
      child: ScopedModelDescendant<UserModel>(
        builder: (context, child, model) {
          return ScopedModel<ClientModel>(
            model: ClientModel(model),
            child: MaterialApp(
              title: "Home Care",
              theme: ThemeData(
                  primarySwatch: Colors.blue,
                  primaryColor: Color.fromARGB(255, 4, 125, 141)), // T
              debugShowCheckedModeBanner: false,
              home: (model.isLoggedIn())?HomeScreen():LoginScreen(),
              onGenerateRoute: RouteGenerator.getRoute,
            ), // MaterialApp
          ); // ScopedModel
        },
      )); // ScopedModelDescendant // ScopedModel
}
```

Fonte: os autores

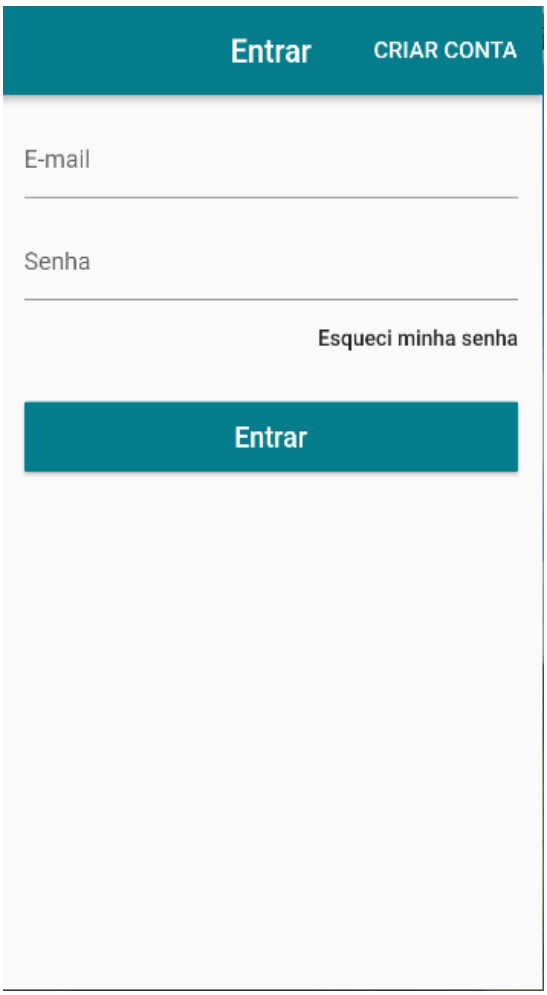
4 RESULTADOS

Nesse tópico são abordados e apresentados os resultados das telas que serão utilizados no aplicativo, bem como, a manipulação dos dados entre aplicativo e banco de dados e como eles estão sendo distribuídos no Firebase.

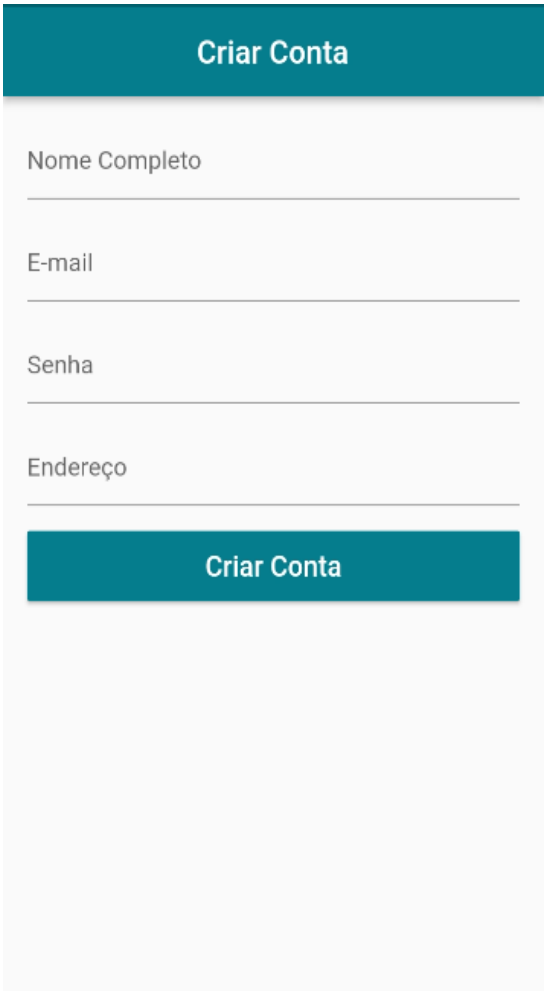
A tela de *login* e cadastro são dois formulários básicos nos quais os usuários deverão preencher seus dados. Após o sistema confirmar os dados do usuário, ele é levado para a tela principal. As telas mencionadas estão representadas nas Figuras 13 e 14.

Figura 13: Tela de Login.

Figura 14: Tela de Cadastro



Fonte: os autores



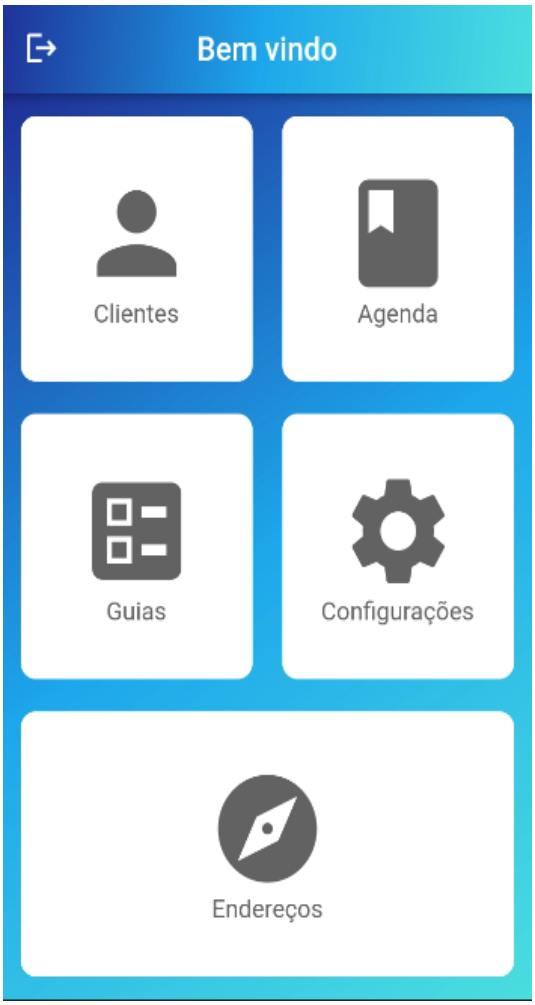
Fonte: os autores

Ao logar em sua conta, o usuário é direcionado à tela principal que consiste em um Menu para interagir com as opções e ferramentas do aplicativo, conforme a Figura 15. Atualmente ela contém 5 botões para filtrar as principais funcionalidades que foram desenvolvidas, e as que ainda serão implementadas.

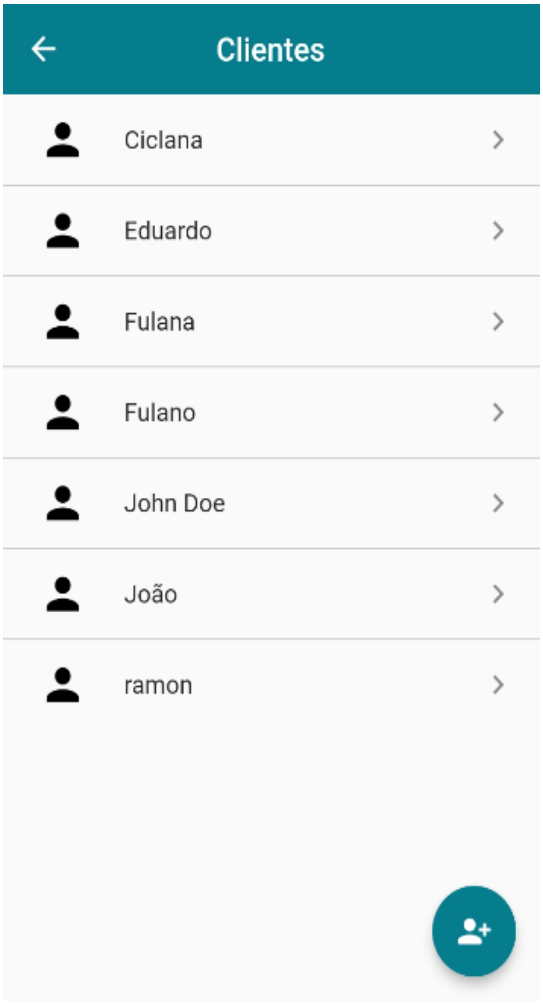
O botão Clientes leva o usuário a uma tela com uma lista dos clientes adicionados ao sistema do usuário logado (Figura 16), que além da lista, possui um botão que direciona a um formulário para adicionar um novo cliente. Assim que os dados forem salvos, o sistema irá retornar o usuário para a lista, já com o novo usuário inserido.

Figura 15: Tela Principal.

Figura 16: Tela de Clientes.



Fonte: os autores



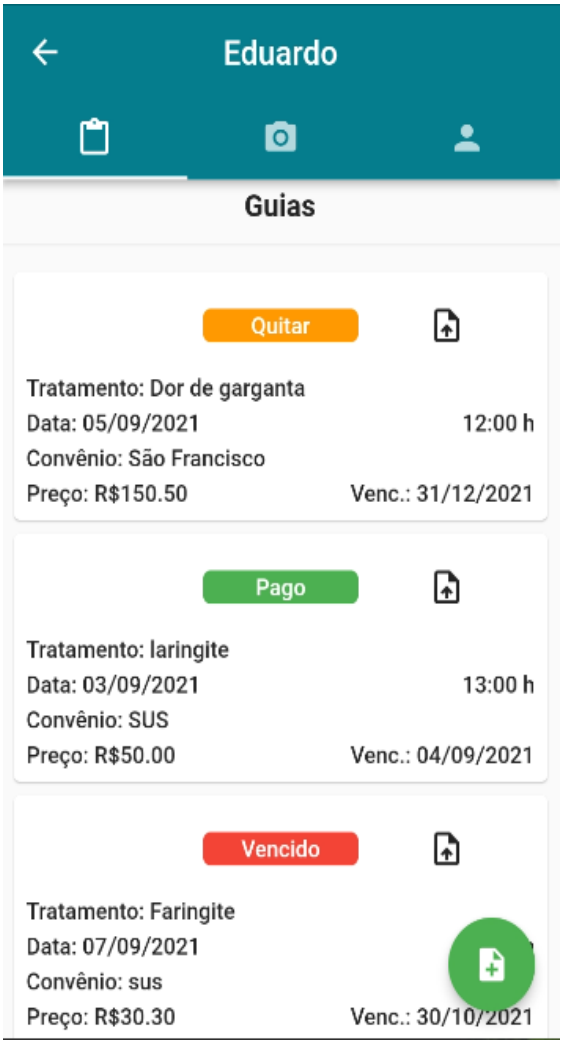
Fonte: os autores

Ao acessar um Cliente, os seus dados são apresentados, incluindo uma lista das guias que foram criadas para ele, conforme a Figura 17. As guias são distribuídas em *cards*, contendo informações do tratamento realizado durante uma consulta, o convênio do cliente, o preço, a data e hora da consulta, bem como, se o cliente pagou ou não o atendimento realizado pelo profissional.

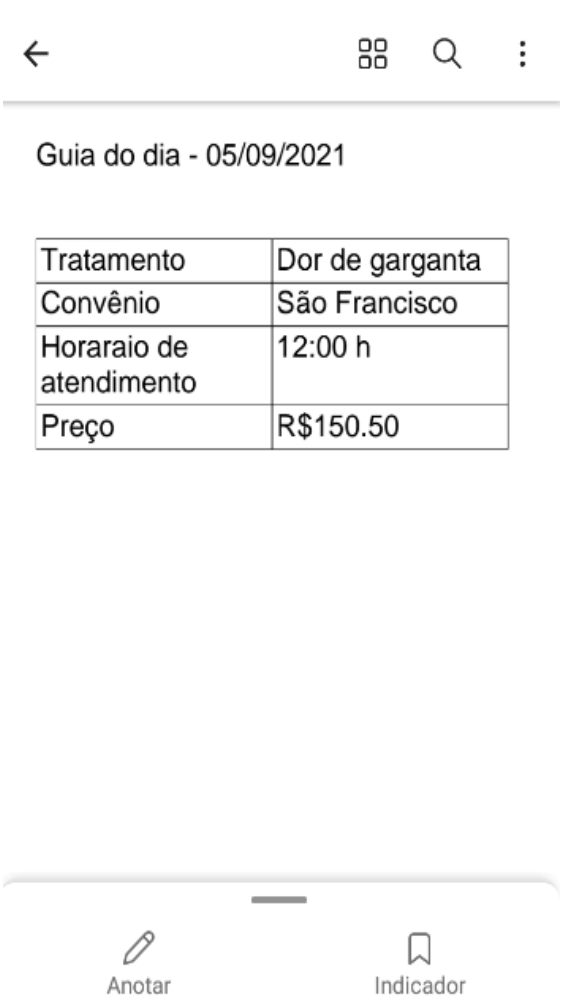
Se o usuário clicar no ícone para exportar os dados da guia para PDF, no canto superior direito de cada card, os dados são enviados a um arquivo como demonstrado na Figura 18, o qual poderá ser assinado pelo Cliente, além de ser salvo ou impresso conforme as necessidades.

Figura 17: Tela do cliente.

Figura 18: Tela para PDF.



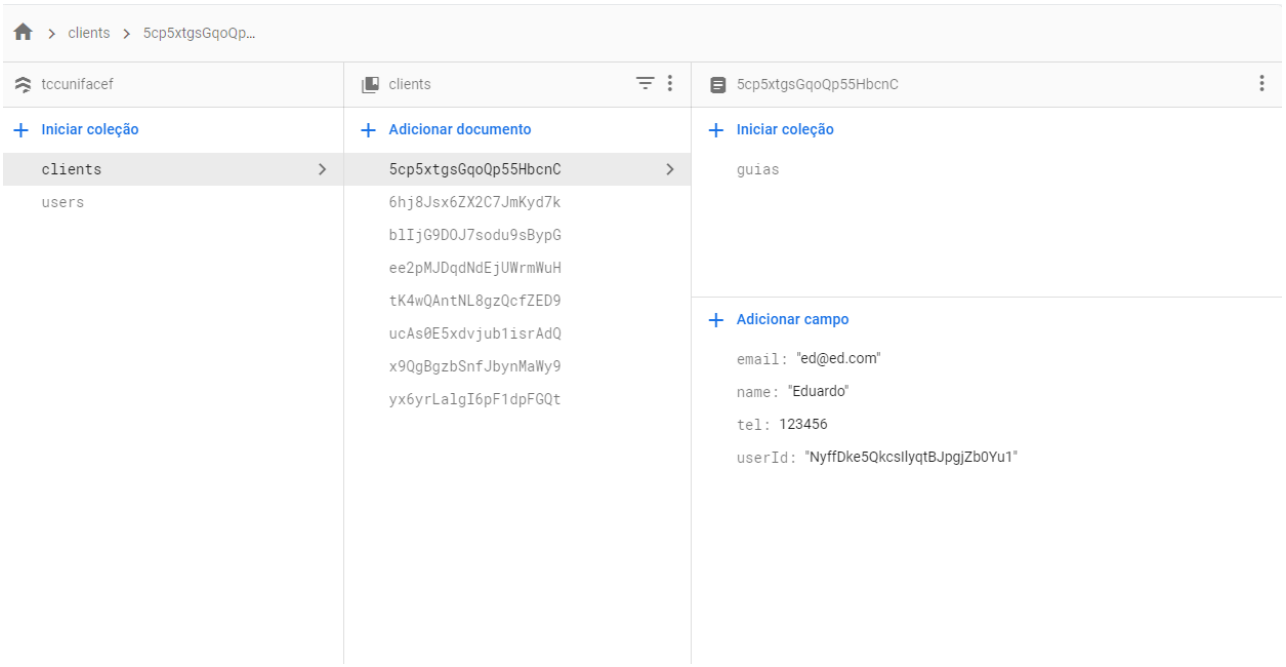
Fonte: os autores



Fonte: os autores

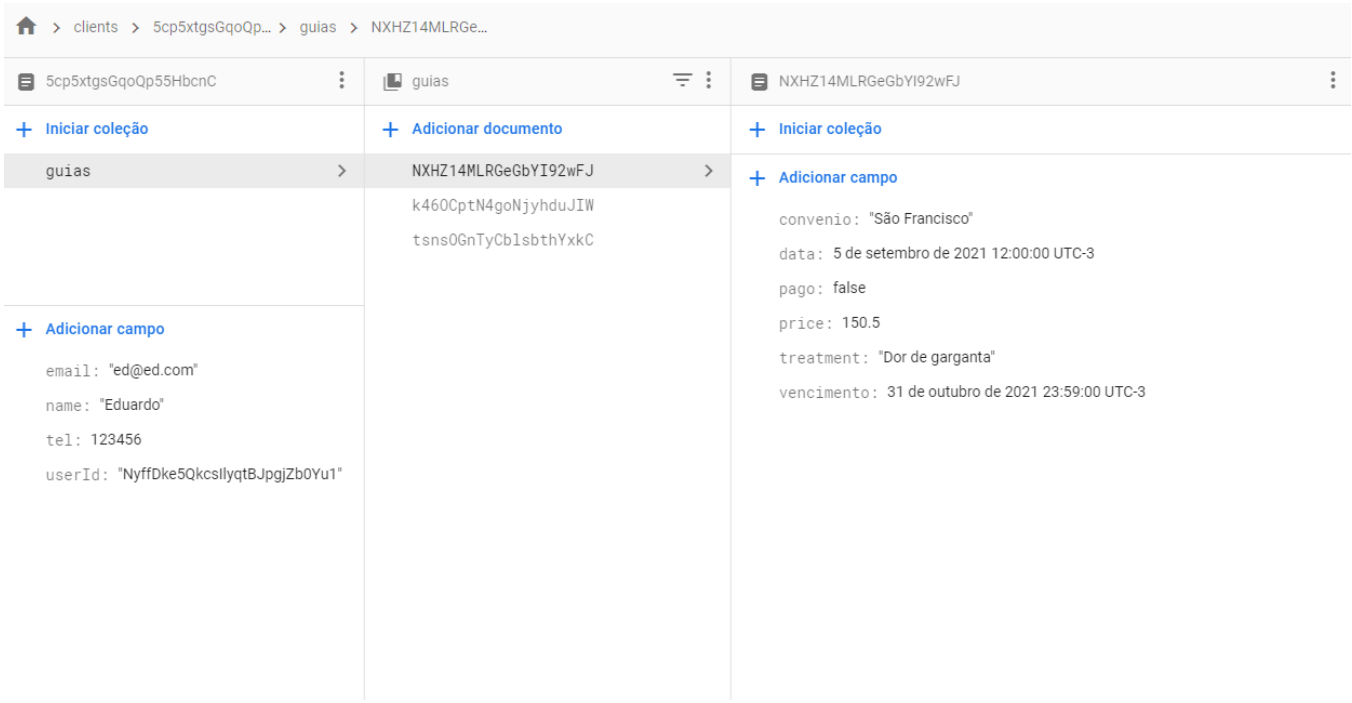
Por fim, como já mencionado neste documento, os dados da aplicação são salvos no Firebase. Cada conjunto de dados é salvo dentro de uma coleção. No caso *clients* e *users*, onde são armazenados documentos de clientes e usuários respectivamente. Dentro de cada documento, fica salvo cada campo pertinente àquele cliente/usuário. Cada documento pode ter uma coleção também. No caso dos clientes, cada um deles terá sua própria coleção de guias, onde os seus documentos irão representar cada guia gerada para aquele cliente. A Figura 19 exibe o *layout* na plataforma do Firebase da coleção de clientes, enquanto que a Figura 20, exibe a coleção de guias que está dentro do documento representando o cliente Eduardo.

Figura 19: Firebase coleção de clientes.



Fonte: os autores

Figura 20: Firebase coleção de guias do cliente Eduardo.



Fonte: os autores

5 CONSIDERAÇÕES FINAIS

O projeto encontra-se em fase inicial, contendo aberturas para aperfeiçoamentos e aplicações de novas funcionalidades. Apesar do desenvolvimento ter entregue a funcionalidade principal para emitir os dados de guias, ainda é necessário tratamento dos dados relevantes, bem como o aprimoramento do *layout* de exibição.

Outro ponto a ser levado em consideração, é que embora o desenvolvimento para gerar guias tenha sido realizado, outras funcionalidades apontadas nos tópicos 3.7 e 3.8 sobre Requisitos, deixaram de ser implementadas e serão melhor elaboradas e aperfeiçoadas conforme as necessidades dos clientes. Os Requisitos mencionados se referem aos listados entre RF008 e RF017 para os funcionais, e RNF003 a RNF007 para os não funcionais, que consistem em permitir que os usuários possam também, adicionar uma conta Empresa e relacioná-la aos seus funcionários. Além de outras opções de navegação, que permitem ao usuário tirar fotos do seu cliente para acompanhamentos, ou editar e excluir os dados criados.

Por fim, a utilização de metodologias aplicadas à área de negócio permitiu determinar os principais detalhes a serem abordados e trabalhados para a elaboração de um produto digital, tendo sido o foco deste artigo, voltado para os quesitos técnicos de desenvolvimento da aplicação.

Assim sendo, mesmo com um objetivo claro, para que esse produto seja oficialmente lançado com boas perspectivas de negócio, rentável tanto para os desenvolvedores, quanto para os usuários, é necessário mais análises e testes para obtenção de *feedback* dos profissionais da área de *home-care,* com uma versão *alpha*, para que se possa determinar o custo real da aplicação, bem como os valores a serem cobrados para sua utilização.

REFERÊNCIAS

CRIBELLI, Teresa. A Modern Monarch: Dom Pedro II's Visit to the United States in 1876. **Journal of The Historical Society**, v. 9, n. 2, p. 223-254, 2009.

DART. **Dart**, 2021, Disponível em: https://dart.dev/overview. Acesso em: 25 de mar. de 2021.

FARLEY, Tom. Mobile telephone history. **Telektronikk**, v. 101, n. 3/4, p. 22, 2005.

FIREBASE. **Firebase Docs**, 2021, Disponível em: https://firebase.google.com/docs/database?hl=pt-br. Acesso em: 15 de abr. de 2021.

FLUTTER. **Flutter Docs**, 2021, Disponível em: https://flutter.dev/docs/resources/architectural-overview. Acesso em: 25 de mar. de 2021.

GIACOMOZZI, Clélia Mozara; LACERDA, Maria Ribeiro. A prática da assistência domiciliar dos profissionais da estratégia de saúde da família. **Texto & Contexto-Enfermagem**, v. 15, n. 4, p. 645-653, 2006.

LINDSTRÖM, Jan. Real Time Database Systems. **Wiley Encyclopedia of Computer Science and Engineering**, p. 1-13, 2007.

QASTHARIN, Annisa R. Business model canvas for social enterprise. **Journal of Business and Economics**, v. 7, n. 4, p. 627-637, 2016.