UNIVERSIDADE ESTADUAL PAULISTA "JÚLIO DE MESQUITA FILHO"
FACULDADE DE CIÊNCIAS
BACHARELADO EM SISTEMAS DE INFORMAÇÃO

Trabalho de Conclusão de Curso

José Vitor Thomaz

MyClinic App - APLICATIVO PARA GESTÃO DE CLÍNICAS DE PSICOLOGIA

Bauru - SP
2023

JOSÉ VITOR THOMAZ

MyClinic App - APLICATIVO PARA GESTÃO DE CLÍNICAS DE PSICOLOGIA

Trabalho apresentado em cumprimento às exigências do Curso de Bacharelado em Sistemas de Informação, da Faculdade de Ciências da UNESP Campus Bauru, para obtenção do título de Bacharel em Sistemas de Informação.

Orientador: Prof. Dr. José Remo Ferreira Brega.

Bauru - SP
2023

Thomaz, José Vitor
T465m MyClinic App - APLICATIVO PARA GESTÃO DE CLÍNICAS DE PSICOLOGIA / José Vitor Thomaz. -- Bauru, 2023
58 p. : il., tabs., fotos

Trabalho de conclusão de curso (Bacharelado - Sistemas de Informação) - Universidade Estadual Paulista (Unesp), Faculdade de Ciências, Bauru
Orientadora: José Remo Ferreira Brega

1. Flutter. 2. Dart. 3. Aplicativo móvel. I. Título.

Sistema de geração automática de fichas catalográficas da Unesp. Biblioteca da Faculdade de Ciências, Bauru. Dados fornecidos pelo autor(a).

Essa ficha não pode ser modificada.

Agradecimentos

Primeiramente, agradeço aos meus pais e a minha família por todo o apoio e incentivo para a realização da graduação. Agradeço também à Thainá pelo apoio durante toda a graduação e durante o desenvolvimento desse trabalho de conclusão de curso.

Gostaria de agradecer ao meu orientador, Professor Dr. José Remo Ferreira Brega, que me acolheu e auxiliou durante o processo de desenvolvimento deste trabalho. Agradeço também a todos os professores que participaram da minha formação e tanto me ensinaram nesses anos de curso.

Por fim, agradeço a Unesp, seus professores, funcionários e a todos que fizeram parte, direta ou indiretamente, desta fase da minha vida.

A todos esses, os meus mais sinceros agradecimentos.

RESUMO

A saúde mental é um tema de extrema importância e que tem ocupado cada vez mais o debate na sociedade. Nesse contexto, o número de pessoas que buscam atendimento psicológico também cresce, de acordo com dados de pesquisas realizadas nos últimos anos no Brasil. Com essa crescente, a sobrecarga de atendimento dos profissionais psicólogos e a necessidade de ter soluções tecnológicas para gerenciamento de suas rotinas de atendimento tem se mostrado nítidas e de grande importância. Este documento reflete um estudo feito para desenvolvimento de um aplicativo móvel que atue em ambas as plataformas Android e iOS, de forma a contemplar o maior número de pessoas possível. Para isso, foi utilizado o framework de desenvolvimento de interfaces Flutter, que atua com o conceito de desenvolvimento único para ambos os sistemas operacionais descritos. Este aplicativo busca auxiliar os profissionais da área da psicologia que possuem e trabalham em clínicas. Busca apresentar ferramentas de gestão de diárias como cadastro de colaboradores, cadastro de novos agendamentos, manipulação de agenda, com interfaces exclusivas para usuários administradores e colaboradores.

Palavras-chave: Aplicativo móvel, Saúde mental, Android, iOS, Flutter, Dart, Riverpod.

ABSTRACT

Mental health is an extremely important topic that has increasingly occupied the debate in society. In this context, the number of people seeking psychological care also grows. With this increase, the overload of care for professional psychologists and the need to have technological solutions to manage their care routines has become clear and of great importance. This document reflects a study carried out to develop a mobile application that works on both Android and iOS platforms, in order to reach as many people as possible. For this, the Flutter interface development framework was used, which works with the concept of single development for both operating systems described. This application seeks to help professionals in the field of psychology who own and work in clinics. It seeks to present daily management tools such as employee registration, registration of new appointments, agenda manipulation, with exclusive interfaces for administrator users and employees.

Keywords: Mobile App, Mental Health, Android, iOS, Flutter, Dart, Riverpod.

SUMÁRIO

LISTA DE FIGURAS

1 INTRODUÇÃO

Independente do tipo e do formato, a busca por atendimento médico é rotineira tanto para pacientes quanto para profissionais. Considerando o contexto da pandemia de Covid-19, em 2020, esses atendimentos precisaram passar por diferentes mudanças, muitas delas possíveis graças à tecnologia.

Considerando o contexto da saúde mental, também foram registrados aumentos na busca por atendimento psicológico. De acordo com o Instituto Ipsos, na pesquisa “One Year of Covid-19”,

> “53% das pessoas entrevistadas no Brasil acreditam que sua saúde mental mudou para pior desde o início da crise de Covid-19. O número de respondentes que afirma ter tido uma melhoria na saúde mental é de 14%, e 34% não notaram qualquer diferença neste quesito”. (2022, Instituto Ipsos)

Outro dado analisado pelo Instituto Ipsos também demonstra um aumento da preocupação dos brasileiros com a saúde mental. O estudo “Global Health Service Monitor 2023” mostrou que cinco em cada dez brasileiros (52%) acreditam que a saúde mental é o principal problema do país em termos de bem-estar da população.

Como resultado, também pôde ser observado um aumento na procura por atendimento especializado de saúde mental. A Associação Brasileira de Psicologia da Saúde revelou que, de acordo com pesquisa da classe, mais de 80% dos psicólogos tiveram aumento de demanda de pacientes durante a pandemia, com dados divulgados na Folha de S. Paulo.

Tendo isso em vista, é possível considerar que o aumento na busca por esse tipo de atendimento também aumenta, como consequência, a necessidade de uma organização funcional e facilitada para profissionais que atuam com a saúde mental.

Partindo desse pressuposto e fazendo uma análise de concorrentes, foi possível encontrar aplicações mobile e sistemas web criados para atender a essa necessidade de organização, como já era esperado.

Porém, foi interessante perceber, também, que muitos desses sistemas não atendiam a requisitos como facilidade de uso, usabilidade e intuitividade. Assim, não

eram capazes de gerar autonomia para os profissionais quando se tratava de seus agendamentos (vide análise no tópico 1.2).

Outro ponto observado nessa análise refere-se ao escopo dos sistemas, geralmente criados considerando uma ampla complexidade de organização e de atendimentos, uma vez que o foco estava no atendimento médico geral, e não focado para a realidade de clínicas de saúde mental.

Após essa análise do contexto mundial, aumento da preocupação com a saúde mental e a busca por atendimento, além dos sistemas existentes, foi possível idealizar um projeto com escopo e requisitos mais específicos.

Com isso, a ideia desse trabalho é desenvolver um sistema de gestão e agendamentos de horários e atendimentos para clínicas voltadas à saúde mental que contam com diferentes profissionais atendendo no local, sejam com diferentes áreas ou não.

Para a elaboração dessa solução tecnológica, foram consideradas também clínicas de pequeno e médio porte, que podem se beneficiar com o uso de um sistema clínico, mas, ao mesmo tempo, não necessitam de um sistema complexo e de alto custo para isso.

O objetivo está em simplificar os agendamentos e aumentar a autonomia dos profissionais para acessar sua agenda de atendimento e visualizar seus atendimentos.

Para isso, foram utilizadas a linguagem de programação Dart, com o *framework* Flutter. Também foram consideradas as boas práticas de programação, assim como conceitos de usabilidade do usuário. Mais informações sobre o desenvolvimento podem ser encontradas nos tópicos a seguir.

1.1 DETALHAMENTO DO PROBLEMA

Vide a complexidade que a administração de uma clínica de saúde requer, o principal objetivo do aplicativo MyClinic App é facilitar a estrutura organizacional da clínica que, sendo uma empresa, busca por melhorias da organização para atendimentos dos clientes e também uma profissionalização da própria clínica.

Em análise feita nos centros de atendimento psicossocial da cidade de São Paulo, Schran, Machineski, Rizzotto e Caldeira (2019) avaliaram que “que a estrutura organizacional dos serviços e a composição das equipes impactam na realização do cuidado integral aos usuários e seus familiares”.

O aplicativo MyClinic App foi feito considerando um contexto de clínica de saúde com atendimento de diversos profissionais, cada um com suas especificidades de horário, de tempo e dia de atendimento.

Considerando esse contexto, foi possível inferir algumas dificuldades que surgem quando é preciso organizar o trabalho de diferentes profissionais como:

- dificuldade no controle de agenda, principalmente quando há mais de um profissional;
- erros de agendamento, considerando dias e horários de cada profissional;
- diferenças de horários de atendimento para cada profissional;
- falta de autonomia dos profissionais para marcar seus próprios atendimentos;
- controle das informações dos profissionais que trabalham na empresa.

Tendo isso em vista, o objetivo é contribuir com a organização da rotina de atendimentos dos profissionais de uma clínica a partir de um aplicativo intuitivo, que permita mais autonomia de cada profissional com sua agenda, além de uma visualização mais ampla da organização da clínica.

A escolha de fazer isso em um aplicativo parte do aumento do uso de dispositivos móveis, o que torna a interface de aplicativo mais intuitiva para o usuário. De acordo com a pesquisa sobre o uso de tecnologias da informação e comunicação nos domicílios brasileiros – TIC Domicílios 2022, divulgada pelo Comitê Gestor da Internet no Brasil (CGI.br) e pela Agência Brasil, “até 2014, o computador era o dispositivo mais usado para acessar a internet, por 80% dos usuários. De lá para cá, muitos dos novos usuários acessavam exclusivamente pelo telefone celular”.

Outro ponto considerado é a facilidade que o profissional tem em fazer a gestão dos atendimentos, já que pode acessar o aplicativo onde e quando quiser.

Essa funcionalidade se mostra útil principalmente quando o profissional trabalha em mais de um local. Assim, mesmo não estando na clínica em questão, consegue acessar o aplicativo e realizar o agendamento.

1.2 SOLUÇÕES EXISTENTES

Antes de iniciar a fase de desenvolvimento do projeto, foi feita a análise dos sistemas e aplicativos já existentes que teriam a mesma solução proposta. A partir

dessa análise, foi possível observar estruturas a serem seguidas e também outras que poderiam ser melhoradas em um novo aplicativo.

1.2.1 AGENDART

Um dos aplicativos analisados foi o Agendart, que é focado em gestão de clínicas de saúde. A intuitividade do aplicativo é um dos pontos positivos analisados, já que o aplicativo é fácil e rápido de ser entendido.

Porém, também tende a ser bastante complexo, servindo a grandes clínicas, o que pode não ser tão útil para clínicas menores, que precisam de soluções mais focadas e de menor custo.

Figura 1: Aplicativo Agendart

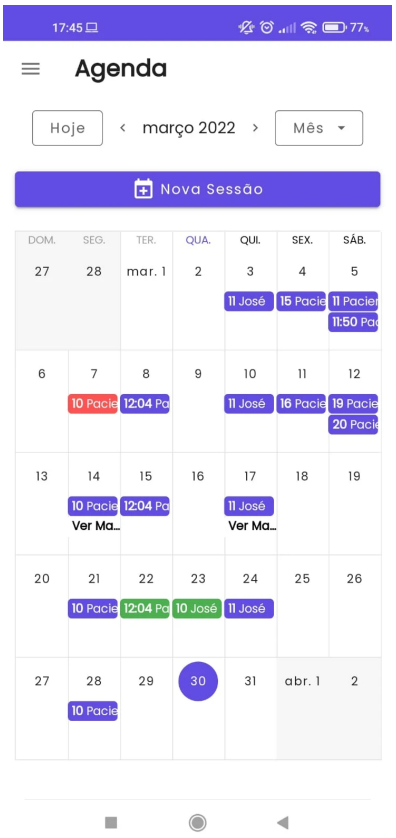
Fonte: Divulgação Agendart

1.2.2 PSICOPLANNER

Outro aplicativo analisado foi o Psicoplanner, um aplicativo de gestão para psicólogos, para controle das sessões, agenda online, etc. É o que mais se assemelha com o projeto proposto e, por isso, foi também usado como referência, principalmente nos agendamentos.

Porém, o aplicativo analisado não está focado na gestão de clínicas em si, e sim no profissional, que organiza apenas seus próprios atendimentos pelo aplicativo. Por isso, a sugestão do projeto proposto é aumentar esse escopo para permitir uma gestão de equipes em uma clínica, de maneira autônoma mas também colaborativa.

Figura 2: Aplicativo Psicoplanner



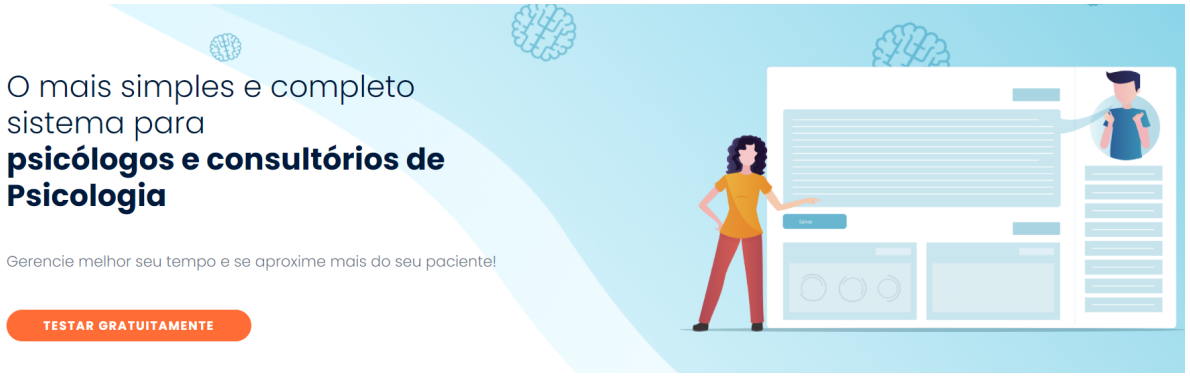
Fonte: Divulgação Psicoplanner

1.2.3 PSICOMANAGER

Outro produto que se assemelha, mas que também apresenta escopo amplo e mais geral. Além disso, seu principal sistema é pensado para desktop e não para mobile.

Como ponto positivo, é um sistema com diversas funcionalidades que podem ser aplicadas posteriormente no projeto proposto neste documento. Porém, para criação de um aplicativo com foco em pequenas clínicas, foi feita uma análise para mapear apenas as principais funcionalidades e aplicá-las no projeto.

Figura 3: Aplicativo PsicoManager



Fonte: https://www.psicomanager.com.br/

No apêndice 2, é possível ver um quadro comparativo entre algumas das funcionalidades do MyClinic App com os aplicativos aqui estudados.

1.4 ESTRUTURA DO DOCUMENTO

Esta seção fez uma breve descrição do problema a ser solucionado, das soluções existentes no mercado que se assemelham a ideia proposta, e do objetivo principal a ser alcançado.

Nas próximas seções serão abordados,de forma mais profunda, o projeto em si, com todas as suas especificações como tecnologias a serem utilizadas, levantamento de requisitos, diagramas criados, processo de desenvolvimento do *front-end* e do *back-end*. Por fim, estão dispostos um capítulo de conclusão, onde se retoma os pontos relevantes do desenvolvimento desse projeto e um capítulo onde se faz uma análise com sugestões para continuidade do desenvolvimento deste projeto.

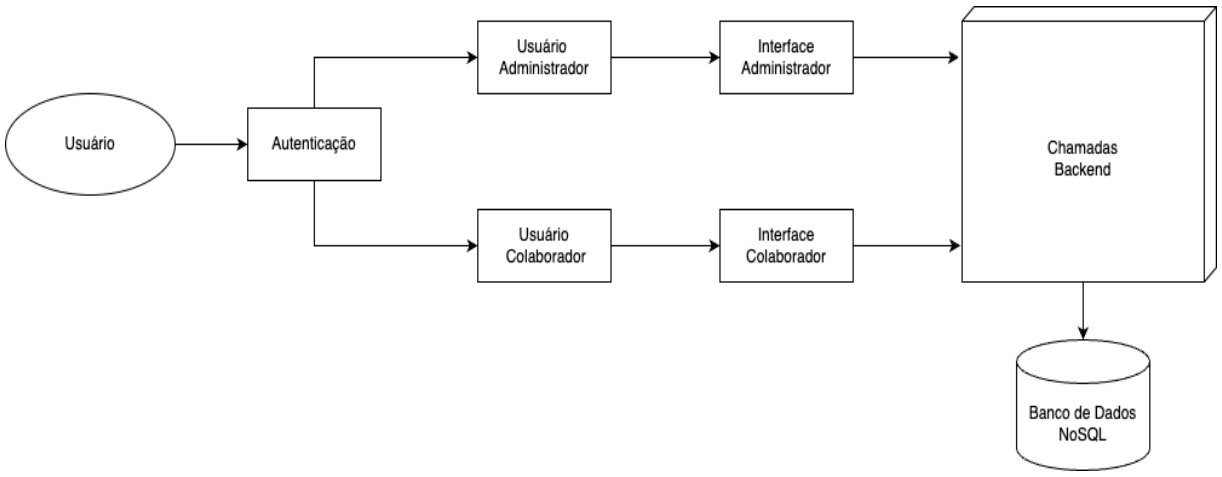
2 O PROJETO

O projeto contempla a criação de um aplicativo que permitirá que o usuário cadastre sua clínica e, dentro desse contexto, possa realizar as atividades básicas de gerenciamento de um estabelecimento, como cadastrar, editar, excluir colaboradores, realizar agendamentos, visualizar horários de atendimento e alterar horários de atendimento.

O aplicativo conta com uma visão customizada para o acesso do usuário administrador da clínica e para o usuário colaborador.

Para apresentar uma visão geral e abrangente da versão inicial do sistema, foi feito um diagrama de blocos, que segue abaixo. O diagrama também foi construído usando a ferramenta draw.io, disponível em https://app.diagrams.net/ .

Figura 4: Diagrama de Blocos



Fonte: O autor (2023).

3 TECNOLOGIAS ESCOLHIDAS

Como já descrito anteriormente, este trabalho possui como objetivo desenvolver um aplicativo que possa ser executado em plataformas Android e IOS. Tendo em vista esse objetivo, as tecnologias escolhidas para o desenvolvimento foram o *framework* Flutter, a linguagem de programação Dart e o serviço de *back-end* Firebase. Nesta seção, será descrito com maior profundidade as características de cada uma das tecnologias citadas.

3.1 DART

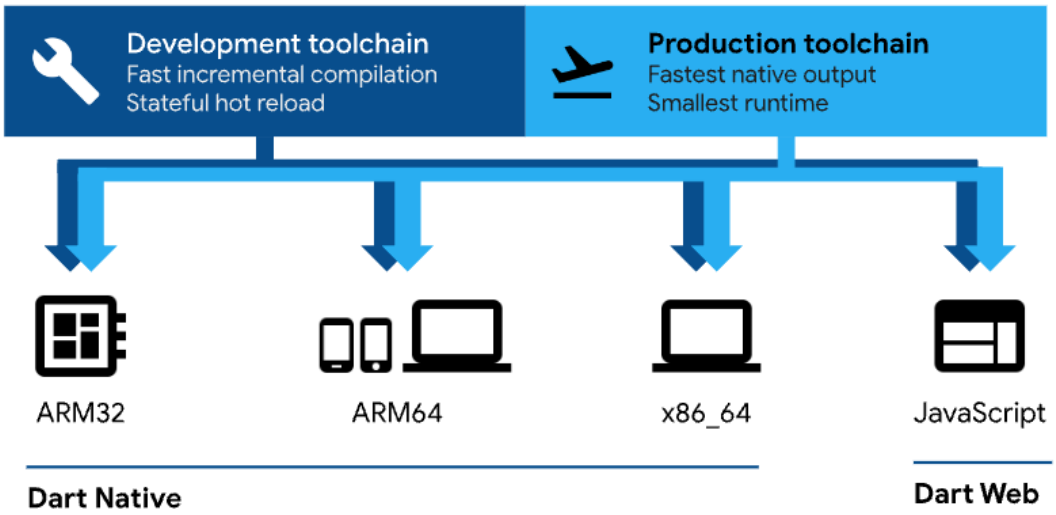
Para o desenvolvimento do aplicativo MyClinic App foi utilizada a linguagem de programação Dart. A linguagem Dart foi criada pela Google em 2011, com a promessa de desenvolver aplicativos rápidos em qualquer plataforma.

Inicialmente, a linguagem foi criada com o intuito de ser uma substituta para o javascript, mas não foi incorporada pela comunidade de desenvolvimento como era esperado. Recentemente, a linguagem voltou a ter espaço com o crescimento do Flutter na comunidade de desenvolvimento.

Dart é uma linguagem de programação orientada a objetos, multiparadigma, *type safe* e *null safe*, que impede que valores sejam nulos, a menos que seja declarado que determinado valor pode ser nulo. Tal funcionalidade impede exceções nulas no tempo de execução.

O Dart pode ser utilizado para desenvolvimento de aplicações *web*, *mobile*, *desktop* e *back-end*. Nesse contexto, o compilador Dart permite executar o código de duas formas diferentes. Em plataforma nativa, para aplicativos *mobile* e *desktop*, o Dart possui uma máquina virtual Dart, onde gera código de máquina através de uma compilação *just-in-time* (JIT) e *ahead-of-time* (AOT). Esse processo de compilação pode ser observado na ilustração abaixo.

Figura 5: Compilação código Dart em diferentes plataformas



Fonte: Dart (2023)

3.2 FLUTTER

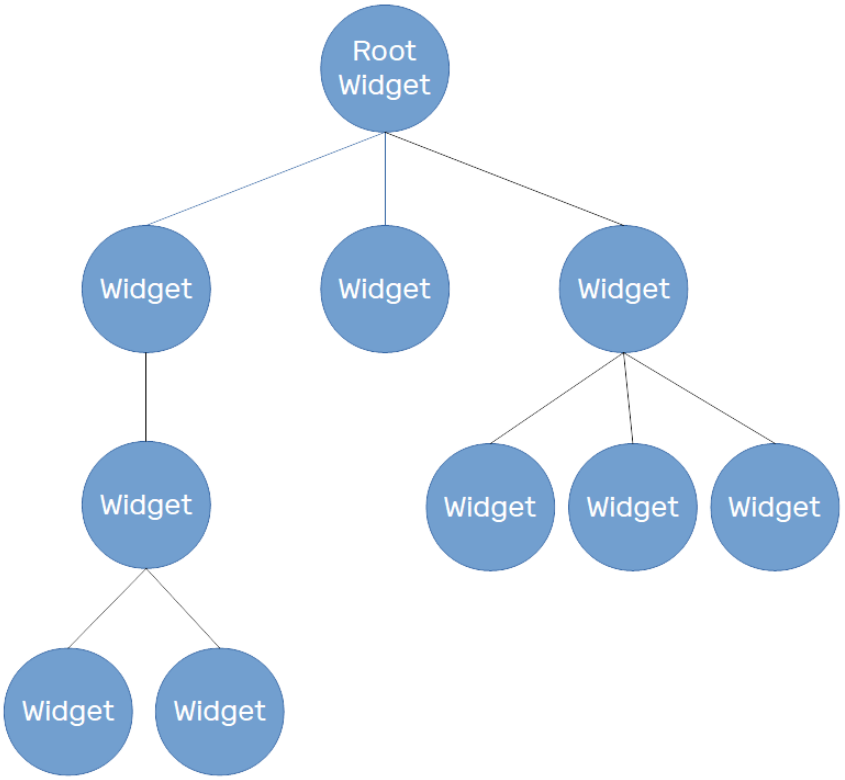
Para o desenvolvimento do *front-end* do aplicativo, será utilizado o Flutter. O Flutter é descrito como um conjunto de ferramentas para criação de interfaces de usuário criado pelo Google para desenvolvimento *mobile*, *web* e *desktop*, com capacidade para criar aplicações híbridas com performance de aplicações nativas.

Na fase de desenvolvimento, o aplicativo desenvolvido em Flutter é executado em uma máquina virtual que permite *hot reload* da aplicação, ou seja, recarregar a aplicação sem a necessidade de uma recompilação completa. Já para lançamento, o aplicativo Flutter é compilado para código de máquina, podendo ser esse Intel x64, ARM, ou JavaScript, em caso de aplicativos *web* (FLUTTER, 2022a).

O Flutter faz uso de *widgets* para o desenvolvimento da sua interface de usuário. Widgets são blocos de construção que compõem a interface que está sendo desenvolvida. O Flutter possui uma vasta lista de *widgets* que podem ser utilizados pelos desenvolvedores, disponível em https://docs.flutter.dev/ui/widgets .

No flutter, os *widgets* são estruturados em uma árvore de *widgets*. A árvore de *widgets* é uma representação lógica dos componentes de widgets que são exibidos para o usuário.

Figura 6: Árvore de *Widgets* do Flutter



Fonte: oreilly(2023)

3.3 FIREBASE

O Firebase é um produto do Google. Ele fornece aos desenvolvedores diversas ferramentas e serviços para auxiliar no processo de desenvolvimento.

Dentre os principais serviços do Firebase, estão o Firestore, Realtime Database, Cloud Storage, Authentication, Hosting, Remote Config, Notifications, Test Lab, Analytics, Dynamic Links e Performance Monitoring.

O uso de uma plataforma com o firebase traz diversos benefícios, dentre os quais pode-se citar:

- Dados em tempo real;
- API pronta;
- Autenticação via e-mail, Google, Facebook e Github;
- Segurança;
- Armazenamento de arquivos pelo Google Cloud Storage;
- Hospedagem de arquivos estáticos;

- Aplicativos altamente escaláveis;
- Sem preocupações quanto à infraestrutura.

Uma abordagem mais profunda, no que tange às funcionalidades utilizadas nesse projeto, será estabelecida no capítulo que abrange o desenvolvimento *back-end* da aplicação.

A documentação oficial do Firebase está disponível em https://firebase.google.com/docs?hl=pt-br (acesso em 10/11/2023).

4 ESPECIFICAÇÃO

Nesta seção será descrita a especificação do sistema do aplicativo. Para essa descrição serão utilizados conceitos de engenharia de software aplicados na criação de diagramas.

Para a especificação foram desenvolvidos, além da análise de requisitos, diagramas de casos de uso e diagramas de sequência Os diagramas e a análise de requisitos serão apresentados nas subseções 4.1, 4.2 e 4.3.

4.1 ANÁLISE DE REQUISITOS

O levantamento dos requisitos de *software* é uma das fases mais importantes do processo de desenvolvimento. É através do levantamento de requisitos que é possível entender e criar uma perspectiva do que a aplicação realmente fará. Estes requisitos podem ser classificados em Requisitos Funcionais e Requisitos Não Funcionais. Dessa forma, para esse trabalho foram elaborados dois quadros, que abrangem os requisitos funcionais e não funcionais, do aplicativo MyClinic App. Ambos os quadros estão dispostos nos capítulos 4.1.1 e 4.1.2, respectivamente.

4.1.1 REQUISITOS FUNCIONAIS

Segundo Pressman (2011), os requisitos funcionais de um sistema referem-se ao seu comportamento, ou seja, o que deve o referido sistema fazer e como deve se comportar em determinadas situações. No Quadro 1, estão dispostos os requisitos funcionais levantados para esta aplicação.

Quadro 1: Requisitos funcionais

Código	Identificação	Ator	Objetivo	Classificação
[RF001]	Cadastro de usuário Administrador	Usuário Administrador	Deve ser possível cadastrar nova conta de usuário administrador.	Essencial
[RF002]	Efetuar login por e-mail e senha	Usuário Administrador e Colaborador	O usuário deve poder acessar o sistema através de e-mail e senha.	Essencial
[RF003]	Alterar senha	Usuário Administrador e Colaborador	O usuário deve poder alterar sua senha.	Importante
[RF004]	Sair do aplicativo	Usuário Administrador e Colaborador	O usuário deve ter, de fácil acesso, um botão para sair do aplicativo	Essencial
[RF005]	Cadastrar clínica	Usuário Administrador	O usuário administrador deve poder cadastrar sua clínica no aplicativo.	Essencial
[RF006]	Cadastrar colaborador	Usuário Administrador	O usuário administrador deve poder cadastrar novos colaboradores para sua clínica.	Essencial
[RF007]	Editar colaborador	Usuário Administrador e Colaborador	O usuário deve ser capaz de editar o perfil de colaborador.	Essencial
[RF008]	Excluir colaborador	Usuário Administrador	O usuário administrador deve ser capaz de excluir o perfil de um colaborador	Essencial
[RF009]	Exibir os colaboradores da sua clínica	Sistema	O usuário administrador deve poder visualizar todos os colaboradores que	Essencial

			atendem na sua clínica.	
[RF010]	Acesso a um calendário	Sistema	O usuário deve ter acesso a um calendário interno do aplicativo.	Essencial
[RF011]	Fazer um agendamento	Usuário Administrador e Colaborador	O usuário deve conseguir selecionar e salvar uma data para agendamento	Essencial
[RF012]	Agenda	Usuário Administrador e Sistema (visualização)	O usuário administrador e colaborador deve poder visualizar sua agenda e a agenda dos colaboradores da sua clínica.	Essencial
[RF013]	Visualizar detalhes de agendamento	Usuário Administrador e Colaborador	O usuário deve ser capaz de ver detalhes de um agendamento selecionado	Essencial
[RF014]	Editar Agendamento	Usuário Administrador e Colaborador	O usuário deve ser capaz de editar as informações de um agendamento selecionado	Importante
[RF015]	Excluir agendamento	Usuário Administrador e Colaborador	O usuário deve ser capaz de excluir um agendamento selecionado	Essencial
[RF016]	Visualizar Histórico de agendamento	Sistema	O usuário deve ter acesso ao seu histórico de agendamentos cadastrados	Essencial
[RF017]	Inserir nota sobre agendamento cadastrado.	Usuário Administrador e Colaborador	O usuário deve ser capaz de inserir uma nota sobre um agendamento cadastrado.	Importante

[RF018]	Editar nota sobre agendamento cadastrado.	Usuário Administrador e Colaborador	O usuário deve ser capaz de editar as notas de agendamentos.	Importante
[RF019]	Deletar nota sobre agendamento cadastrado.	Usuário Administrador e Colaborador	O usuário deve ser capaz de deletar as notas de agendamentos.	Importante
[RF020]	Acesso a Câmera	Sistema	O aplicativo deve permitir que o usuário acesse a câmera através do aplicativo.	Importante
[RF021]	Cadastrar foto de perfil	Usuário Administrador e Colaborador	O usuário deve poder cadastrar uma foto de perfil	Importante
[RF022]	Alterar foto de perfil	Usuário Administrador e Colaborador	O usuário deve poder alterar sua foto de perfil	Importante

Fonte: O autor (2023).

4.1.2 REQUISITOS NÃO FUNCIONAIS

Os requisitos não funcionais desempenham papel crítico no desenvolvimento de *software*. São requisitos que abrangem questões como, custo, performance, confiabilidade, manutenibilidade e portabilidade. A seguir, no Quadro 2, é possível ver os requisitos não funcionais do aplicativo MyClinic App.

Quadro 2: Requisitos não funcionais

Código	Tipo	Descrição	Classificação
[RNF001]	Usabilidade	A aplicação deverá ser de simples acesso e navegação	Importante
[RNF002]	Disponibilidade	A utilização do aplicativo depende da conexão com a internet.	Importante

[RNF003]	Segurança	Os dados do usuário devem estar protegidos, e só devem ser acessados com autorização.	Essencial
[RNF004]	Disponibilidade	Ao efetuar um agendamento, o banco de dados deve ser atualizado e sincronizado instantaneamente.	Essencial
[RNF005]	Compatibilidade	Deve ser possível executar a aplicação em todas as plataformas suportadas (Android e IOS).	Essencial
[RNF006]	Usabilidade	A aplicação deve oferecer experiência de usuário semelhante em ambas as plataformas Android e IOS.	Essencial
[RNF007]	Desenvolvimento	A aplicação deve ser desenvolvida com as seguintes tecnologias: Flutter, Dart e Firebase	Importante

Fonte: O autor (2023).

4.2 DIAGRAMAS DE CASOS DE USO

Os Diagramas de Casos de Uso fornecem um modo de descrever a visão externa do sistema e suas interações com o mundo exterior. Assim, trazem uma visão de alto nível das funcionalidades intencionais mediante solicitações efetuadas pelo usuário. Estes diagramas apresentam uma visão externa sobre como esses elementos podem ser utilizados no contexto do sistema sendo representado (FURLAN, 1998).

Para Bezerra (2006, p. 45) modelo de casos de uso podem ser definidos como “uma representação das funcionalidades externamente observáveis do sistema e dos elementos externos ao sistema que interagem com ele.”

Para este MyClinic App, foram desenvolvidos diagramas de Caso de Uso. Neste caso, foram desenvolvidos diagramas que contemplam os fluxos de autenticação, os de usuário administrador e usuário colaborador. Eles podem ser vistos a seguir, nas Figuras 7, 8, 9, e 10.

4.2.1 DIAGRAMAS DE CASOS DE USO DE AUTENTICAÇÃO

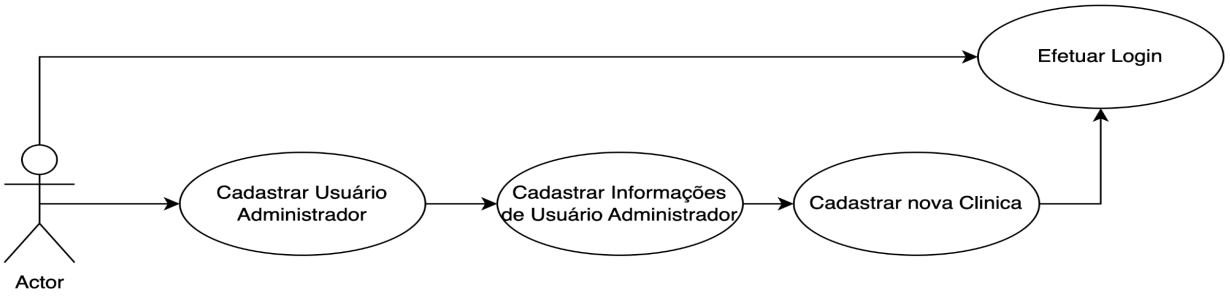
O fluxo de autenticação do usuário está representado na figura 7, onde estão dispostas as opções de entrada do aplicativo MyClinic App.

Nela o usuário pode criar uma nova conta de administrador da clínica. A conta de colaborador é criada somente dentro do aplicativo, mais especificamente, dentro da interface do usuário administrador. Somente o usuário administrador pode criar uma nova conta de colaborador. Dessa forma o colaborador já é criado vinculado a clínica.

O processo de criação de nova conta de usuário administrador foi elaborado para ser o mais simples e direto possível. Assim, inicialmente, é exibida uma tela de cadastro de novo usuário administrador e em seguida uma tela de cadastro de nova clínica.

Quando é feito o login, a aplicação identifica o perfil do usuário, podendo ser esse administrador ou colaborador e o direciona para uma versão customizada criada para cada tipo de usuário.

Figura 7: Diagrama de Caso de Uso Autenticação

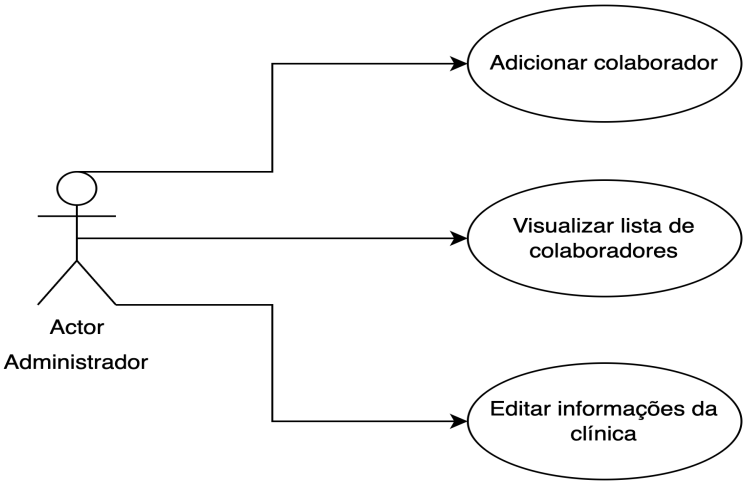


Fonte: O autor (2023).

4.2.2 DIAGRAMAS DE CASOS DE USO DE USUÁRIO ADMINISTRADOR

A figura 8 representa a visão geral que o usuário administrador tem do aplicativo assim que é feito o login. Nessa primeira tela, o usuário visualiza uma lista com os colaboradores cadastrados em sua clínica com a opção de adicionar novo colaborador e editar informações da clínica.

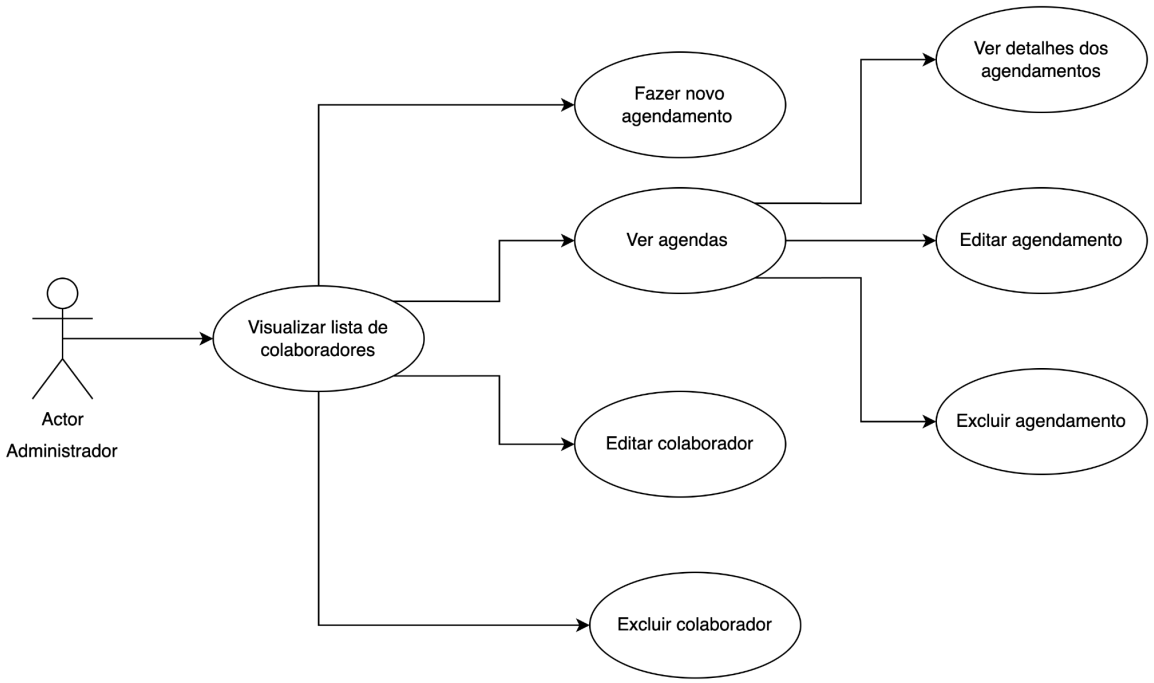
Figura 8: Diagrama de Caso de Uso Administrador - Visão Geral



Fonte: O autor (2023).

Na tela inicial o usuário administrador pode interagir com a sua lista de colaboradores. Nela, como pode ser visto na Figura 9, o usuário pode fazer um novo agendamento para si próprio ou para algum colaborador, pode ver o agendamento realizados, realizar operações como, editar o um agendamento, excluir um agendamento, editar um colaborador e excluir um colaborador.

Figura 9: Diagrama de Caso de Uso Administrador - Ações

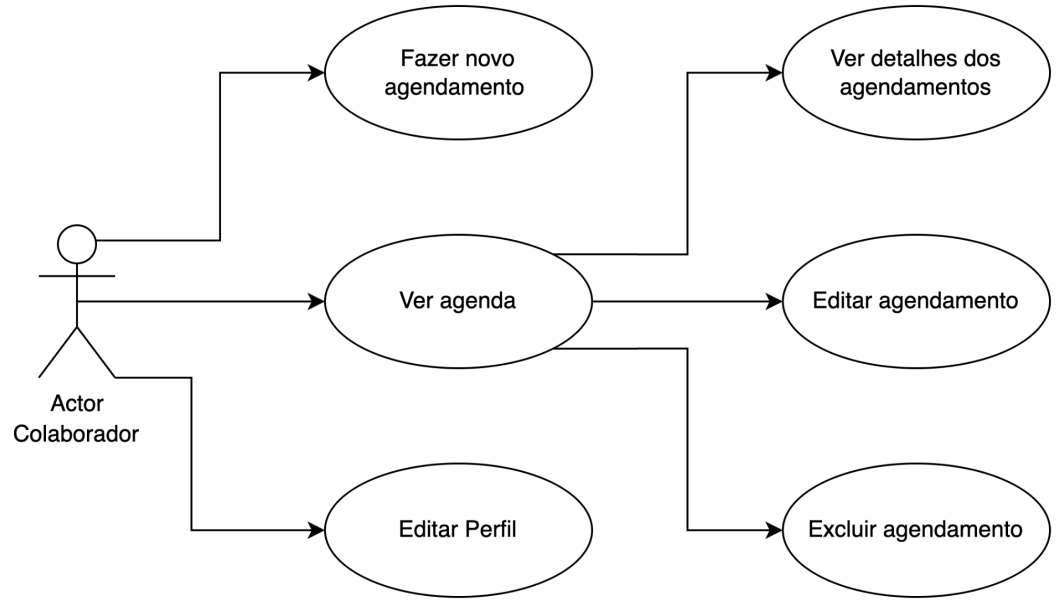


Fonte: O autor (2023).

4.2.1 DIAGRAMAS DE CASOS DE USO DE USUÁRIO COLABORADOR

Por fim, existe uma visão do aplicativo pela perspectiva do colaborador. Quando o colaborador faz login e acessa o aplicativo ele pode ver somente as suas informações de forma customizada. Nesse cenário, o colaborador pode ver sua agenda pessoal, cadastrar um novo agendamento, editar um agendamento existente, excluir um agendamento e editar seu perfil pessoal.

Figura 10: Diagrama de Caso de Uso Colaborador



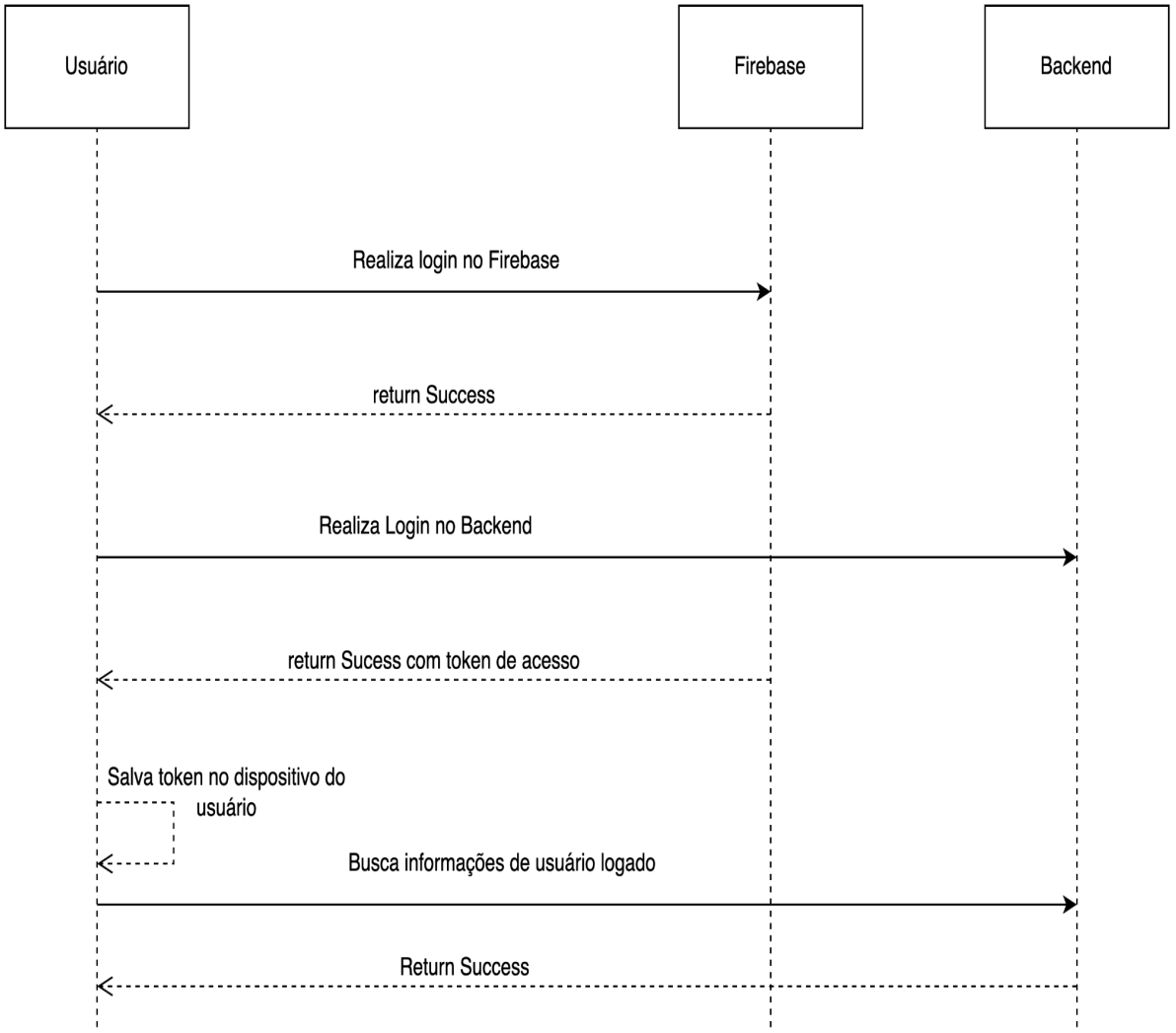
Fonte: O autor (2023).

4.3 DIAGRAMAS DE SEQUÊNCIA

O diagrama de sequência dá ênfase à ordenação temporal em que as mensagens são trocadas entre os objetos de um sistema. Este diagrama é construído a partir do Diagrama de Casos de Usos. Primeiro, define-se qual o papel do sistema (*Use Cases*), depois, é definido como o *software* realizará seu papel (Sequência de operações).

A figura 11 representa um diagrama de sequência para o processo de *login*. Nele é feito o processo de login no firebase, seguido pela autenticação no *back-end* e a geração do *token* de acesso do usuário.

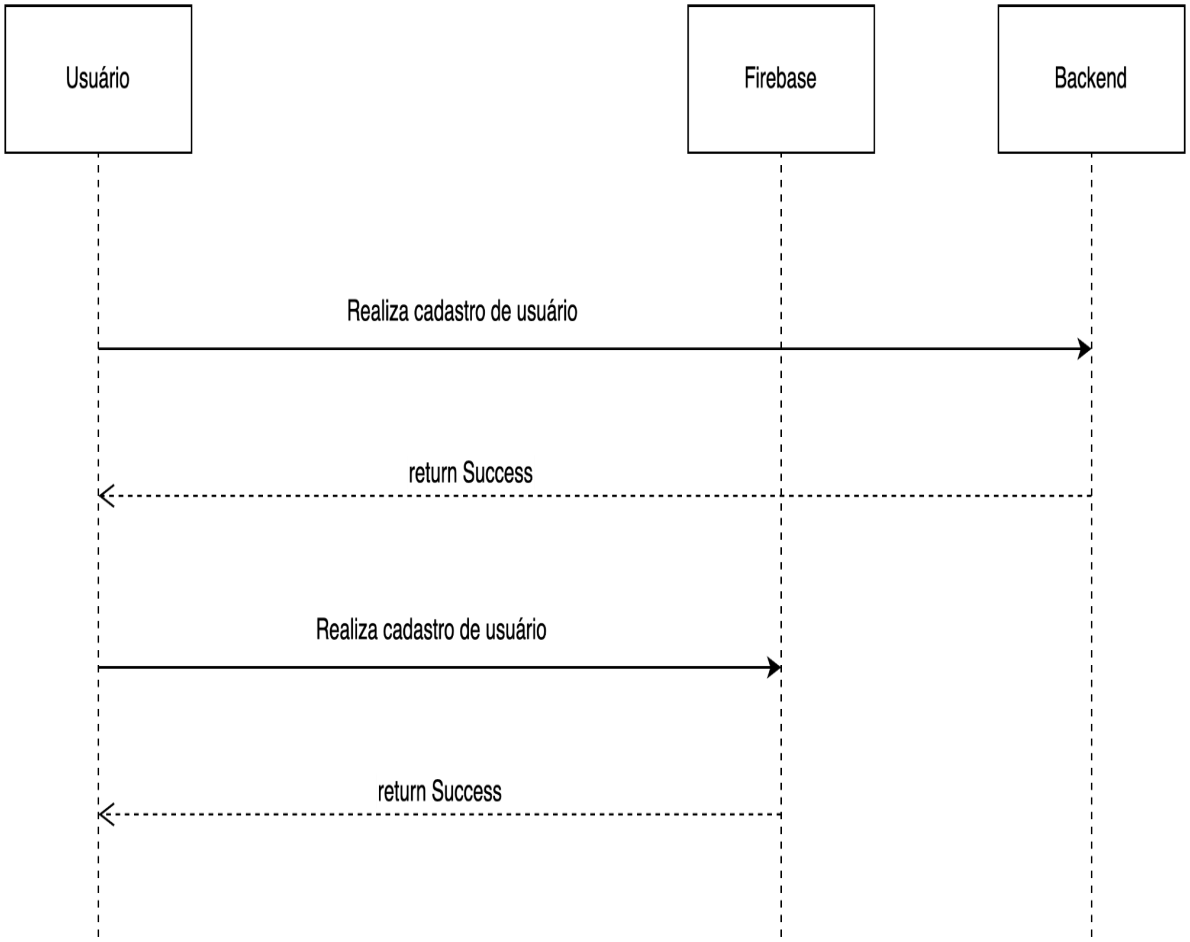
Figura 11: Diagrama de Sequência - Login



Fonte: O autor (2023).

Já a figura 12, representa um diagrama de sequência do fluxo de cadastro de usuário. No cadastro de novo usuário, inicialmente é realizado o cadastro no servidor *back-end*. Se o cadastro no servidor for concluído com sucesso, o usuário é então cadastrado no firebase.

Figura 12: Diagrama de Sequência - Cadastro de Usuário



Fonte: O autor (2023).

5 DESENVOLVIMENTO DO PROJETO BACK-END

Para construção de parte da infraestrutura do *back-end*, será utilizado um *BaaS*, que significa *back-end as a Service*. No caso, como citado anteriormente, foi escolhido o Firebase.

Um *BaaS* é um serviço de computação em nuvem que tem como objetivo automatizar o desenvolvimento do *back-end.* O *BaaS* permite abstrair do desenvolvimento tarefas consideradas repetitivas para que seja possível focar nas partes centrais da aplicação.

A escolha de utilizar parte dos serviços do Firebase foi feita pensando no desenvolvimento de novas funcionalidades para o futuro do projeto, que serão abordadas no capítulo 8 deste documento.

5.1 FIREBASE AUTHENTICATION

O processo de autenticação de usuário é extremamente delicado. Por esse motivo foi escolhido utilizar uma dupla camada de autenticação, incluindo serviços de criação de novo usuário administrador e colaborador e para login destes usuários.

Dessa forma, todos os usuários criados estarão salvos no servidor *back-end* e no Firebase. Assim como todos os logins realizados terão essa dupla validação.

5.2 SERVIDOR BACK-END

O json_rest_server é um pacote da linguagem de programação Dart que disponibiliza um servidor *RESTful* baseado em *JSON* criado para desenvolvedores Flutter criarem projetos pessoais. Após ser instalado ele atua como um *software* executável dentro da máquina do desenvolvedor.

Esse pacote gera um arquivo com de uma árvore *json*, onde o desenvolvedor pode configurar os listas de objetos json que armazenam os dados da aplicação. A figura 13 exemplifica o formato que os dados são armazenados.

Figura 13: Exemplo de árvore *JSON* para armazenamento de dados.

```
{
    "products": [
        {
            "id": 0,
            "title": "Academia do flutter"
        },
        {
            "id": 1,
            "title": "Jornada Dart"
        },
        {
            "id": 2,
            "title": "Jornada GetX"
        }
    ]
}
```

Fonte: Json_rest_server(2023)

A partir da nomeação dessas listas, as rotas da aplicação são geradas pelo pacote. Na figura 14, vê-se um exemplo das rotas geradas para "*products*", da figura 13.

Figura 14: Exemplo de rotas criadas pelo json_rest_server.

```
GET     /products
GET     /products?title=jornada
GET     /products?page=1&limit=10
GET     /products?page=1&limit=10&title=Jornada
GET     /products/1
POST    /products
PUT     /products/1
PATCH   /products/1
DELETE  /products/1
```

Fonte: Json_rest_server(2023)

O json_rest_server também disponibiliza uma configuração de autenticação, que gera um *token* de autenticação para ser passado nas chamadas ao *back-end*, de forma a trazer mais segurança para as chamadas realizadas ao servidor.

A documentação completa do json_rest_server pode ser encontrada através link: https://pub.dev/packages/json_rest_server (acesso em 10/11/2023).

No apêndice 1 deste documento está disponível um quadro com todas as rotas criadas e utilizadas no desenvolvimento, junto com um exemplo de chamada.

6 DESENVOLVIMENTO DO PROJETO FRONT-END

Conforme descrito no Capítulo 3 desse documento, as tecnologias escolhidas para o desenvolvimento do *front-end* do aplicativo MyClinic App foram as linguagens de programação Dart e o *framework* Flutter.

6.1 PACOTES DART/FLUTTER

O ecossistema de desenvolvimento Flutter e Dart fornece para os desenvolvedores a possibilidade do uso de pacotes, que podem facilitar etapas do processo de desenvolvimento ou fornecer funcionalidades prontas.

Esses pacotes são criados por desenvolvedores e publicados e podem ser adquiridos em um repositório de pacotes Flutter e Dart, o pub.dev, disponível em https://pub.dev/ .

Os pacotes utilizados nesse projeto estão dispostos a seguir, com a sua respectiva versão:

1) cupertino_icons (versão 1.0.2): Utilizado para obtenção de ícones do tipo cupertino;

2) flutter_riverpod (versão 2.4.5): Utilizado para o gerenciamento de estado da aplicação;

3) asyncstate (versão 2.0.2): Utilizado para sinalizar carregamento de telas para o usuário durante chamadas assíncronas;

4) loading_animation_widget (versão 1.2.0+4): Utilizado para exibir animação de carregamento na tela do usuário;

5) dio (versão 5.3.3): Utilizado para requisições *http*;

6) shared_preferences (versão 2.2.2): Utilizado para armazenamento de informações simples de preferências do usuário no dispositivo;

7) validatorless (versão 1.2.3): Utilizado para validações de campos de formulários.

8) top_snackbar_flutter (versão 3.1.0): Utilizado para exibição de caixas de diálogo na tela do usuário;

9) table_calendar (versão 3.0.9): Utilizado para exibição de calendário simples na tela do usuário;

10) intl (versão 0.18.1): Utilizado para ajuste do padrão de data e hora dentro da aplicação;

11) firebase_core (versão 2.22.0): Utilizado para integração com o Firebase;

12)firebase_auth (versão 4.14.0): Utilizado para autenticação de usuário via Firebase Authentication;

13) uuid (versão 3.0.7): Utilizado para geração de identificadores únicos;

14)syncfusion_flutter_calendar (versão 23.1.44): Utilizado para exibição de calendário no formato de agenda;

6.2 ARQUITETURA FRONT-END

Para o desenvolvimento do *front-end* do aplicativo MyClinic App, foi escolhido o padrão de arquitetura MVVM (*Model-View-ViewModel*). O padrão de arquitetura MVVM foi escolhido devido a sua alta compatibilidade com desenvolvimento de dispositivos móveis, mais especificamente com Flutter, onde o desenvolvimento é direcionado para a interface de usuário, a *View.*

A arquitetura MVVM possui como grande destaque a característica de separar a interface de usuário da lógica de negócios da aplicação. Assim, é assegurado que o processo de desenvolvimento irá gerar um código de fácil manutenibilidade e escalabilidade.

Esse padrão consiste na separação da aplicação em três camadas, a *Model*, a *View* e a *ViewModel*. A camada da *View*, como já citado, é responsável pela interface de usuário. De forma mais genérica, o que é exibido nas telas dos

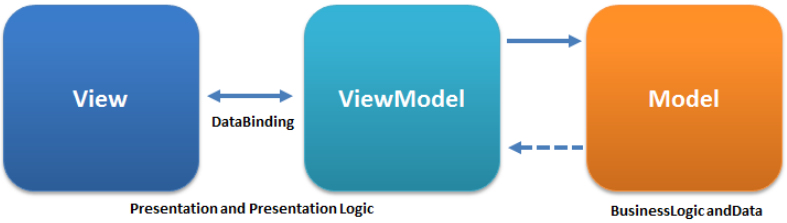
dispositivos. Nela são implementados os *Widgets* do Flutter, devendo ser desenvolvida de forma simples e restrita a elementos da interface de usuário.

A *Model*, por sua vez, é encarregada de armazenar a lógica de negócios do aplicativo. Ela encapsula os objetos de dados e apresenta métodos para manipular esses dados. Isto é, a *Model* é responsável por buscar os dados no *back-end*.

Por fim, há a *ViewModel*. A *ViewModel* age como intermediária entre a *View* e a *Model*. É uma camada que possui o papel de comunicar a *Model* das interações de usuário e entregar os dados atualizados para a *View*. Ou seja, ela busca na *Model* os dados solicitados pela *View* e os entrega atualizados para a *View*.

É imprescindível ressaltar a característica da *ViewModel* de ser vinculadora de dados. Isso significa que ela deve permitir uma sincronização automática com a *View*. Dessa forma, é uma camada que pode ser implementada utilizando um gerenciador de estado.

Figura 15: Padrão de arquitetura MVVM.



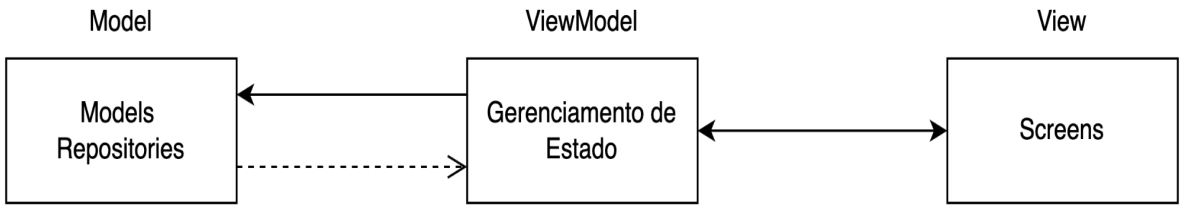
Fonte: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel.

6.2.2 ARQUITETURA MVVM APLICADA NESSE PROJETO

De forma geral, a arquitetura MVVM foi aplicada neste projeto na sua abordagem tradicional. Onde a *View* guardou os layouts das telas, a *Model* ficou encarregada de fazer a busca dos dados solicitados no *back-end* e a *ViewModel* do gerenciamento de estado da aplicação, informando a *Model* das solicitações de alterações de dados da *View* e retornando esses dados atualizados para a View.

A estrutura de uso do padrão *MVVM* utilizada neste projeto pode ser observada na Figura 16, apresentada abaixo.

Figura 16: Padrão de arquitetura MVVM aplicado ao projeto



Fonte: O Autor (2023).

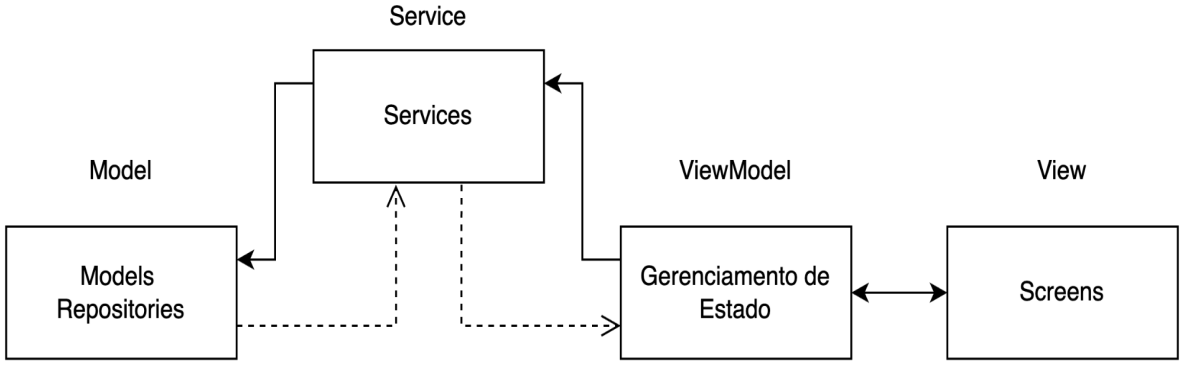
A exceção, ocorre na situação de cadastro de usuário administrador e login de ambos os tipos de usuário. Como pode ser observado no capítulo 4.3 deste trabalho, os processos de login e cadastro de novo usuário possuem dupla camada no *back-end*, sendo feito cadastro e verificação em ambos servidor *RESTful* e Firebase. Assim, para manter uma arquitetura limpa, com separação de responsabilidades, foi utilizado uma adaptação do modelo MVVM. Nos cenários de cadastro e login, foi utilizado uma arquitetura MVVM + camada de *Service*.

Nesse modelo, no registro de usuário administrador, a camada de *service* possui a finalidade de após realizado a tentativa de cadastro no servidor e no Firebase, executar o login desse novo usuário de forma automática e direcionar o usuário administrador para a sua tela inicial em caso de sucesso no cadastro. Caso o cadastro não seja executado de forma automática, é disparada uma exceção.

No processo de login, a camada de *service* é responsável por armazenar no dispositivo do usuário um *token* de acesso após o login ser realizado com sucesso. Esse *token* é utilizado como registro de login de usuário e é deletado quando o usuário faz *logout* da aplicação. Portanto, enquanto o usuário não realizar o *logout* do aplicativo, a existência do *token* irá garantir que mesmo que o usuário feche o aplicativo, ele não precise refazer o login quando abri-lo novamente.

A implementação dessa adaptação de arquitetura descrita acima, está exemplificada na Figura 17.

Figura 17: Padrão de arquitetura MVVM + Service aplicado ao projeto



Fonte: O Autor (2023).

6.3 GERENCIADOR DE ESTADO

O gerenciamento de estado possui grande importância dentro do desenvolvimento com Flutter. Dentro de um aplicativo, um estado representa um dado que pode ter alteração de valor.

Em um projeto de *software* pode ser necessário compartilhar informações entre telas e propagar uma alteração de valor dentro do aplicativo. Nesse cenário entra o gerenciamento de estado. O gerenciador de estado observa se houve alguma mudança na aplicação e reconstrói a interface para o usuário.

Existem diversos gerenciadores de estado dentro do Flutter. Para esse projeto foi utilizado o Riverpod.

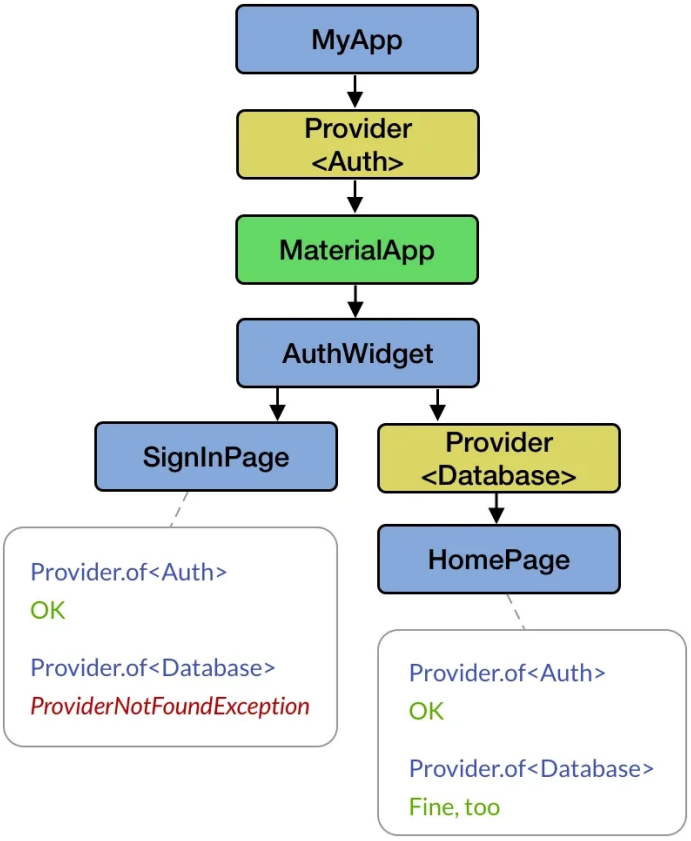
6.3.1 RIVERPOD

O Riverpod foi concebido como uma evolução natural de outro gerenciador de estado muito utilizado no desenvolvimento com Flutter, o Provider. Visto que ambos foram criado pela mesma pessoa, o desenvolvedor Remi Rousselet e que a palavra Riverpod é um anagrama para a palavra Provider.

Remi Rousselet criou o Riverpod pois havia uma relevante limitação no Provider. Considerando que o Flutter é um *framework* que estrutura seus *widgets* em uma árvore de *widgets*, o Provider somente consegue compartilhar e atualizar o estado da aplicação entre os filhos do *widget* que detém o Provider, não sendo possível compartilhar o estados com os *widgets* irmãos da árvore. Uma ilustração de um erro decorrente dessa limitação pode ser vista na figura 18.

O Riverpod soluciona esse problema. Dessa forma, o estado fica disponível para toda a aplicação, isto é, fica disponível de forma global na aplicação. No Riverpod pode-se obter um estado, ouvir um estado, observar um estado.

Figura 18: Ilustração do fluxo de execução do Provider.



Fonte: CAP SISTEMA (2023)

7 CONCLUSÃO

O aplicativo MyClinic App foi um projeto realizado como Trabalho de Conclusão de Curso do curso bacharel em Sistemas de Informação. Como tal, abrangeu não apenas os aspectos tecnológicos, mas também toda a estrutura organizacional por trás de um sistema a ser utilizado por empresas.

A ideia do projeto, portanto, não é apenas apresentar um produto tecnológico, mas também resolver uma problemática que envolve a organização de informações dentro de uma instituição. Com isso, foi possível utilizar tanto conceitos e práticas da computação, quanto da área administrativa e organizacional, outro foco desse curso.

Partindo desse pressuposto, o aplicativo MyClinic App busca contribuir com a organização interna de clínicas de saúde, focando principalmente naquelas de saúde mental. Vale ressaltar que foi idealizado para clínicas de pequeno e médio porte, que necessitam de funcionalidades organizacionais, mas sem grandes complexidades.

Pensando nisso, foi feita a análise de concorrentes para entender as funcionalidades mais comuns. Depois, foi preciso mapear os dispositivos utilizados, assim como as tecnologias. Também foram considerados os conceitos de experiência do usuário, para melhorar a usabilidade.

A partir dessas análises, foram tomadas as decisões de criar um aplicativo mobile, multiplataforma, que seja simples, intuitivo e com as principais funcionalidades necessárias, sem aumentar a complexidade para o usuário. Também foi mapeada a arquitetura do sistema, para entender quais linguagens, tecnologias e ferramentas mais se adequam às necessidades.

Como resultado, temos o aplicativo MyClinic App, feito com as tecnologias Dart, Flutter e Firebase, que atende aos requisitos descritos acima, focando na organização estrutural da clínica e em como a tecnologia pode contribuir com isso.

Dessa forma, conclui-se que o trabalho foi realizado com sucesso dentro das possibilidades dispostas, utilizando as boas práticas de desenvolvimento e considerando melhorias que ainda podem ser feitas em processos de integração e entrega contínuas, a fim de adicionar e aprimorar as funcionalidades propostas.

8 TRABALHOS FUTUROS

Considerando a possibilidade de avanço do projeto e pelo entendimento de que o aplicativo pode passar por processo de melhoria e de lançamentos de novas versões, foi possível considerar algumas funcionalidades futuras que poderão ser adicionadas.

Uma delas está relacionada com o registro de usuário. A ideia é realizá-lo exclusivamente pelo Firebase, o que permite que sejam enviadas notificações para o usuário, salvar fotos e documentos no serviço de *Cloud Storage*. Para isso, também será feita a migração do banco de dados para o *Cloud Firestore*.

Para finalizar, além dessa mudança para aprimorar o uso do aplicativo, outra funcionalidade útil para o usuário seria a criação de um aplicativo *web*, principalmente para os administradores das clínicas, que podem optar por usar uma versão em dispositivo *desktop*.

REFERÊNCIAS

ALMEIDA, Jhoisnáyra Vitória Rodrigues de. O que é gerenciamento de estados no Flutter e principais ferramentas. Plataforma educacional Alura. Acesso em: 09/2023 Link de acesso: https://tinyurl.com/3bbuxrjj

BEZERRA, E. Princípios de Análise e Projeto de Sistemas UML. 2 ed. Ed. Campus, Rio de Janeiro: Elsevier, 2006.

BIESSEK, Alessandro. Flutter for Beginners. Site O'reilly. Acesso em: 10/08/2023. Link de acesso: https://www.oreilly.com/library/view/flutter-for-beginners/9781788996082/69b9f485-e0f4-4776-b1b4-5824bec909f6.xhtml

CAP SISTEMA. Flutter Riverpod 2.x: O Guia Definitivo. Acesso. Cap Sistema. Acesso em: 05/10/2023. Link de acesso: https://capsistema.com.br/index.php/2023/08/09/flutter-riverpod-2-x-o-guia-definitivo/

DART. Dart overview. 2022a. <https://dart.dev/overview>,. Acesso em 26/09/2023.

Firebase Documentation. firebase.google. Link de acesso: https://firebase.google.com/docs/firestore?hl=pt-br . Acesso em: 26/09/2023.

FURLAN, J. D. Modelagem de objetos através da UML-the unified modeling language. [S.l.]: Makron books, 1998.

FLUTTER. Flutter architectural overview. 2022b. <https://docs.flutter.dev/resources/architectural-overview>,. Acesso em 26/09/ 2023.

Json_rest_server. Pubdev. Acesso em 10/11/2023. Link de acesso: https://pub.dev/packages/json_rest_server

LEVI, R. Simplificando o desenvolvimento de aplicativos com o Flutter em arquitetura MVVM. Medium. Acesso em 10/09/2023. Link de acesso:

https://medium.com/@rafaellevissa/simplificando-o-desenvolvimento-de-aplicativos-com-o-flutter-em-arquitetura-mvvm-f4727bedff16

Model–view–viewmodel. Em Wikipedia. Acesso em 10/09/2023. Link de acesso: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel.

PRESSMAN, Roger S. Engenharia de Software uma abordagem profissional, 7.ed. AMGH Editora Ltda, 2011.

SCHRAN, Letícia da Silva. MACHINESKI, Gicelle Galvan. RIZZOTTO, Maria Lúcia Frizon. CALDEIRA, Sebastião. Percepção da equipe multidisciplinar sobre a estrutura dos serviços de saúde mental: estudo fenomenológico. Universidade Estadual do Oeste do Paraná (UNIOESTE). Rev. Gaúcha Enferm. 40. 2019. Disponível em: https://doi.org/10.1590/1983-1447.2019.20180151 . Acesso em: 10/11/2023

APÊNDICES

APÊNDICE 1

Quadro 3 - RELAÇÃO DE ENDPOINTS UTILIZADOS NO BACK-END

	Funcionalidade	**Método Http**	**Url do Endpoint gerado**
1	Cadastro de usuário ADM	POST	http://192.168.1.184:8080/users
2	Buscar dados de usuário logado	GET	http://192.168.1.184:8080/me
3	Login de usuário	POST	http://192.168.0.12:8080/auth
4	Buscar colaboradores da clínica	GET	http://192.168.1.184:8080/users?place_id=3
5	Excluir usuário	DELETE	http://192.168.1.184:8080/users/id
6	Editar usuário	PUT	http://192.168.1.184:8080/users/id
7	Registrar clínica	POST	http://192.168.1.184:8080/place
8	Buscar informações da clínica que usuário está cadastrado.	GET	http://192.168.0.12:8080/place?user_id=%21userAuthRef
9	Buscar agendamentos de colaborador	GET	http://192.168.1.184:8080/schedules?user_id=18&date=2023-11-08T00:00:00
10	Excluir agendamento	DELETE	http://192.168.1.184:8080/schedules/id
11	Buscar todos os agendamentos	GET	http://192.168.1.184:8080/schedules
12	Criar novo agendamento	POST	http://192.168.1.184:8080/schedules
13	Editar agendamento	PUT	http://192.168.1.184:8080/schedules/id

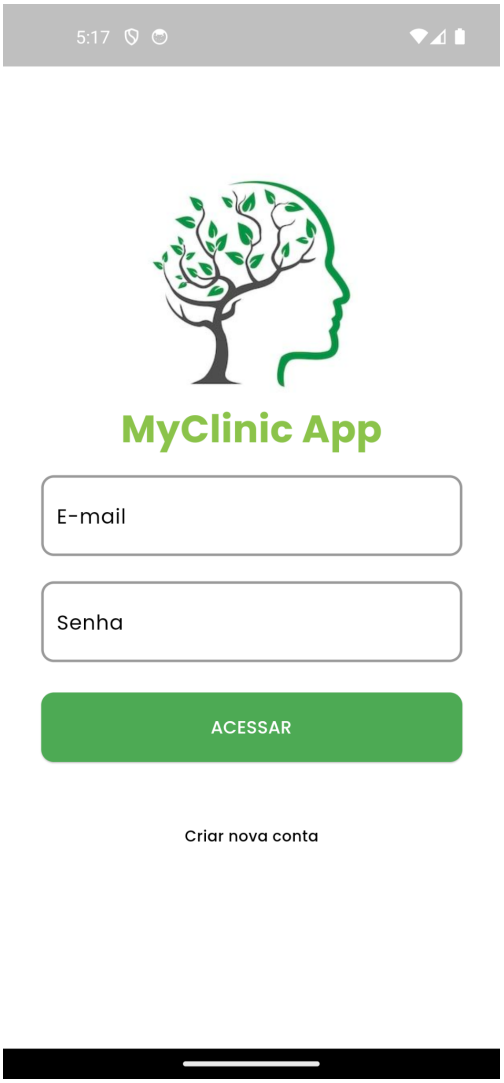
APÊNDICE 2 - QUADRO COMPARATIVO ENTRE APLICAÇÕES

Quadro 4 - QUADRO COMPARATIVO ENTRE APLICAÇÕES

Funcionalidade	**MyClinic App**	**Agendart**	**PsicoPlanner**	**PsicoManager**
Versão Web	NÃO	SIM	SIM	SIM
Integração com Whatsapp	NÃO	SIM	SIM	SIM
Exibir lista de colaboradores cadastrados	SIM	SIM	NÃO	NÃO
Cadastrar colaborador na clínica	SIM	SIM	NÃO	NÃO
Cadastrar agendamento	SIM	SIM	SIM	SIM
Visualizar agendamentos em agenda	SIM	SIM	SIM	SIM
Inserir foto de perfil	SIM	SIM	SIM	SIM
Inserir nota sobre agendamento	SIM	SIM	SIM	SIM
Sala virtual	NÃO	SIM	SIM	SIM
Visualizar histórico de agendamentos	SIM	SIM	SIM	SIM
Acesso a notificações	NÃO	SIM	SIM	SIM
Controle Financeiro	NÃO	SIM	SIM	SIM

APÊNDICE 3 - MANUAL DO USUÁRIO

1. Tela Login

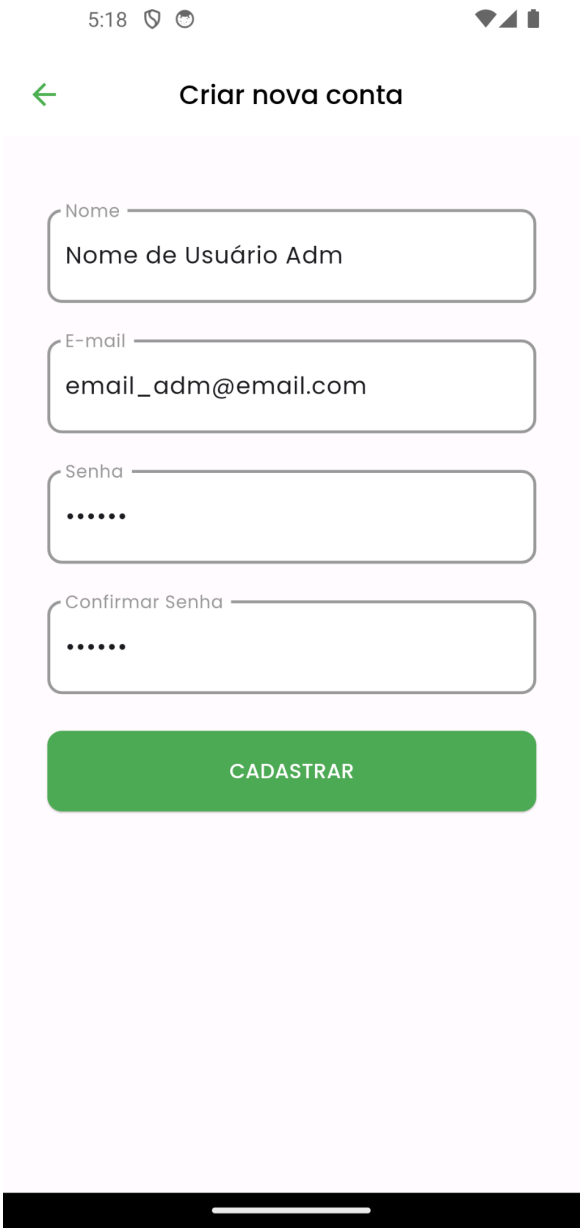


A tela de login parece assim que o aplicativo é aberto, caso o usuário não tenha feito nenhum login.

Na tela de login temos o logo e nome do aplicativo. Abaixo, os campos de formulário de e-mail e senha, caso a pessoa já tenha feito o cadastro. Após cadastro validado, o usuário acessa a tela 4.

Caso não tenha cadastro, o usuário acessa “Criar uma nova conta”.

2. Tela de Cadastro de Novo Usuário



Na tela de cadastro de novo usuário temos os campos de formulário para preenchimento dos dados.

Campos solicitados: nome, e-mail, senha, confirmação da senha.

O registro é feito no banco após o clique no botão “Cadastrar”.

3. Tela de Nova Clínica

Após criar uma nova conta, o usuário irá cadastrar sua clínica. Todas as outras funcionalidades (como cadastro de profissionais e agenda) dependem do cadastro da clínica para serem feitos, já que estão relacionados com os dados da clínica.

Na tela, temos os campos de formulário de preenchimento livre: nome e e-mail da clínica. Também há as opções selecionáveis: dias da semana em que a clínica está aberta e quais são os horários em que a clínica funciona.

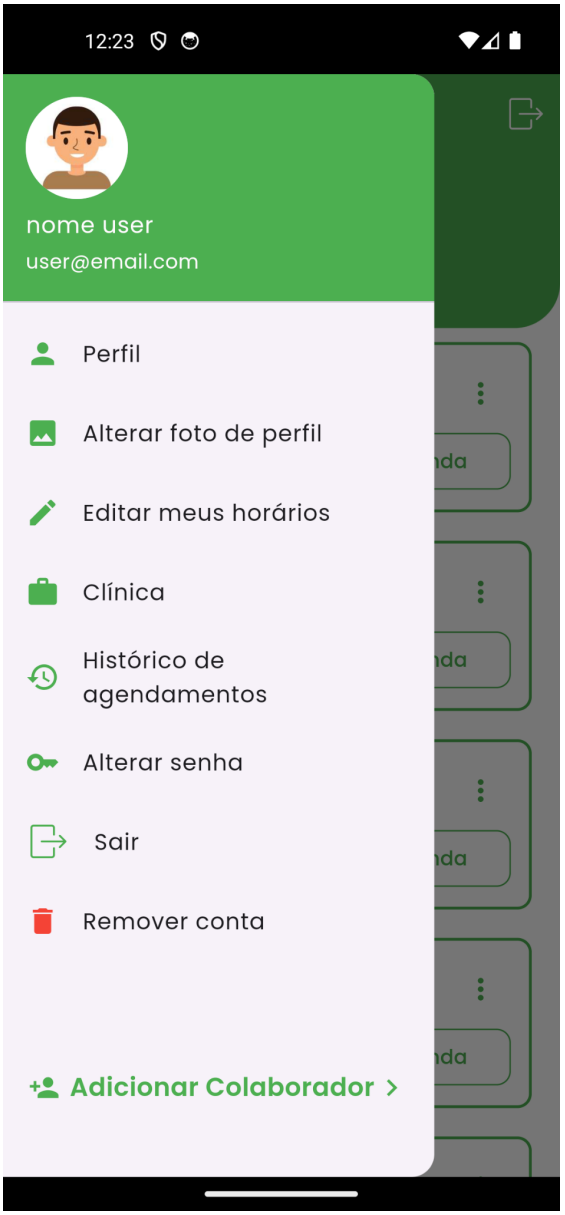
Esse preenchimento é essencial para, depois, realizar o cadastro dos parceiros, pois só é liberado o cadastro dos horários de cada um de acordo com o que foi cadastrado como padrão da clínica. Ou seja, só podem agendar consultas dentro do horário de funcionamento determinado nessa etapa.

4. Tela Inicial Inicial

A tela inicial logada é visualizada após o login inicial do aplicativo (tela 1) ou após a finalização do cadastro de novo usuário e nova clínica (tela 2 a 3).

Nesta tela temos o nome da clínica que está sendo gerenciada. Na parte principal, os colaboradores e parceiros cadastrados pelo administrador, assim como os botões para agendamentos de cada um deles e o botão para visualização da agenda de cada um.

4.1 Menu de ações



No menu, o usuário pode executar uma lista de ações, incluindo ver suas informações de perfil, alterar sua foto de perfil, editar seus horários de atendimento, ver as informações da clínica em que trabalha, ver seu histórico de agendamentos, alterar sua senha, sair da aplicação, deletar sua conta e, por fim, também há o botão para adicionar novo colaborador.

5. Tela de Cadastro de Novo Colaborador

Após clicar no botão de adicionar novo colaborador, o usuário administrador visualiza a tela de cadastro em que pode colocar um cadastro próprio, para também ter sua agenda mapeada pelo aplicativo, ou cadastrar um terceiro.

Para cadastro de terceiros, temos os campos: nome, e-mail, seleção dos dias e horários de atendimento desse colaborador. Botão “cadastrar”.

5.1 Caso o administrador queira se cadastrar como colaborador

Se o administrador também quiser se cadastrar como colaborador, ao selecionar o *checkbox*, os campos de nome e e-mail são fechados, já que essas informações já estão cadastradas.

Assim, ficam liberados apenas os campos de seleção dos dias e horários. Conforme as informações vão sendo selecionadas, mudam da cor branca para a verde, para ficar visualmente claro para o usuário.

5.2 Administrador cadastra colaborador

Caso esteja cadastrando um novo colaborador, o primeiro *checkbox* não é selecionado e todos os dados precisam ser preenchidos de acordo com os horários desse colaborador específico.

6. Tela de agendamento

Para acessar a tela de agendamento, basta clicar no botão de “novo agendamento” referente ao colaborador que prestará o serviço, na tela inicial (tela 4).

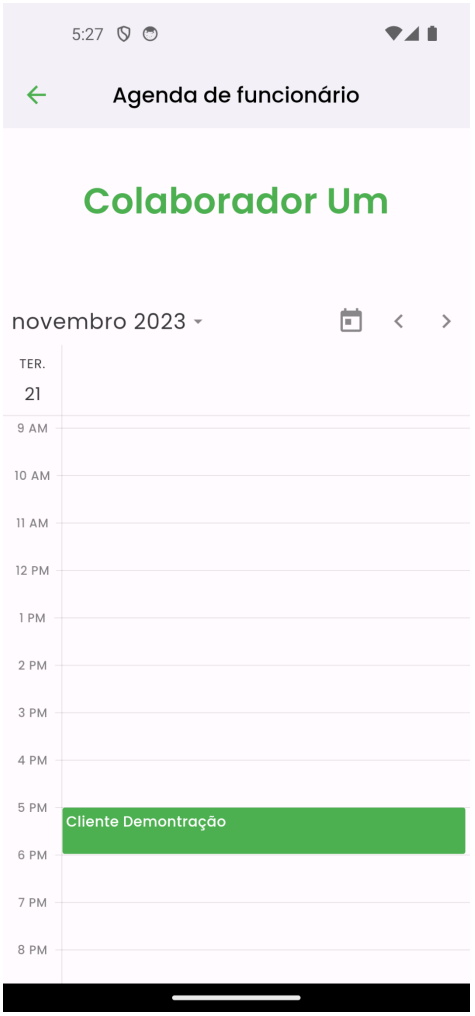
Nesse passo, basta colocar o nome do paciente/cliente, selecionar uma data e horário específico.

Nesse caso, só aparecerão disponíveis os dias e horários que foram determinados como padrão no cadastro do colaborador. Por isso, alguns horários ficam em cinza escuro, pois não podem ser selecionados, já que esse colaborador, por padrão, não trabalha na clínica nesses horários.

6.1 Tela de agendamento - Seleção de Data

Ainda no agendamento, ao clicar em “selecionar dia”, abre-se o *pop-up* de calendário, sempre no dia atual. A partir dele, o usuário consegue escolher dia e mês para fazer o agendamento do cliente.

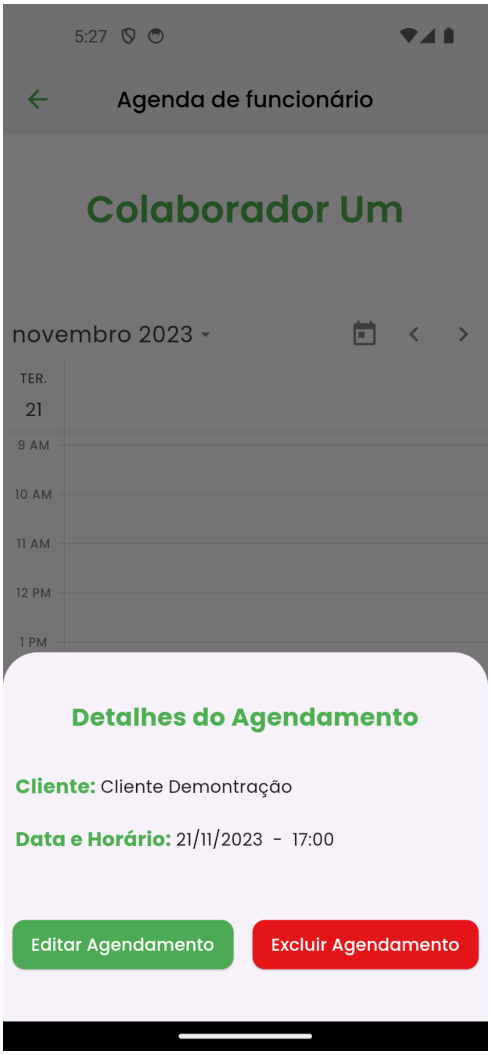
7. Visualização de agendamento de colaborador



Na tela inicial do aplicativo após login (tela 4), também é possível visualizar a agenda dos colaboradores cadastrados. Ao clicar em “ver agenda”, é possível visualizar a agenda do dia desse colaborador, com os nomes e horários dos clientes/pacientes.

Também é possível ver os dias anteriores e os próximos. Esse é um acesso de administrador e cada colaborador pode ver apenas sua agenda.

7.1 Tela de agendamento - Detalhes do agendamento



Ao clicar em cima do agendamento é possível ver os detalhes do agendamento, assim como editá-lo ou excluí-lo.

Caso opte pela edição do agendamento, retorna-se para a tela 6. Caso seja excluído, o agendamento sai da agenda desse colaborador.

8. **Tela inicial do colaborador, após logado**

Os colaboradores cadastrados pelo administrador também podem acessar o aplicativo, sendo reconhecidos como colaboradores de uma clínica específica a partir do seu email.

Ao logar (na tela 1), entram para a tela inicial do seu aplicativo, que conta com informações do seu nome, quantidade de agendamentos no dia atual e no próximo.

Também pode optar por fazer um agendamento em “agendar cliente’”, que levará para a tela 6.

E também pode visualizar sua agenda, com os agendamentos já cadastrados, sendo direcionado para a tela 7, também com opção de ver detalhes do agendamento.

Cada colaborador só pode acessar sua própria agenda. Apenas o administrador pode visualizar a agenda dos colaboradores cadastrados.