Agendei: proposta de desenvolvimento de uma aplicação móvel para realização de oferta de serviços e controle de agendamentos online

Érico Rosiski Weber[1], Gustavo Stangherlin Cantarelli[1]

[1] Sistemas de Informação – Universidade Franciscana (UFN)
97010-491 – Santa Maria – RS - Brasil

{ericoweber1996, gus.cant}@gmail.com

Abstract. *Based on the benefits that computerization of processes can guarantee in the context of service schedules, this work aimed to propose a hybrid mobile application focused on scheduling in establishments, in order to streamline, facilitate and deliver the best cost / benefit ratio for customers. To achieve this end, it was used the best practices of FDD for modeling, while for the hybrid mobile development the Flutter framework integrated to the Firebase Cloud Firestore database system was used.*

Resumo. *Com base nos benefícios que a informatização de processos pode garantir no contexto de agendamentos de serviços, este trabalho teve por objetivo propor uma aplicação móvel híbrida focada em agendamentos em estabelecimentos, de forma a agilizar, facilitar e entregar a melhor relação custo/benefício para os clientes. Para atingir esse fim, foi utilizado das boas práticas do FDD para modelagem, enquanto para o desenvolvimento móvel híbrido empregou-se o framework Flutter integrado ao sistema de banco de dados Firebase Cloud Firestore.*

1. Introdução

A oferta de serviços que exijam um agendamento prévio ainda segue padrões não informatizados, onde a necessidade por disponibilidade para realizar um agendamento está desprendendo tempo e recursos para empresas e clientes.

Em consideração ao fato exposto, este trabalho surgiu com a concepção de desenvolver uma aplicação móvel híbrida que permitirá aos usuários realizarem agendamentos em estabelecimentos que demandam para realização de algum serviço, enquanto para os prestadores de serviço uma maneira fácil, prática e rotineira de controlar estes agendamentos. Assim, espera-se criar uma ferramenta eficiente que forneça a melhor relação custo/benefício aos prestadores de serviço e para o cliente uma maneira cômoda de realizar os agendamentos de serviços, utilizando um método informatizado.

1.1. Justificativa e motivação

A necessidade de agendar consultas médicas, lavagem de carros, salões de beleza e, até mesmo, um serviço em um *pet shop*, exige que tanto o prestador de serviço quanto o cliente desprendam um tempo para esta comunicação. Assim, a possibilidade de se realizar agendamentos por meio de uma aplicação móvel, permitindo consultas de horários disponíveis para clientes possibilita uma melhora na gestão de agendas.

1.2. Objetivo Geral

O presente trabalho tem por objetivo o desenvolvimento de uma aplicação móvel capaz de oferecer ao cliente uma interface para realizar agendamentos com horários marcados, focada na agilidade e praticidade; para o prestador de serviço uma interface capaz de controlar a agenda de forma intuitiva, tanto por serviço como por funcionário.

1.3. Objetivos Específicos

Os objetivos específicos deste trabalho são:

- Realizar um levantamento dos requisitos gerais considerando um grupo de serviços ofertados;
- Pesquisar métodos de implementação de agendas, horários e gerenciamento de funcionários;
- Utilizar do *framework*[1] Flutter baseado em linguagem Dart para desenvolvimento da aplicação móvel híbrida;
- Utilizar o sistema gerenciador de banco de dados (SGBD) Firebase Cloud Firestore Database, possibilitando um dinamismo nos horários de agendamento e de sua estrutura;
- Garantir a possibilidade de realização de agendamentos com prestadores de serviços de forma prática, e seu gerenciamento pelos prestadores.

2. Referencial Teórico

Nesta seção, serão elucidados os assuntos que se fizeram necessários para elaboração do presente trabalho. Serão apresentados conceitos sobre o mercado a ser abordado, a estrutura de desenvolvimento, linguagem de programação, banco de dados, modelagem do problema e solução.

2.1. O mercado de serviços com agendamento

O mercado de serviços que exigem agendamento compreende todo serviço oferecido que demanda de um horário pré-determinado, pré-agendado, para atender seu cliente. Quando se refere a serviços com agendamento, supõe-se somente o processo de realizar o agendamento, não envolvendo os processos durante a realização do serviço em si.

Em diversos estados, serviços que até então existia o agendamento, porém de maneira não digital, hoje passam a oferecer o digital como opção principal. Entre os diversos estabelecimentos que ofertam serviços utilizando deste meio informatizado, destacam-se: concessionarias de automóveis, consultórios médicos, lavagens automotivas, estabelecimentos que envolvam cuidado com a estética e *pet shops*.

Hoje, a existência de agendamento online nos estabelecimentos chama a atenção do cliente e ganha destaque na mídia, como por exemplo, em 2019, a Volkswagen que ampliou a digitalização na jornada do cliente com o lançamento do agendamento online e o atendimento no pós-venda [A Hora 2019].

Outro exemplo é o sistema de agendamento online de consultas da UNIMED que está disponível para os clientes da Unimed Poços desde o fim de 2014. É uma ferramenta

[1]Framework é uma abstração que une códigos comuns entre vários projetos de software provendo uma funcionalidade genérica.

moderna, inovadora e de fácil utilização, oferecendo mais comodidade para o cliente, que poderá escolher o médico e o melhor horário para sua próxima consulta ou de seus familiares. O serviço foi planejado para possibilitar, em tempo real, 24 horas por dia, o encontro entre a demanda do cliente e a disponibilidade dos médicos [Unimed 2017].

Porém, as categorias de serviços que necessitam de softwares de agendamento online não se limitam nos exemplos citados anteriormente. O jornal Acrítica reporta que visita de familiares a presídios de Manaus contará com agendamento online [Acritica 2017].

Junto com esta demanda crescente por agendamentos online, cresce o número de softwares com esta finalidade. De acordo com reportagem, um dos serviços que tem crescido com o aperfeiçoamento da tecnologia é o de plataformas online para agendamento de consultas médicas [JE Online 2017].

Entre as vantagens mencionadas pela reportagem, comodidade e praticidade são os principais benefícios, mas melhor gestão, redução de custos e aumento na produtividade também são fatores favoráveis à nova tecnologia.

2.2. Por que oferecer agendamento online?

Em um infográfico publicado por Oiana, 77% dos clientes acreditam que agendar, alterar ou cancelar um agendamento online é importante. Ainda segundo a mesma fonte, 74% dos *millennials* (jovens entre 18 e 34 anos) valorizam o agendamento online.

Quanto ao mercado existente, o infográfico ressalta que grandes prestadores de serviços principalmente os relacionados a saúde como hospitais e clínicas já oferecem a possibilidade de agendamento para seus clientes. Entre os obstáculos ao acesso, o artigo destaca que 1 a cada 5 pessoas agenda serviços através de meios não tradicionais, sendo 85% dos usuários de smartphones utilizam algum tipo de agenda digital [Oiana 2017].

O infográfico conclui que 44% do público acha conveniente agendar online, mas eles não têm ferramenta para fazer isso. Dentre as vantagens citadas no infográfico, a implementação de um agendamento online pode trazer como benefícios: redução de custos com equipes de atendimento, possibilidade de agendamento a qualquer dia e qualquer hora, diminuição de gastos com ligações telefônicas e diminuição e controle das taxas de ausência. A qualidade do serviço de agendamento online também esta relacionada a tecnologia utilizada no desenvolvimento.

2.3 Tecnologias

Nesta seção serão apresentadas as ferramentas computacionais utilizadas no desenvolvimento da aplicação.

2.3.1 *Framework* Flutter

O Flutter é o kit de ferramentas de interface do usuário do Google para criar aplicativos belos e compilados nativamente para dispositivos móveis, web e desktop a partir de uma única base de código [Flutter 2019].

O *framework* foi totalmente desenvolvido em Dart, uma linguagem de propósito geral criada pela Google e muito similar a C# e Java, compilando código nativo para ARM e x86[2].

O carregamento rápido do Flutter ajuda o desenvolvedor a experimentar de forma rápida e fácil, criar *UI*s (*User Interfaces*), adicionar recursos e corrigir erros mais rapidamente. Tempos de recarga de menos de um segundo, sem perder o estado, em emuladores, simuladores e hardware para iOS e Android.

O Flutter possui os *widgets*[3] que seriam a mesma coisa de um componente. Um *widget* é uma árvore que pode conter um ou mais filhos (*widgets)*, e esses filhos são renderizados conforme a construção desta árvore [Santana 2019].

No Flutter há um detalhe, *Everything's a widget* (Tudo é um *widget*). Esse é um lema que os desenvolvedores levaram a sério ao criar o Flutter. Dessa forma elementos estruturais, como botões, caixas de texto e menus; elementos de estilo como fonte, cores e temas; aspectos de *layout* como espaçamento e posicionamento; até o controle de gestos e o próprio aplicativo são *widgets* [Henrique 2018].

Os *widgets* do *framework* incorporam todas as diferenças críticas de plataforma, como rolagem, navegação, ícones e fontes, para fornecer desempenho nativo completo no iOS e no Android.

Segundo Santana (2019): "Diferente do React Native que possui um intermediário (*bridge*) entre a *UI* e o dispositivo, o Flutter fica na camada do *UI* e não chama os componentes nativos do SO (sistema operacional), ele é desenhado diretamente em um canvas que aumenta a performance e fluidez a nível de um aplicativo desenvolvido exclusivamente nativo. Além do ganho com performance, o Flutter é uma tecnologia recente de fácil aprendizado e já possui algumas *features*[4] sendo desenvolvidas".

2.3.2 Linguagem Dart

Dart é uma linguagem de programação otimizada para o cliente para aplicativos rápidos em qualquer plataforma. Ela é uma linguagem orientada a objetos, definida por classe e *garbage-collected*[5]. Usando uma sintaxe no estilo C que se compara opcionalmente ao javascript. Ela suporta interfaces, classes abstratas, genéricos reificados e tipagem estática [Dart 2019a], [Dart 2019b], [Dart 2019c].

O Dart é compilado "antecipadamente" (AOT[6]) no código nativo para várias plataformas. Isso permite que o Flutter se comunique com a plataforma sem passar por uma ponte javascript que faz uma alternância de contexto. Compilar com código nativo também melhora o tempo de inicialização do aplicativo [Hackernoon 2017].

---

[2]ARM e x86 são arquiteturas de processadores.
[3]Um *widget*, numa interface gráfica, é um elemento de interação - tal como uma janela, um botão - podendo também se referir aos pequenos aplicativos que flutuam pela área de trabalho e fornecem funcionalidades específicas ao utilizador.
[4]Característica, recurso ou funcionalidade de um sistema computacional.
[5]*Garbage Colletion* (GC) é uma forma de gerenciamento automático de memória. O GC tenta recuperar o lixo ou a memória ocupada por objetos que não estão mais em uso pelo programa.
[6]Compilação AOT (*Ahead-Of-Time*) é quando todo o código é transformado de uma só vez antes de atingir as plataformas que o executam.

2.4 Cloud Firestore Firebase Database

O Firebase Cloud Firestore é um banco de dados hospedado na nuvem. Os dados são armazenados como *JSON*[7](*JavaScript Object Notation*) e sincronizados em tempo real com todos os clientes conectados. Quando você cria apps (aplicativos) em plataformas cruzadas com os SDKs[8] para iOS e Android, todos os clientes compartilham uma instância do Cloud Firestore e recebem automaticamente atualizações com os dados mais recentes. Com ele, é possível criar aplicativos avançados e colaborativos ao conceder acesso seguro ao banco de dados diretamente do código do cliente. O Cloud Firestore fornece uma linguagem de regras flexíveis baseadas em expressão, denominadas regras de segurança, para definir como os dados são estruturados e quando podem ser lidos e gravados. Por meio da integração com o Firebase *Authentication*, os desenvolvedores podem definir quem tem acesso, a quais dados e como esses dados podem ser acessados [Firebase 2019].

O Cloud Firestore é um banco de dados NoSQL (não relacional) e, por isso, tem otimizações e funcionalidades diferentes de um banco de dados relacional. A API do Cloud Firestore foi desenvolvida para autorizar apenas operações que possam ser executadas com rapidez. Isso possibilita uma ótima experiência em tempo real que atende a milhões de usuários sem comprometer a capacidade de resposta [Firebase 2019].

2.4.1 Banco de dados NoSQL

O banco de dados Firebase utiliza o modelo NoSQL para sua estruturação. Este modelo já é amplamente difundido e apresenta diversas vantagens.

Bancos de dados desse tipo são criados para modelos de dados específicos e têm esquemas flexíveis para a criação de aplicativos modernos. Eles são amplamente reconhecidos por sua facilidade de desenvolvimento, funcionalidade e performance em escala. Eles usam vários modelos de dados, incluindo documento, gráfico, chave-valor, memória e pesquisa [AWS 2019].

2.5. Modelagem

Existem diversos métodos para realizar a modelagem de um software. Um dos mais difundidos na engenharia de software é o FDD (*Feature Driven Development)*.

O método busca o desenvolvimento por funcionalidade, ou seja, por um requisito funcional do sistema. É prático para o trabalho com projetos iniciais ou projetos com codificações existentes [DevMedia 2019].

A Figura 1 demonstra os principais processos utilizados no FDD e que serão utilizados na metodologia deste trabalho.

---

[7]*JSON (JavaScript Object Notation)* é um formato compacto, de padrão aberto independente, de troca de dados simples e rápida entre sistemas que utiliza texto legível a humanos, no formato atributo-valor.
[8]SDKs ou *Software development kit*: Kit de desenvolvimento de software.

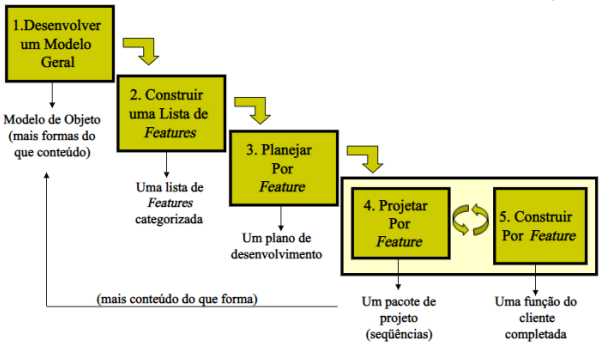

Figura 1. Processos Básicos FDD.

Este trabalho irá utilizar das boas práticas desta metodologia para realização do processo de projeto e desenvolvimento.

3. Trabalhos Correlatos

Nesta seção, serão apresentados trabalhos correlatos sobre o desenvolvimento de softwares de agendamento para prestadores de serviços, desenvolvimento móvel com *framework* Flutter, banco de dados NoSQL e empreendedorismo no setor de agendamentos.

3.1 Um aplicativo multiplataforma desenvolvido com Flutter e NoSQL para o cálculo da probabilidade de apendicite

O trabalho desenvolvido por Corazza (2018), propõe um aplicativo multiplataforma desenvolvido com Flutter e NoSQL para cálculo da probabilidade de apendicite e destaca o uso de linguagens de programação multiplataforma como vantagem para disponibilizar a aplicação para o maior número de usuários possíveis.

Como tecnologia de desenvolvimento utilizada em seu trabalho, Corazza explora o uso do *framework* Flutter chamando a atenção para características do seu fluxo de desenvolvimento orientado ao design e *widgets* (blocos básicos da interface). Ele destaca que diferentemente dos demais *frameworks*, o Flutter utiliza de um mecanismo próprio de renderização de alto desempenho para desenhar sua interface, proporcionando maior performance frente aos demais, mesmo em multiplataforma.

Ainda segundo Corazza, a utilização de bancos de dados NoSQL, com o uso do Firebase *Cloud Firestore,* foi possível implementar o armazenamento dos dados na nuvem e diretamente a partir do aplicativo, eliminando a necessidade de registrar essas informações em alguma ferramenta externa. Isso acontece pois o *framework* Flutter possui uma facilidade de integração com o Firebase.

No trabalho, Corazza conclui que o Flutter possibilita a criação de uma interface que atendesse ao atributo da usabilidade, o que era fundamental para a disseminação da aplicação, não deixando de lado o fator desempenho e atraindo assim a maior quantidade de usuários possíveis.

3.2 Aplicativo para agendamento de horário em salões de beleza

Santos (2016), desenvolveu em seu trabalho de conclusão de curso um sistema para agendamento de horário em salões de beleza, com módulos web e móvel.

Para levantamento e extração dos requisitos de negócio, Santos utilizou do RUP *(Rational Unified Process)* como processo de engenharia de software para aumentar a produtividade. Como tecnologias utilizadas para o desenvolvimento, Santos utiliza do *framework* Ionic baseado em javascript para a aplicação móvel e para a aplicação web da linguagem Java e do *framework* AngularJS, também ambos baseados em javascript.

Santos, baseado nas experiências adquiridas em seu trabalho, concluiu que a inclusão de promoções relacionadas à horários é uma possibilidade de inclusão para este tipo de sistema proposto e surge como demanda.

3.4 Softwares Correlatos

Nesta seção, serão apresentados softwares correlatos que possuem alguma similaridade com o presente trabalho.

3.4.2 Gendo Super Agendador

O software Gendo oferece um sistema de agendamento online, disponibilizando versão web e móvel, com funções de agenda, financeiro, clientes, serviços, produtos e profissionais. Seu foco principal está no gerenciamento de agendamentos de forma simples e moderna, possibilitando abertura e fechamento de agendas em períodos determinados.

Possui como funções extras o envio de SMS de lembrete, visualização diária ou semanal da agenda e ficha com cadastro de clientes. Além disto, o sistema gera relatórios financeiros e de abandono de clientes.

Entre as opções de utilização, o software é vendido como SaaS (*Software as a service*) e disponibilizado em três planos mensais sendo a diferença entre eles a quantidade de usuários administrativos do sistema e o número de SMS de lembretes por mês.

O diferencial do Gendo se dá por oferecer a possibilidade de criação de uma interface personalizada para agendamento de serviços. Tendo como base que é possível baixar um aplicativo padrão e, o mesmo, conforme as configurações online é personalizado para o estabelecimento.

Atualmente, a empresa atua na região do estado de São Paulo, possuindo mais de 170 mil empresas cadastradas, mais de 47 milhões de agendamentos realizados e 19 milhões de clientes que já utilizaram a plataforma como clientes finais. Não há informações sobre o ano de surgimento da empresa.

3.5 Considerações sobre os trabalhos relacionados

Os trabalhos correlatos citados ilustram diversos aspectos relacionados ao desenvolvimento do presente trabalho, enfatizando desde tecnologias utilizadas, metodologias de compreensão, levantamento de requisitos, sistemas criados voltados com o mesmo intuito e a situação do mercado de agendamentos online.

Dos trabalhos citados, o trabalho de Corazza (2018) assemelha-se com o presente trabalho pela utilização da mesma tecnologia de desenvolvimento, trazendo conceitos e exemplificando porque esta tecnologia está de acordo com o trabalho que será desenvolvido.

Enquanto Corazza explora a mesma tecnologia a ser desenvolvida neste trabalho, porém com objetivo final diferente; Santos (2016), desenvolve um software muito semelhante ao proposto no presente trabalho, porém, utilizando tecnologia diferente e focado exclusivamente a um nicho. Seu trabalho contribuiu para compreensão dos requisitos levantados e exemplificação de *layouts* de interface.

Já o trabalho de Franco (2013) propõe um sistema genérico de agendamento, porém focado para uso web e não utilizando de tecnologias mais atuais, sendo seu trabalho muito bem documentado, ajudando na compreensão dos diagramas UML criados por Franco.

Os softwares correlatos citados ajudam no presente trabalho, fornecendo uma dimensão de *features* oferecidas e aceitas pelo mercado. Além disto, eles evidenciam a existência de um amplo mercado consumidor para utilização de softwares de agendamento online.

Todos os trabalhos citados tiveram importância para garantir que a tecnologia a ser utilizada, os métodos de modelagem do problema e da solução assim como a existência de um mercado para este produto, estejam adequados para a solução, garantindo que a concepção da ideia inicial do presente trabalho seja válida, propondo uma nova forma de interação entre prestadores de serviços e usuários para automatizar agendamentos de uma maneira simplificada e econômica.

Diferentemente dos trabalhos citados, o software deste trabalho irá buscar a interação com o usuário cliente e usuário prestador de serviço totalmente móvel, diferenciando-se das demais por focar no desenvolvimento híbrido atingindo o maior número de clientes e gerando notificações com alertas promocionais, mantendo uma experiência de interface amigável e intuitiva.

4. Metodologia

Nesta seção são apresentadas as etapas desenvolvidas com o uso da metodologia FDD. Sua escolha esta relacionada a prática conveniente de modelagem e elaboração através de *features*. Esta metodologia está conceituada na seção referencial teórico.

4.1. Desenvolvimento de um modelo abrangente

A Figura 2 exibe o modelo abrangente a ser desenvolvido utilizando da notação BPMN (*Business Process Model and Notation*)[9]. Este modelo exemplifica como ocorre a interação cliente e prestador de serviço para realização de agendamentos sem a utilização de um software ou aplicação.

[9]Notação de Modelagem de Processos de Negócio é uma notação da metodologia de gerenciamento de processos de negócio e trata-se de uma série de ícones padrões para o desenho de processos, o que facilita o entendimento do usuário.

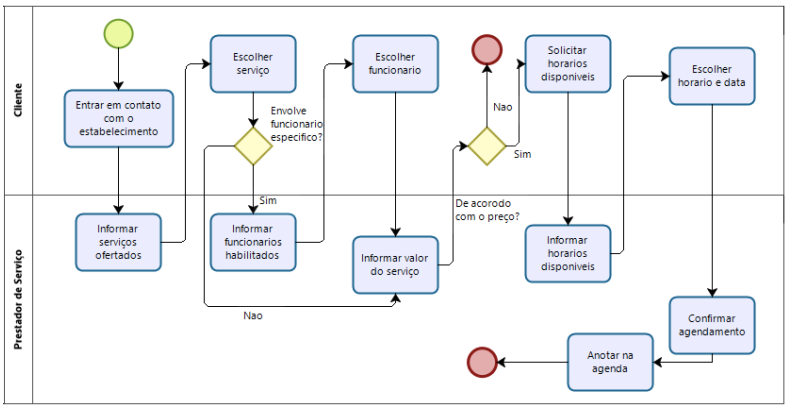

Figura 2. Modelo Abrangente.

Baseado no modelo abrangente apresentado, projetou-se uma aplicação móvel a qual possuirá dois níveis de interface: uma voltada para usuários que desejam realizar um agendamento em algum estabelecimento; e outra, para os prestadores de serviço administrarem estes agendamentos. As próximas seções apresentarão o projeto desta aplicação com lista de funcionalidades, planejamento por funcionalidades principais, detalhes por funcionalidades principais, resultados e conclusão.

4.2. Lista de funcionalidades

Nesta subseção são apresentadas algumas das diversas funcionalidades que atendem aos requisitos do sistema proposto, podendo serem classificadas entre requisitos funcionais, isto é, que descrevem o que o sistema fará, ou não funcionais, que retratam como o sistema fará.

Na Tabela 1 encontram-se os principais requisitos funcionais (RF) do usuário "cliente" do sistema, sendo os demais apresentados no Apêndice A.

Tabela 1. Principais Requisitos Funcionais (RF) Cliente.

ID	Descrição	Relevância	Complexidade
RF1	Efetuar Login	Essencial	Baixa
RF2	Agendar Compromisso RF2.1 Pesquisar/Buscar RF2.2 Escolher Estabelecimento RF2.3 Escolher Serviço RF2.4 Escolher Funcionário RF2.5 Escolher Horário e Data RF2.6 Verificar Disponibilidade	Essencial	Alta

Na Tabela 2 são listados os principais requisitos funcionais (RF) do usuário "prestador de serviço". Demais requisitos são apresentados no Apêndice B.

Tabela 2. Requisitos Funcionais (RF) Prestadores de Serviço.

ID	Descrição	Relevância	Complexidade
RF1	Efetuar Login	Essencial	Baixa
RF2	Gerenciar Serviços	Essencial	Alta
RF3	Gerenciar Funcionários	Essencial	Alta

Os requisitos não funcionais podem ser encontrados no Apêndice C.

4.3. Planejamento por funcionalidade

Nesta etapa foi planejado o desenvolvimento das funcionalidades por requisito e o tempo para desenvolvimento em dias. As tabelas encontram-se no Apêndice D.

4.4. Detalhar por funcionalidade

Nesta seção, é explorada de forma mais aprofundada e detalhada os requisitos anteriormente determinados, esclarecendo o fluxo de execução do sistema por meio de diagramas de casos de uso (Apêndice E) e descritivos (Apêndice F). As funcionalidades foram implementadas seguindo um cronograma estipulado (Apêndice D).

Com o auxílio dos diagramas de atividades (Apêndice G) foi possível construir por funcionalidade. A Figura 3 demonstra o diagrama de atividades do cliente para a atividade de agendar um compromisso, atividade a qual será mais recorrente no uso da aplicação pelo cliente.

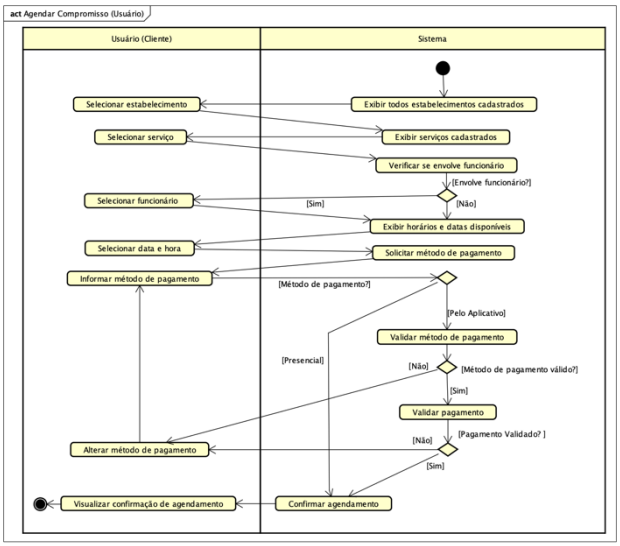

Figura 3. Diagrama de Atividades Agendar Compromisso (cliente).

O diagrama de atividade (Figura 4) demonstra a atividade principal da aplicação para o prestador de serviço.

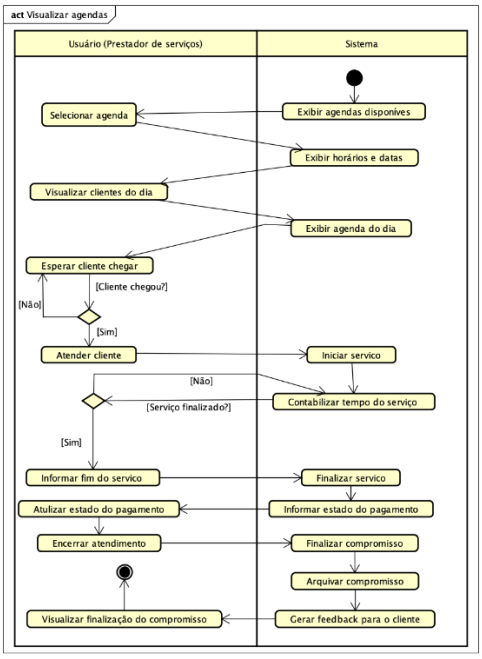

Figura 4. Diagrama de Atividades Visualizar Agendas (prestador de serviço).

Por fim, na Figura 5, é apresentado o Diagrama de classes, expondo uma nova visão estrutural e demonstrando as classes que constituirão o sistema com seus respectivos atributos e métodos, além das relações entre as mesmas.

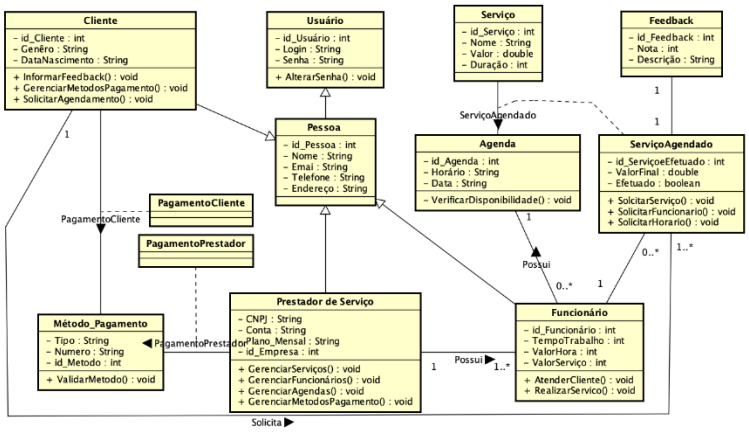

Figura 5. Diagrama de Classes.

Para representar a estrutura dos objetos e suas relações propostas, foi criado um banco de dados (Figura 6). Por ser um banco de dados estruturado em linguagem não relacional, ele é composto por documentos e coleções. Um diagrama explicativo de sua estrutura pode ser encontrado no (Apêndice H).

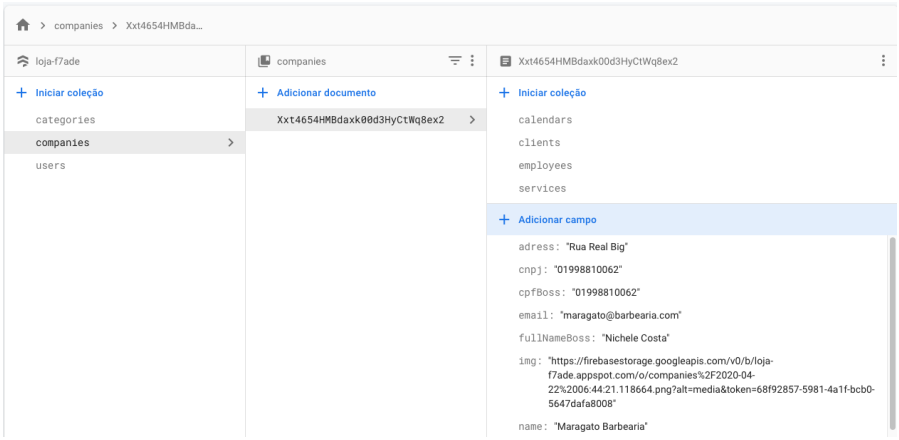

Figura 6. Banco de dados criado.

4.5 Construir por funcionalidade

Por fim, construiu-se a aplicação por meio de funcionalidades, utilizando do banco de dados criado para armazenamento das variáveis, e dos diagramas de atividades para orientar a sequências de interações internas na aplicação.

A Figura 7 exibe a funcionalidade de busca no banco de dados de calendários referentes a um serviço em uma empresa. O código encontra-se comentado.

```
void getCalendars() async { // FUNCAO QUE BUSCA TODOS CALENDARIOS COM DETERMINADO SERVICO
  final QuerySnapshot query = await Firestore.instance.collection('companies').document(widget.uidCompany).collection('calendars')
      .where('uidService', isEqualTo: selectedService).getDocuments(); // REALIZA A CONSULTA NO BANCO
  List<DocumentSnapshot> calendarItems = []; // CRIA UMA LISTA DE DOCUMENTOS
  for (int i = 0; i < query.documents.length; i++) {
    calendarItems.add(query.documents[i]); // ADICIONA CADA DOCUMENTO DA CONSULTA NA LISTA
  }
  print('calendarios com o servico encontrados: ' + calendarItems.length.toString());
  getEmployee(calendarItems); //CHAMA A FUNCAO GETEMPLOYEE E PASSA O PARAMETRO LISTA
}
```

Figura 7. Funcionalidade de busca de calendários.

Já a Figura 8 demonstra a busca por todos funcionários que realizam determinado serviço e inclusão em uma lista. Ao fim, pode ser observada a chamada de outra funcionalidade de tempo de duração de um serviço.

```
void getEmployee(List<DocumentSnapshot> calendar) async { //VERIFICA QUAIS FUNCIONARIOS PARTICIPAM DOS CALENDARIOS
  for (int i = 0; i < calendar.length; i++) {
    print(calendar[i].data['name']); // EXIBE O NOME DESTES CALENDARIOS
    final DocumentSnapshot employee = await Firestore.instance.collection('companies').document(widget.uidCompany)
        .collection('employees').document(calendar[i].data['uidEmployee']).get(); //CONSULTA O NOME DO FUNCIONARIO DO CALENDARIO
    print('funcionario: ' + employee.data['fullName']);// EXIBE O NOME DO FUNCIONARIO
    if (!employeeItems.contains(employee.documentID)) {
      setState(() {
        employeeItems.add(DropdownMenuItem( //ADICIONA O FUNCIONARIO A UM COMPONENTE CASO O MESMO NAO ESTEJA PRESENTE
          child: Text(employee.data['fullName']),
          value: '${employee.documentID}',
        )); // DropdownMenuItem
        print(employeeItems.length);
      });
    }
  }
```

Figura 8. Funcionalidade de busca de funcionários.

5. Resultados

A proposta principal deste trabalho foi a criação de uma aplicação para agendamento e controle de horários. A aplicação desenvolvida para o prestador de serviços (Figura 9) inclui *login* (Figura 9.A), possibilidade de gerenciamento de serviços (Figura 9.B) e agendas, sendo o prestador capaz de visualizar os agendamentos em um calendário (Figura 9.C), atender o cliente (Figura 9.D) e gerir seus dados, principais funções cotidianas do prestador de serviço, todas elas validadas por meio de testes funcionais.

Demais funções e interfaces da aplicação do prestador de serviço podem ser visualizadas no Apêndice I.

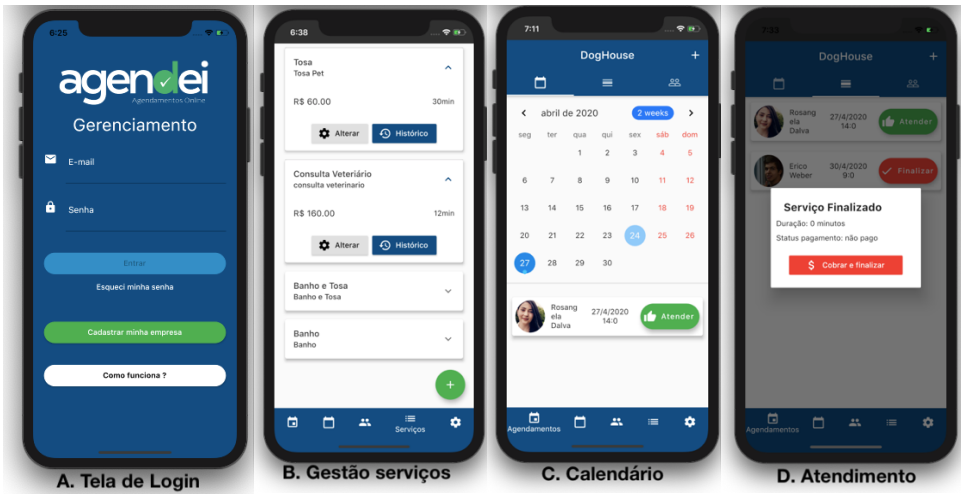

Figura 9. Principais interface aplicação prestador de serviço.

Percebe-se a facilidade de atender um cliente, uma vez que o próprio cliente pode realizar o agendamento e o prestador utilizar o software para realizar o serviço e finalizá-lo. Caso necessário o próprio prestador pode realizar o agendamento de um cliente e até mesmo incluir um cliente em seu sistema. Assim é possível integrar todos agendamentos que não forem realizados pela aplicação com a aplicação.

Já a versão da aplicação para o cliente oferece como recursos principais (Figura 10) a visualização de todos estabelecimentos (Figura 10.A), a tela para realização de agendamento (Figura 10.B).

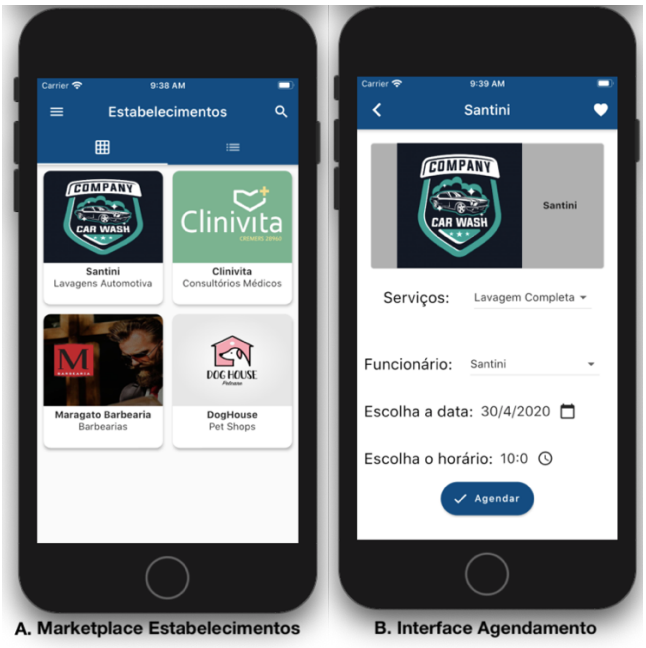

Figura 10. Principais interfaces aplicação cliente.

Algumas das demais interfaces úteis ao cliente são: *login* (Figura 11.E), visualização por categoria (Figura 11.D) e menu principal (Figura 11.C).

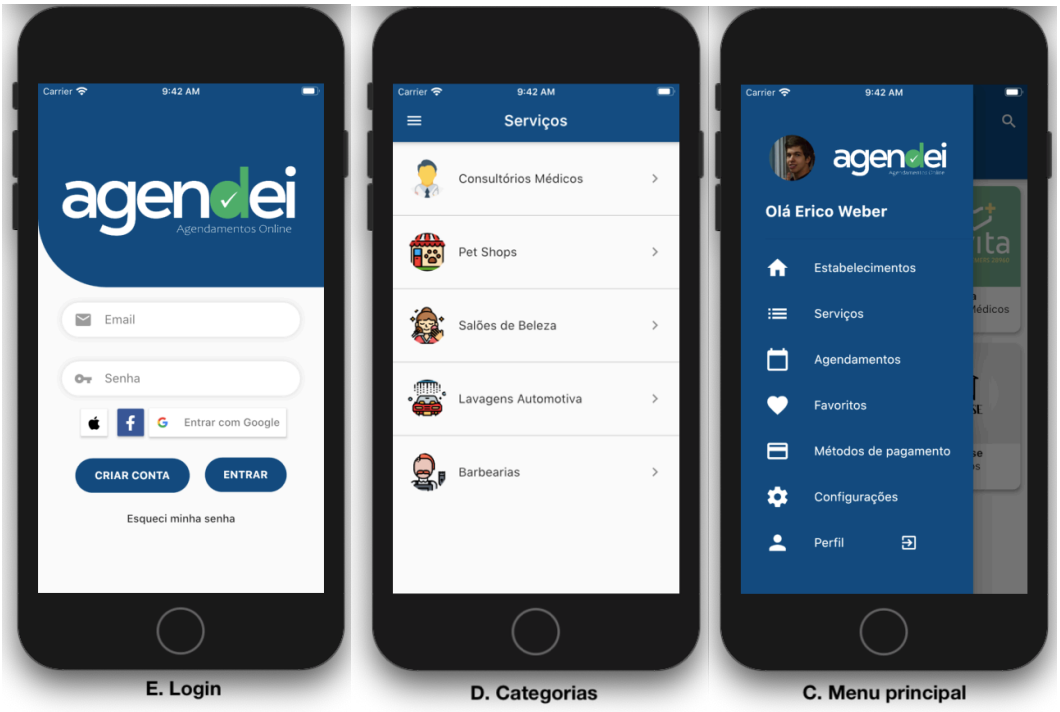

Figura 11. Demais funções aplicação cliente.

Interfaces do menu (Figura 11.C) encontram-se no Apêndice J para consulta.

6. Conclusão

Baseado nas pesquisas sobre o mercado de serviços de agendamento e uso da tecnologia para melhoramento do setor, este trabalho desenvolveu uma aplicação móvel capaz de realizar agendamentos e controle de horários de atendimento para clientes e prestadores de serviços, atingindo todos objetivos traçados e entregando uma aplicação com alta qualidade e fidelidade aos requisitos funcionais traçados.

Os trabalhos correlatos sumarizaram a importância das tecnologias escolhidas neste trabalho e exemplificaram casos reais de desenvolvimento, trazendo junto as *features* mais utilizadas e aceitas pelos usuários.

A escolha do Flutter como *framework* de desenvolvimento possibilitou uma integração com mais plataformas, entregando uma solução mais ágil. O Banco de dados escolhido possibilitou realizar todos os objetivos traçados, contudo ao final do projeto, percebeu-se que a escolha do mesmo não se encaixava como melhor opção para os tipos de dados a serem manipulados, principalmente por tratar-se de dados de horários.

Quanto a metodologia, escolheu-se as boas práticas da metodologia FDD, o que favoreceu o desenvolvimento por funcionalidades e garantiu a execução de testes ao longo de todo projeto.

Para trabalhos futuros, sugere-se a integração com outro tipo de banco de dados, de preferência relacional, a possibilidade de integração da aplicação para uso web, a implementação de um método de pagamento online vinculado a um programa de pontos e um sistema de notificações eficiente tanto para o prestador de serviço como para o cliente.

Referências

Acrítica. (2019) “Visita de familiares a presídios de Manaus contará com agendamento online”. Disponível em: https://www.acritica.com/channels/cotidiano/news/visita-a-presidios-de-manaus-contara-com-agendamento-online. Acesso: Setembro/2019.

Agendo. (2019) “Funções do Sistema”. Disponível em: https://www.gendo.com.br/funcoes.html. Acesso: Setembro/2019.

AWS - Amazon Web Services. (2018) “Banco de dados NoSQL”. Disponível em: https://aws.amazon.com/pt/nosql/. Acesso: Setembro/2019.

Cedro Tecnologias. (2019) “Feature Driven Development (FDD)”. Disponível em: https://blog.cedrotech.com/feature-driven-development-fdd/. Acesso: Setembro/2019.

Corazza, Paulo V. (2018) “Um aplicativo multiplataforma desenvolvido com Flutter e NoSQL para o cálculo da probabilidade de apendicite”. Disponível em: https://www.lume.ufrgs.br/bitstream/handle/10183/190147/001088749.pdf?sequence=1&isAllowed=y. Acesso: Agosto/2019.

Dart. (2019a) Disponível em: https://dart.dev/. Acesso: Setembro/2019.

Dart. (2019b) "A Tour of the Dart Language". Disponível em:www.dartlang.org. Acesso: Setembro/2019.

Dart. (2019c) "The Dart type system". Disponível em: https://dart.dev/. Acesso: Setembro/2019.

DevMedia. (2013) “Introdução ao FDD – Feature Driven Development”. Disponível em: https://www.devmedia.com.br/introducao-ao-fdd-feature-driven-development/27971. Acesso: Setembro/2019.

Esplin, Chris. (2016) “Firebase Data Structures: Complex Data”. Disponível em: https://medium.com/google-cloud/firebase-data-structures-complex-data-eb76b5a31124. Acesso: Setembro/2019.

FRD - Firebase Realtime Database. (2019) “Documentation” Disponível em: https://firebase.google.com/docs/database. Acesso: Setembro/2019.

Henrique, Paulo. (2018) “Conheça o Flutter, a aposta da Google para a criação de apps nativos multiplataforma”. Disponível em: https://medium.com/liferay-engineering-brazil/conhe%C3%A7a-o-flutter-a-aposta-da-google-para-a-cria%C3%A7%C3%A3o-de-apps-nativos-multiplataforma-e59c610134d8. Acesso: Setembro/2019.

Heptagon. (2017) “FDD – Feature Driven Development”. Disponível em: http://heptagon.com.br/fdd/. Acesso em: Setembro/2019.

https://www.jornalahora.com.br/2019/08/28/vw-cria-agendamento-online/. Acesso: Setembro/2019.

Je Online. (2017) “Cresce o número de plataformas online para agendamento de consultas médicas”. Disponível em: https://jeonline.com.br/noticia/11281/cresce-o-numero-de-plataformas-online-para-agendamento-de-consultas-medicas. Acesso: Setembro/2019.

Jornal A Hora. (2019) “VW cria agendamento online”. Disponível em:

https://www.jornalahora.com.br/2019/08/28/vw-cria-agendamento-online/. Acesso: Setembro/2019.

Luisi, Ana M. Vieira. (2017) “Sistema de Agendamento Online de Consultas”. Disponível em: https://www.unimed.coop.br/web/pocosdecaldas/noticias/treinamento-sobre-agendamento-online-de-consultas. Acesso: Setembro/2019.

Oiana. (2017) “Por que você deve oferecer agendamento online aos seus clientes?”. Disponível em: https://oiana.com.br/blog/atendimento/por-que-voce-deve-oferecer-agendamento-online-aos-seus-clientes/. Acesso: Agosto/2019.

Santana, Fabiano. (2019). “Flutter: porque você deveria apostar nesta tecnologia”. Disponível em: https://medium.com/tableless/flutter-porque-voc%C3%AA-deveria-apostar-nesta-tecnologia-94a510fffd18. Acesso: Setembro/2019.

Santos, Silvanei S. (2016) “Aplicativo para agendamento de horário em salões de beleza”. Disponível em: https://acervodigital.ufpr.br/bitstream/handle/1884/50206/R%20-%20E%20%20SILVANEI%20SOARES%20SANTOS.pdf?sequence=1&isAllowed=y. Acesso em: Agosto/2019.

Strauch, Christof. (2011) “NoSQL Databases”. Disponível em: https://christof-strauch.de/nosqldbs.pdf. Acesso: Setembro/2019.

Apêndice A. Requisitos Funcionais Usuário Cliente

Tabela 3. Requisitos Funcionais (RF) Clientes.

ID	Descrição	Relevância	Complexidade
RF1	Efetuar Login	Essencial	Baixa
RF2	Agendar Compromisso RF2.1 Pesquisar/Buscar RF2.2 Escolher Estabelecimento RF2.3 Escolher Serviço RF2.4 Escolher Funcionário RF2.5 Escolher Horário e Data RF2.6 Verificar Disponibilidade RF2.7 Verificar Preço	Essencial	Alta
RF3	Gerenciar Agenda RF 3.1 Visualizar Compromissos RF3.2 Alterar Horário ou Data RF3.3 Cancelar Compromisso RF3.4 Verificar Próximos Compromissos	Essencial	Alta
RF4	Gerenciar Favoritos RF5.1 Gerenciar Estabelecimentos Favoritos	Não Essencial	Baixa
RF5	Gerenciar Configurações RF6.1 Gerenciar Métodos de Pagamento RF6.2 Gerenciar Informações de Perfil RF6.3 Excluir Conta	Essencial	Alta
RF6	Gerenciar *Feedback*[10] RF7.1 Informar *feedback*	Essencial	Baixa

[10]*Feedback*: informação que o prestador de serviço obtém da reação do cliente a seu serviço, e que serve para avaliar os resultados dos serviços prestados. Valor fornecido em formato numérico.

Apêndice B – Requisitos funcionais Usuário Prestador de Serviços

Tabela 4. Requisitos Funcionais (RF) Prestadores de Serviço.

ID	Descrição	Relevância	Complexidade
RF1	Efetuar Login	Essencial	Baixa
RF2	Gerenciar Serviços	Essencial	Alta
RF3	Gerenciar Funcionários	Essencial	Alta
RF4	Gerenciar Agendas RF4.1 Organizar Agendas RF4.2 Alterar Horários RF4.3 Vincular Serviços/Funcionários	Essencial	Alta
RF5	Visualizar Agendamentos RF5.1 Controlar Horários RF5.2 Atender cliente RF5.3 Finalizar Atendimento	Essencial	Alta
RF6	Gerenciar Notificações RF6.1 Criar Promoção/Notificação RF6.2 Visualizar Alcance/Adesão	Não Essencial	Baixa
RF7	Gerenciar Configurações RF7.1 Gerenciar Dados da Empresa	Essencial	Alta
RF8	Gerenciar Relatórios RF8.1 Gerar Demonstrativos Financeiros RF8.2 Gerar Relatórios de Serviço, Funcionários, Clientes	Essencial	Alta
RF9	Agendar Cliente	Essencial	Alta

Apêndice C – Requisitos não funcionais (RFN)

Tabela 5. Requisitos Não Funcionais (RNF).

ID	Descrição	Relevância	Complexidade
RNF1	A aplicação será desenvolvida em Flutter baseado em linguagem Dart.	Essencial	Média
RNF2	O sistema fará uso do banco de dados Firebase Realtime Database.	Essencial	Média
RNF3	A aplicação será híbrida para os sistemas operacionais móveis iOS e Android.	Essencial	Alta
RNF4	O sistema terá integração com a agenda nativa do aparelho.	Não Essencial	Baixa

Apêndice D – Planejamento por funcionalidade

Tabela 6. Planejamento por Funcionalidade do Usuário Cliente.

Sequência	Funcionalidade	Tempo de Desenvolvimento Estimado (Dias)
1	RF1	5
2	RF4, RF5, RF7	5
3	RF6	5
4	RF2	20
5	RF3	20
6	RF8	10

Tabela 7. Planejamento por Funcionalidade do Usuário Prestador de Serviço.

Sequência	Funcionalidade	Tempo de Desenvolvimento Estimado (Dias)
1	RF1	5
2	RF2	5
3	RF3	5
4	RF4, RF5	20
5	RF7	20
6	RF6	5
7	RF8	10
8	RF9	10

Apêndice E – Diagramas de Casos de Uso

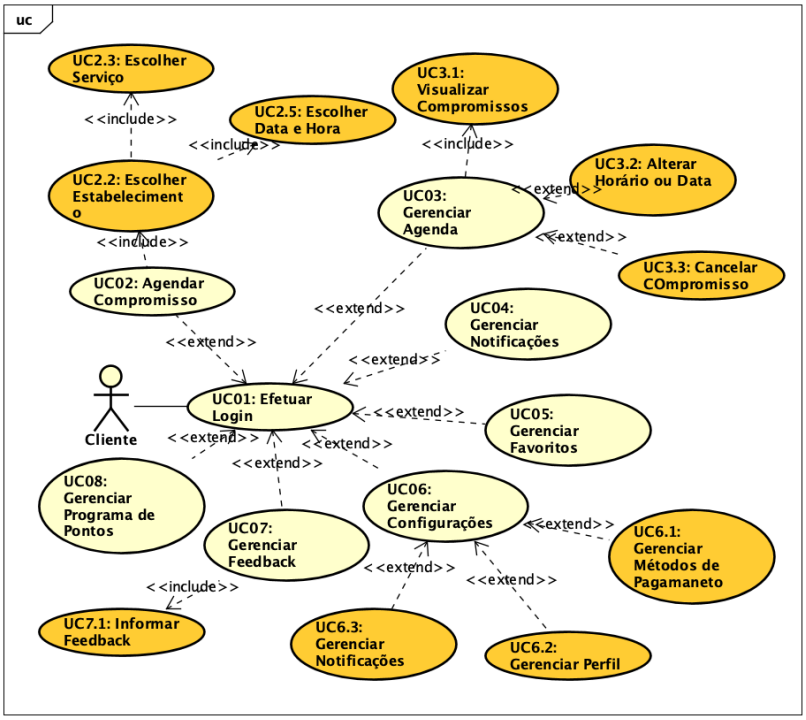

Figura 12. Diagrama de Casos de Uso (Cliente).

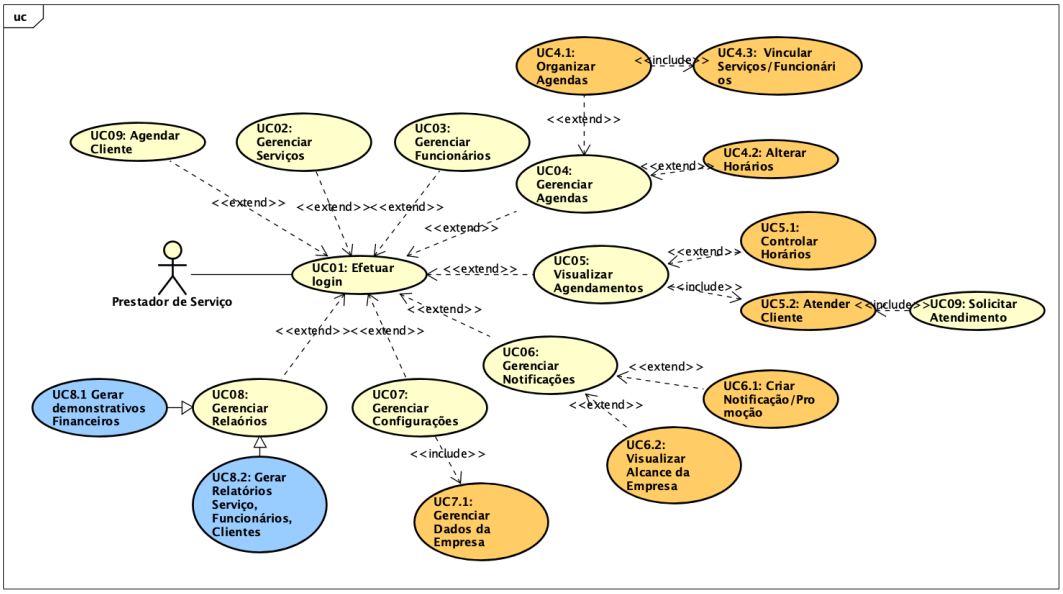

Figura 13. Diagrama de Casos de Uso (Prestador de Serviços).

Apêndice F – Descritivos de casos de uso

Tabela 8. Descritivo do Caso de Uso Agendar Compromisso (Cliente).

Identificação	UC02
Casos de Uso	Agendar Compromisso.
Descrição	Permite ao usuário (cliente) realizar o agendamento de um serviço, escolhendo as opções necessárias para o agendamento.
Ator Principal	Usuário Final (Cliente).
Ator(es) Secundário(s)	N/A
Pré Condições	
PC1	O usuário deve estar logado no sistema.
Pós Condições	
PO1	Agendamento realizado com sucesso com pagamento efetivado.
PO2	Agendamento abortado.
PO3	Agendamento realizado com sucesso com pagamento pendente.
Fluxo Principal	
FP1	O sistema exibe os estabelecimentos credenciados ao agendamento online. Logo em seguida é exibido os serviços [FP1.1], funcionários [FA1] [FA2], horários e datas [FP1.2], formas de pagamento [FP1.3] e é realizado o agendamento. FP1.1 Exibir serviços oferecidos. FP1.2 Exibir horários e datas disponíveis. FP1.3 Exibir formas de pagamento.
Fluxo Alternativo	
FA1	Não é necessário informar funcionários, pois todos realizam o serviço.
FA2	É necessário exibir os funcionários habilitados para a realização do serviço, pois nem todos realizam o serviço.
Fluxo de Exceções	
FE1	Método de pagamento não autorizado, informar outro método de pagamento.
Regras de Negócio	Somente são exibidos os horários que o prestador de serviço oferecer.
Casos de Uso Estendidos	N/A
Casos de Uso Incluídos	UC2.2, UC2.3, UC2.5

Tabela 9. Descritivo Caso de Uso Atendimento (prestador de serviço).

Identificação	UC5.2
Casos de Uso	Atender cliente agendado (acompanhamento de agenda).
Descrição	Permite ao usuário (prestador de serviço) acompanhar a sua agenda e realizar o atendimento ao cliente agendado.
Ator Principal	Usuário Final (Prestador de Serviço).
Ator(es) Secundário(s)	Usuário Final (Cliente)
Pré Condições	
PC1	O usuário cliente deve ter agendado um horário antecipadamente.
PC2	O usuário prestador de serviço deve estar logado ao sistema
PC3	O usuário prestador de serviço deve ter permissão de acesso a agenda a qual deseja visualizar.
Pós Condições	
PO1	O usuário cliente deve fornecer um *feedback* para conclusão do agendamento.
PO2	O usuário prestador de serviço irá visualizar os agendamentos já concluídos.
Fluxo Principal	
FP1	O prestador de serviço abre a agenda e verifica por serviço ou funcionário os próximos clientes agendados. Após a chegada do cliente do horário indicado, é confirmado o início do serviço [FP1.1], e quando encerrado é informado a aplicação [FP1.2], e então exibido o status de pagamento o qual o prestador de serviço irá atualizar [FP1.3] e informado a finalização do atendimento. Logo o usuário cliente pode fornecer um feedback anônimo do serviço realizado. [FP1.4]. FP1.1 Informar o início do atendimento. FP1.2 Informar finalização do serviço. FP1.3 Atualizar estado do pagamento. FP1.4 Informar *feedback* (Cliente).
Fluxo Alternativo	
FA1	O cliente não realizou o pagamento online e é necessário atualizar o campo de pagamento.
Fluxo de Exceções	
FE1	Não foi informado a finalização do serviço. A aplicação irá abortar a execução do serviço após um período de tempo.

Regras de Negócio	Somente é exibido os agendamentos realizados online pelos clientes.
Casos de Uso Estendidos	N/A
Casos de Uso Incluídos	UC09

Apêndice G – Diagramas de atividade

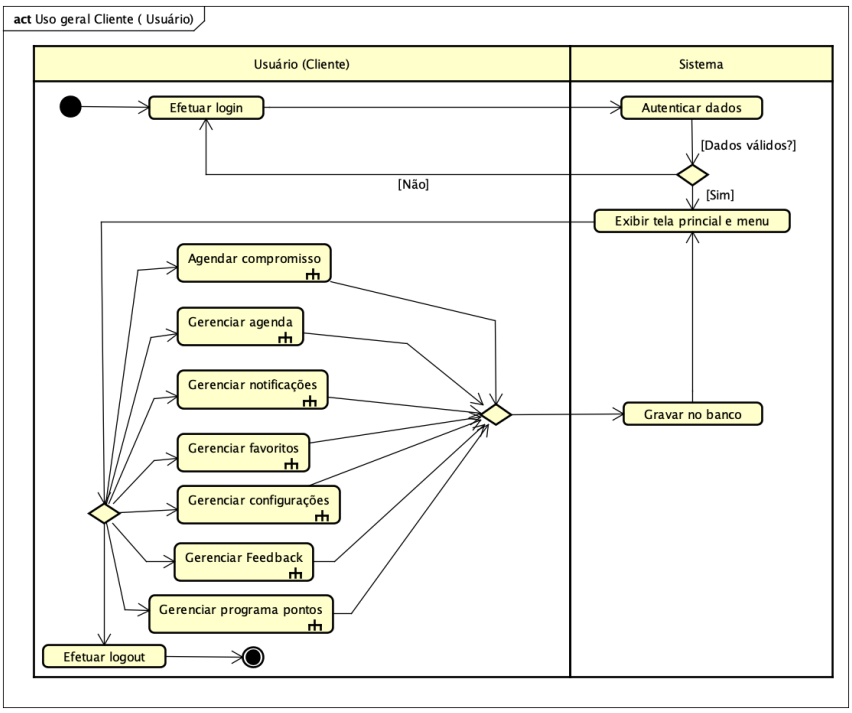

Figura 14. Diagrama de Atividades Sobre Uso Geral (Cliente).

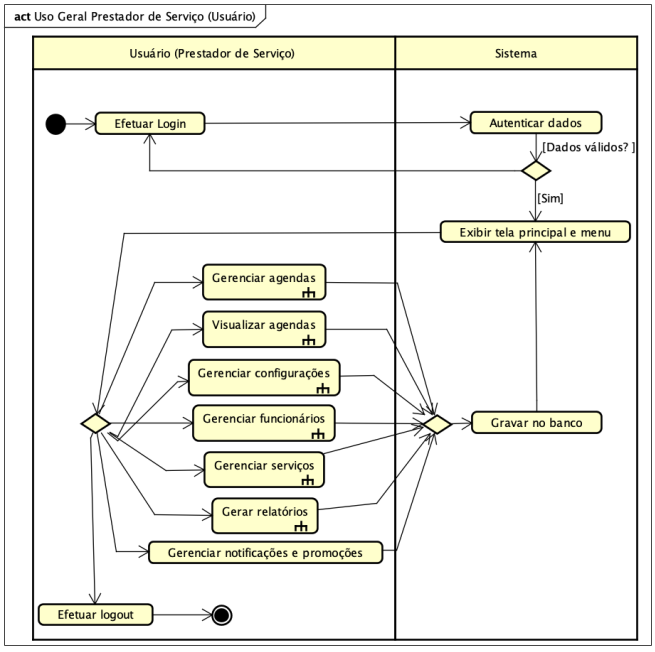

Figura 15. Diagrama de Atividades Sobre Uso Geral (prestador de serviço).

Apêndice H – Diagrama explicativo Banco de dados

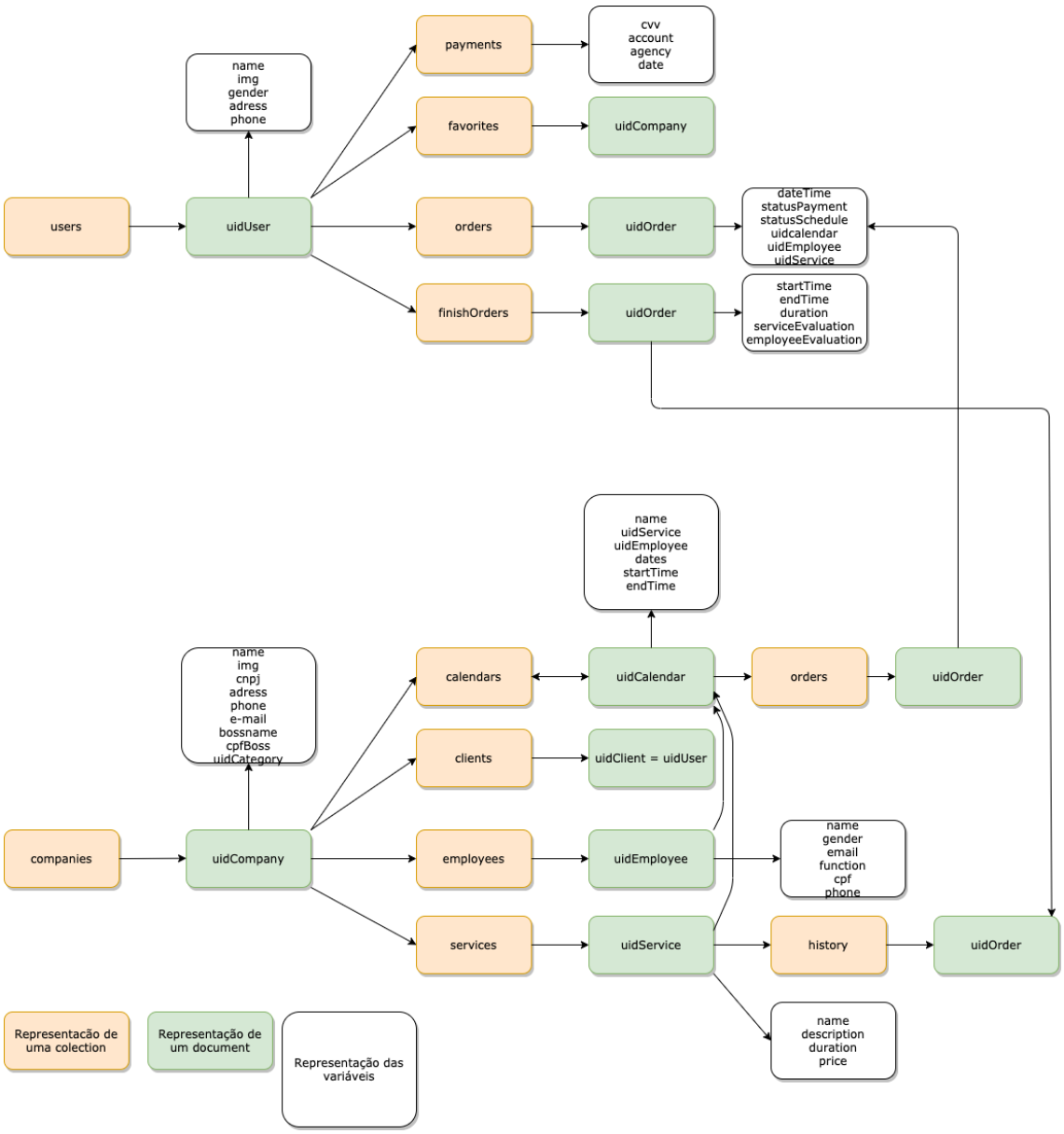

Figura 16. Diagrama explicativo do Banco de Dados.

Apêndice I – Demais interfaces da aplicação do prestador de serviço.

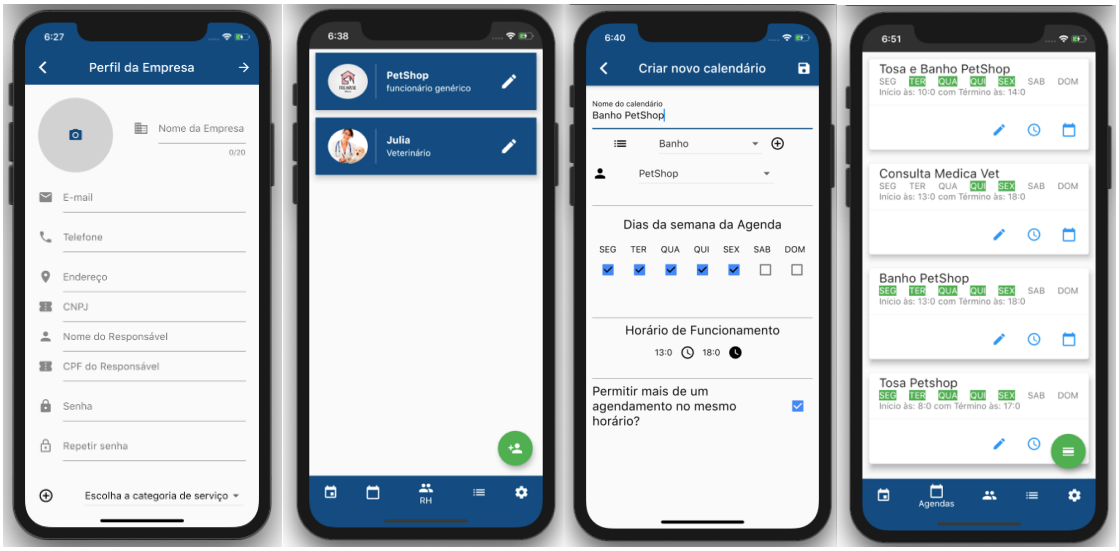

Figura 17. Interfaces: Criação de perfil; Gestão Funcionários; Criação de calendário; Configuração de calendários.

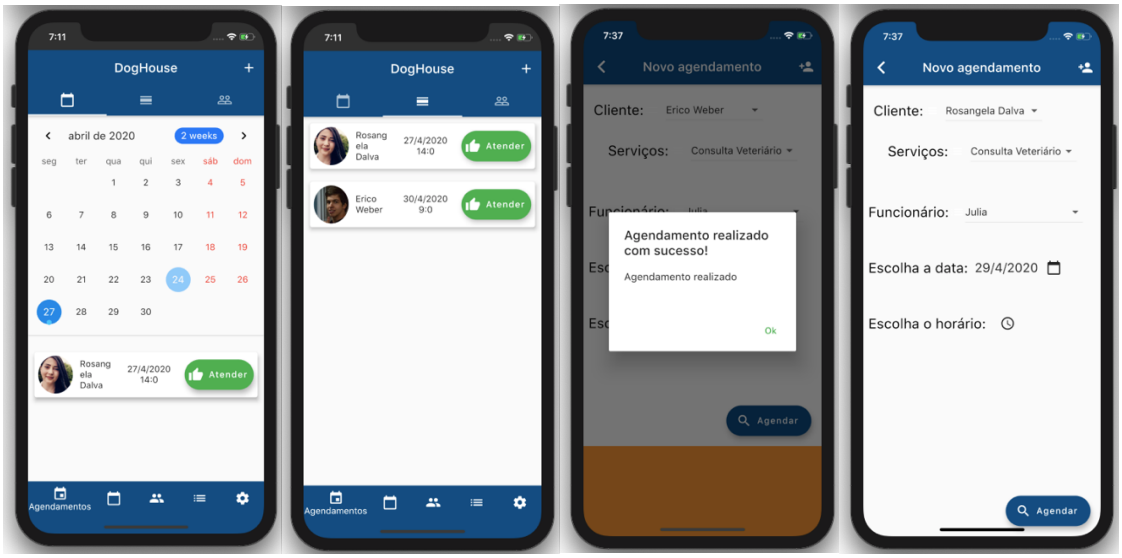

Figura 18. Interfaces: Visualização de agendamentos; Visualização de todos agendamentos; Finalização de um agendamento; Realização do agendamento de um cliente.

Apêndice J – Demais interfaces aplicação cliente

Figura 19. Interfaces: Edição de perfil; Edição meio de pagamento; Escolha criação de conta; Favoritos

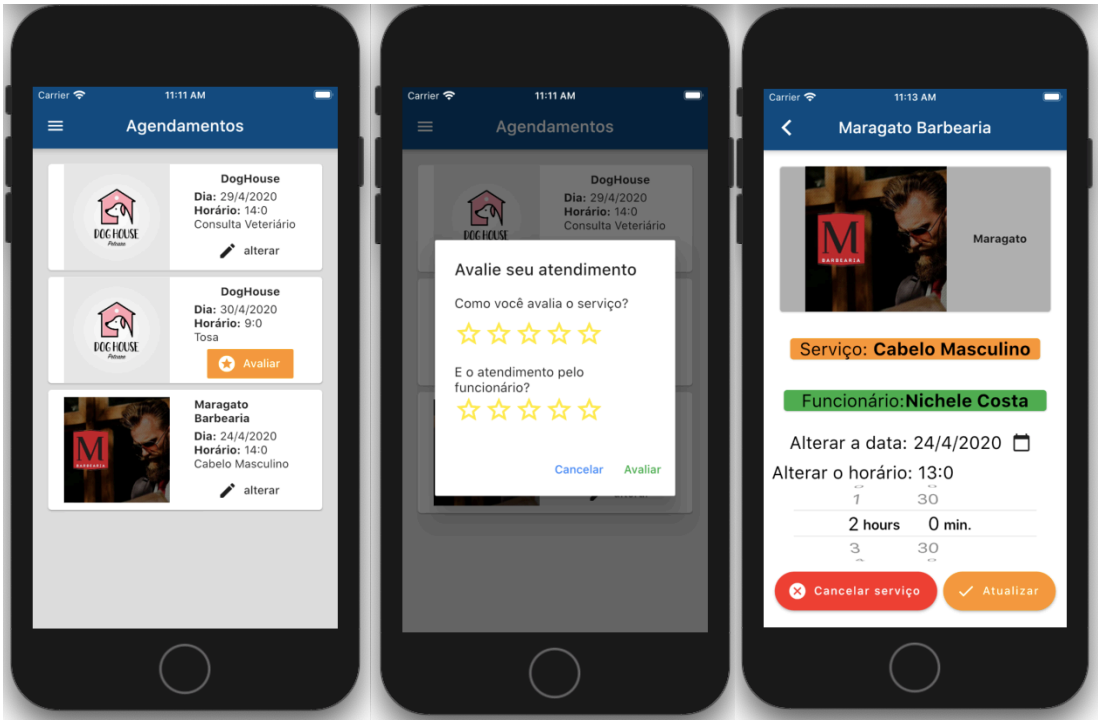

Figura 20. Interfaces: Agendamentos; Avaliação serviço; Alteração agendamento