RECA
REVISTA ELETRÔNICA DE COMPUTAÇÃO APLICADA
VOL. 1 NRO. 1 2020

APP MarqFacil:
GERENCIAMENTO DE AGENDA DE CLÍNICAS ESTÉTICAS

José Hamilton Martins Leite
Graduando em Engenharia de Software – Uni-FACEF
josehamiltonmartinsleite@gmail.com

Débora Pelicano Diniz
Mestre em Computação – Uni-FACEF
deboradiniz@facef.br

Resumo

Este artigo tem como finalidade apresentar o desenvolvimento de um aplicativo que irá facilitar o agendamento de horários em clínicas estéticas na área da saúde. Facilitará, também, a vida de muitas pessoas que não irão necessitar sair de onde estão para verificar os procedimentos e produtos que algumas clínicas fornecem e vendem. O cliente também poderá consultar as suas avaliações feitas na clínica por meio do aplicativo. O projeto é uma aplicação para dispositivos móveis, como por exemplo, celulares, tablets, entre outros dispositivos. Foi utilizado o *Framework React Native* para o desenvolvimento *FrontEnd*; e para o *BackEnd*, utilizou-se a linguagem de programação Java com o uso do *Framework Spring Boot*. Na criação dos modelos conceitual, lógico e físico do Banco de Dados foram utilizadas as ferramentas Br Modelo (desenvolvimento do diagrama de entidade-relacionamento), DB Designer (desenvolvimento do diagrama de entidade-relacionamento) e o Sistema Gerenciador de Banco de Dados (SGBD) *MySQL* e o *MySQL Workbench*. A parte de diagramas e modelagem de todo o software foi realizada utilizando a ferramenta Enterprise Architect, e para a prototipação de telas foi utilizado o Adobe XD. No artigo estão descritas as funcionalidades da aplicação, quais os benefícios para as clínicas estéticas, e o passo a passo do desenvolvimento do aplicativo, que possui níveis de acesso às funcionalidades sendo divididas em três tipos de usuários: administrador, colaborador e paciente.

Palavras-chave: Aplicativo. Agendamento. Procedimentos. Produtos. Clínicas estéticas.

Abstract

This article aims to show the development of an application that will facilitate scheduling times at aesthetic clinics in health area. It'll facilitate life off people that will not have to go away from where they are to verify procedures or products that clinics offer/sells. Costumers will can consult their evaluations done by clinics with the app. Project is an application made to mobile devices like cellphones, tablets among others. Framework React Native was used for develop FrontEnd and to BackEnd programming language Java was used with Framework Spring Boot. The creation of conceptual, logical and physical database models was done using Br Model (development of entity-relationship diagram), DB Designer (development of entity-relationship diagram) and Database system manager MySQL and MySQL Workbench. All software diagrams and models were done using Enterprise Architect tool and

Adobe XD was utilized to prototype screens. At the article were described functionalities of application, what are the benefits for aesthetic clinics and step-by-step of application development that have three user types: manager, collaborator and patients.

Keywords: *Application. Scheduling. Procedure. Products. Aesthetic Clinics.*

1 Introdução

Muitos prestadores de serviço realizam as atividades de gerenciamento de agenda e controle de avaliações de seus pacientes de forma manual, o que pode ocasionar perdas de informação, demora no processo e, consequentemente, descontentamento dos clientes. Mas, buscando melhorias e agilidade nos processos, muitos profissionais estão preferindo utilizar sistemas que possuem o gerenciamento de agendas e controle de avaliações de pacientes a partir de plataformas que fazem os processos automatizados.

Existem muitas plataformas e sistemas já desenvolvidos que auxiliam esse processo, mas, em geral, eles são genéricos, ou seja, a empresa, ou os prestadores de serviço, é que deve se adaptar ao sistema para utilizá-lo. Além disso, poucos são os sistemas que são voltados especificamente em clínicas de estética. Os profissionais da área estética comentam que não conseguem encontrar um software adequado às suas necessidades.

Assim, o trabalho apresentado nesse artigo teve como objetivo desenvolver um software para auxiliar no gerenciamento de agenda e controle de avaliações de pacientes de clínicas estéticas, e auxiliar na divulgação dos serviços e produtos oferecidos por essas clínicas, ou seja, criar um MVP (Mínimo Produto Viável) com as principais funcionalidades e que seja utilizado em dispositivos móveis, como celulares e tablets, mas almeja-se atribuir algumas funcionalidades em um sistema WEB no futuro.

Foram utilizadas técnicas da engenharia de software e metodologia ágil para organização do projeto, de forma a desenvolver um sistema de qualidade.

2 Referencial Teórico

Esta seção é dedicada a apresentar as tecnologias e ferramentas utilizadas no desenvolvimento do projeto MarqFacil.

2.1 Integração de API's ou microsserviços

Integração pode ser definida como sendo a combinação de conceitos que são reunidos para formarem um conceito único, e esses conceitos podem ser entendidos como sendo as API's (*Application Programming Interface*) e os microsserviços que são combinados entre si para que se tornem um sistema. Porém, os sistemas devem possuir uma web semântica que é entendida como o compartilhamento de dados para que haja a integração de recursos (FEITOSA, 2005).

A integração faz com que os sistemas de informações trabalhem de forma automatizada. Essa integração auxilia os sistemas na otimização da interação de informações vinda de diferentes sistemas. A interoperabilidade é algo que os sistemas devem possuir para que a integração seja mais eficiente, o que melhora a

qualidade e eficiência dos processos (CUNHA, 2005).

Em algumas empresas é mais difícil possuir todos os setores trabalhando juntos e, com o auxílio da integração de todos os módulos, se torna muito mais efetivo ter dados com consistência e qualidade. A integração faz com que as transações de dados entre os diferentes setores sejam mais seguras e rápidas (RICCIO, 2001).

Em uma empresa, a utilização da integração faz com que seja utilizado apenas um sistema para todos os setores da empresa, esse sistema pode ser chamado de ERP (Sistema Integrado de Gestão Empresarial), com o armazenamento de todos os dados corporativos podendo ser centralizado em um único banco de dados. Dessa forma, os funcionários da empresa conseguem visualizar, em tempo real, uma atividade passando pelos setores da empresa. Essa integração ajuda na questão de controle e integridade dos dados e faz com que não haja redundância de informações (MENDES; ESCRIVÃO FILHO, 2002).

2.2 Estética

A palavra estética tem origem na língua grega e seu significado está relacionado a sensação e percepção. De acordo com o que foi dito por Baumgarte (apud RIBEIRO FILHO, 2016), estética seria o conhecimento sensório da perfeição.

A estética está ligada ao que é belo, e em uma elaboração feita pelo filósofo Aristóteles, essa elaboração diz respeito a três palavras: o bom, o belo e ao verdadeiro. O verdadeiro está ligado a lógica, o bom está relacionado a ética, e o belo está caracterizado ao que deve ser contemplado (RIBEIRO FILHO, 2016).

A sociedade, com o passar dos anos, foi criando alguns padrões de beleza e, caso o indivíduo não esteja dentro destes padrões, ele se sente pressionado, e tenta encontrar modos para se adequar ao padrão da sociedade para aumentar a sua autoestima. Os procedimentos estéticos estão entre esses modos pois permite que a pessoa se sinta bem, melhor e mais bonita diante a sociedade (FERRAZ; SERRALTA, 2007 apud BORBA; THIVES, 2011).

Branden (2009, p. 13) diz que “a autoestima é a confiança em nossa capacidade para pensar e enfrentar os desafios da vida”.

No meio estético, existem muitos procedimentos e tratamentos, alguns dos procedimentos mais utilizados são: as massagens, que podem ser para relaxamento muscular ou modeladoras de corpo (FRANÇA et al, 2016); e a criolipólise, que permite eliminar a gordura localizada que o indivíduo possui (SILVA; MERCADO, 2015).

2.3 Ferramentas e Procedimentos

Nesta seção estão comentadas as ferramentas e procedimentos utilizados durante o desenvolvimento do projeto.

2.3.1 React Native

O *React Native* é um *framework* que foi desenvolvido pelo Facebook e foi construído a partir do *React*. O desenvolvimento pode ser feito para gerar aplicativos em Android ou IOS, pois se trata de um *framework* multiplataforma, que

utiliza componentes nativos de cada linguagem para a criação das aplicações para dispositivos móveis (BECKER, 2020).

Ainda de acordo com Becker (2020), o que diferencia o *React Native* dos outros *frameworks* é que o código gerado por ele pode ser convertido para a linguagem nativa do sistema operacional, como por exemplo, no Android, a linguagem Java e no IOS, a linguagem *Objetive-C*.

Uma das vantagens em utilizar este *framework* no desenvolvimento é que a experiência na utilização do aplicativo pelo usuário é mais fluída, isto quer dizer, os carregamentos são mais rápidos e não ocorrem muitos travamentos na utilização. Além disso, permite a integração com muitas funções do celular, como câmera e giroscópio, o que permite uma melhor performance.

O *React Native* utiliza o JSX (Extensão da Sintaxe do JavaScript) para escrever estruturas de componentes na UI (interface do usuário). Esse tipo de extensão é parecido com o XML (Linguagem Extensível de Marcação Genérica), porém, permite escrever estruturas do tipo HTML (Linguagem de Marcação de Hipertexto) em conjunto com códigos em *JavaScript* e, com o auxílio do Babel (ISAC JUNIOR, 2020), transformar tudo em código *JavaScript*.

Alguns exemplos de aplicativos famosos que foram criados a partir do *React Native* são (WARCHOLINSKI, 2020):

- *Facebook* - tinha por objetivo tornar a inicialização de painéis de eventos mais ágeis;
- *Facebook Ads* - escolheu o *React Native* para tornar suas interfaces mais limpas com UX (experiência do usuário) intuitivo e navegação simples;
- *WalMart* - optaram pela plataforma para que os seus usuários tenham uma melhor experiência ao utilizarem o aplicativo;
- Wix - utiliza o *framework* porque é mais ágil e rápido do que outros que existem no mercado.

2.3.2 Android

O Android teve seu surgimento no ano de 2003 e, no começo, sua principal ideia era criar um programa para câmeras digitais. Os criadores viram que o programa não teria um mercado bom no ramo de câmeras digitais e então resolveram partir para a área do mercado mobile (MEYER, 2015).

Ainda segundo o autor, a partir desse momento, a equipe que criou o Android resolveu que iria oferecer um sistema gratuito que poderia ser utilizado por qualquer pessoa, o projeto era *open source*, baseado no Kernel Linux. Em 2005, o Android foi adquirido pela *Google* e, após a compra, foi criada a *Google Mobile Division* que tinha o objetivo de ser a divisão de pesquisa em tecnologia móvel.

Apenas no ano de 2007 que foi surgir o primeiro Android, com versão beta, que foi disponibilizado para o público utilizar. No ano de 2008 foi desenvolvida e lançada a primeira versão comercial do Android (que era o Android 1.0 - Astro), que possuía poucas funcionalidades, como por exemplo, o *android market* (loja de aplicativos Android), navegador, pastas, acesso à internet, integração a aplicações do google, reprodução de mídias, notificações, ligações por comandos de voz e o suporte a câmera, Wi-Fi e bluetooth (MEYER, 2015).

Em 2014 surgiu o primeiro lançamento específico para aparelhos vestíveis, que são os *Wereables*, como por exemplo, relógios que possuem na sua estrutura um *android* próprio, com algumas funcionalidades como suporte ao GPS e reprodução de música offline. A versão do *android* que foi desenvolvida para aparelhos vestíveis foi a 4.4W (MEYER, 2015).

2.3.3 Metodologia Ágil

Utilizar uma metodologia definida e padronizada pode ajudar muito durante o desenvolvimento de um *software*, caso os envolvidos no projeto tenham pelo menos um conhecimento básico de como funciona o processo da metodologia que será utilizada. As metodologias ágeis são muito utilizadas atualmente, utilizando técnicas de gestão dinâmicas nas etapas de criação de sistema, com auxílio de procedimentos e ferramentas para a criação do projeto (INVENTTI, 2018).

Com o uso de metodologias ágeis a equipe corre menos riscos no desenvolvimento, existe uma melhor integração entre os membros do time, o cliente está mais relacionado com o projeto e os profissionais de diferentes áreas conseguem trabalhar juntos, como por exemplo, os profissionais de negócios juntamente com os de tecnologia (BISSI, 2007).

O trabalho apresentado neste artigo foi desenvolvido seguindo o *framework Scrum*, com a utilização do sistema *Kanban* para uma melhor visualização das atividades a serem desenvolvidas e, portanto, ambos serão comentados a seguir.

O *Scrum* é um *framework* ágil cujo principal objetivo é fazer com que os projetos sejam criados e entregues com um alto valor agregado e com um prazo menor. Para utilizar o *Scrum* é necessário possuir uma equipe, com papéis bem divididos. A equipe é dividida nos seguintes papéis ((INVENTTI, 2018):

- *Scrum Master*: é um facilitador no desenvolvimento do projeto, é a pessoa que lidera toda a equipe, faz as mudanças necessárias para que o projeto seja entregue em tempo, ajuda o time a remover os impedimentos que surgirem durante o desenvolvimento;
- *Product Owner*: ele faz o papel do cliente dentro da equipe que está desenvolvendo o projeto;
- *Development Team*: Essas são as pessoas que são especializadas em desenvolver as tarefas do projeto, é o famoso de time de desenvolvimento.

O *Kanban* é um “registro visual” das etapas que estão sendo realizadas durante todo o desenvolvimento do projeto. O uso do sistema *Kanban* faz com que os profissionais fiquem mais focados no projeto com tarefas limitadas para que haja o equilíbrio nas demandas, com isso a equipe consegue melhorar o seu rendimento e faz entregas mais rápidas e de qualidade (MARIOTTI, 2012).

Mariotti (2012) também comenta que com a utilização desse sistema se torna mais fácil para a equipe definir as datas de entrega. Os profissionais não se sentem pressionados, conseguindo fazer as tarefas de acordo com a demanda, e isso faz com que as pessoas relacionadas ao projeto vejam o quanto que a equipe já evoluiu durante as etapas do desenvolvimento do projeto.

Para utilizar o *Kanban* é necessário que o time determine quais são as tarefas que serão feitas durante um determinado período; essas tarefas são inseridas

no início, na coluna *backlog*. Enquanto a equipe estiver desenvolvendo, os desenvolvedores vão mudando a tarefa no quadro *kanban* para indicar em que etapa que está o desenvolvimento da tarefa em questão. Após o período determinado, a equipe deve ter finalizado todas as tarefas e então é feita uma nova reunião para determinar as novas tarefas que deverão ser feitas até a finalização do projeto.

A quantidade de tarefas que são determinadas no *backlog* são limitadas em uma quantidade específica, para que a equipe não fique sobrecarregada de serviços a serem desenvolvidos, para que o trabalho seja desenvolvido com tranquilidade e qualidade.

2.3.4 WEB Services REST

Um *WEB Service* é um sistema com interoperabilidade, isto quer dizer que foi desenvolvido para que fosse possível fazer a interação de diferentes aplicações entre si, como por exemplo, fazer uma comunicação de um sistema com outros sistemas desenvolvidos em diferentes plataformas, e cada uma dessas aplicações podem possuir uma própria linguagem, com uma 'tradução' para uma linguagem que seja universal entre todas, podendo citar essa linguagem universal como sendo o formato *JSON* (DAL MORO; DORNELES; REBONATTO, 2011).

A palavra REST tem como significado *Representational State Transfer*, que traduzido para o português é a Transferência de Estado Representativo. O padrão REST é um meio de desenvolver aplicação de *web services*, tendo surgido na tese de Roy Fielding (SAUDATE, 2014).

Existem alguns princípios do estilo em aplicações *REST* que devem ser seguidos, como por exemplo, se possível, as aplicações devem ser divididas em cliente-servidor. Com isso, todos os serviços disponíveis que o cliente pode acessar estarão no servidor e só podem ser acessados a partir de requisições, podendo ser retornadas ou não. Outra característica que se pode citar é em relação a interface uniforme, que diferencia a arquitetura *REST* das demais. Nessa interface existem quatro métodos básicos mais comuns que são: os métodos *GET* (utilizado para retorno de informações); *POST* (para envio de informações); *PUT* (para substituir informações) e *DELETE* (para remoção de informações) (DAL MORO; DORNELES; REBONATTO, 2011).

No padrão *REST* existem algumas boas práticas que podem ser seguidas, como por exemplo: o uso adequado de *URL's* (*Uniform Resource Locator*); utilizar algumas estruturações padronizadas para retornar erros ou falhas do sistema; o uso adequado dos cabeçalhos *HTTP* e a interligação de diferentes recursos (SAUDATE, 2014).

2.3.5 Spring Framework

Johnson (2004) comenta que o *Spring Framework* é uma plataforma Java que auxilia no desenvolvimento de aplicativos Java, fornecendo a parte de suporte a estruturação do aplicativo. Ele prepara a parte da infraestrutura do projeto fazendo com que o desenvolvedor consiga focar na parte do desenvolvimento do aplicativo solução. O autor comenta também que é muito utilizada nas empresas, sendo uma solução leve que conta com várias funcionalidades que o torna mais cogitado, além de ser muito simples para o desenvolvedor criar projetos utilizando o *Spring Framework*, isolando as dependências do código base.

O ecossistema do *Spring* é muito amplo e possui diferentes serviços,

como por exemplo, serviços na área de computação em nuvem, *Big Data*, persistência de dados sem esquema, uma programação reativa e tem a opção de desenvolvimento de aplicações no lado do cliente (WALLS, 2016).

2.3.6 MySQL

O MySQL surgiu na década de 90 e foi desenvolvido por David Axmark, Allan Larson e Michael “Monty” Widenius. É um sistema gerenciador de banco de dados (SGBD) que pode ser utilizado em diferentes aplicações, sendo elas pequenas, médias ou grandes aplicações (MILANI, 2007).

É um banco de dados relacional, com licença dupla, sendo uma delas de software livre. Pode ser utilizado para aplicações de tipo *desktop* e web e, como possui uma boa compatibilidade, pode ser utilizado em diferentes sistemas operacionais, como exemplo, Linux, Unix, Mac OS X Server e Windows. É um programa que foi desenvolvido com as linguagens C e C++, e isso faz com que sua portabilidade seja muito grande (MILANI, 2007).

A linguagem utilizada para execução de códigos é a *Structured Query Language* (SQL) que é uma linguagem textual para interagir com banco de dados relacionais. É uma linguagem que possui uma velocidade muito grande para execução de códigos e é bem otimizada (ANLEY, 2002).

3 Contexto Empreendedor do Projeto

Nesta seção está apresentada a questão do Modelo de Negócio Canvas, que tem por objetivo fazer com que seja entendido se o modelo de negócios terá uma visão adequada, se a proposta criada irá gerar e possibilitar diferentes oportunidades. A partir deste modelo, pode-se entender a viabilidade econômica do projeto que está sendo desenvolvido e entender se a ideia irá ter um bom retorno (OSTERWALDER; PIGNEUR, 2010).

O Modelo de Negócio Canvas que foi desenvolvido no projeto MarqFacil pode ser visualizado na Figura 1. Durante o desenvolvimento foram seguidas algumas etapas para desenvolvimento.

Na primeira etapa foram executadas as seguintes atividades: (i) SEGMENTO DE CLIENTE, na qual foi determinado o público-alvo do software, ou seja, as Clínicas Estéticas; (ii) PROPOSTA DE VALOR, em que foram avaliados os benefícios do software, sendo definidos como principais a facilidade para controle de atendimentos e a organização de agenda; (iii) CANAIS, determinando os meios de distribuição do produto, que no caso do aplicativo definiu-se que será disponibilizado para uso na plataforma mobile (Android); (iv) RELAÇÃO COM O CLIENTE, na qual se informa os meios de comunicação que são fornecidos ao cliente, e para esse negócio foi determinado o treinamento para uso do software e os serviços de suporte.

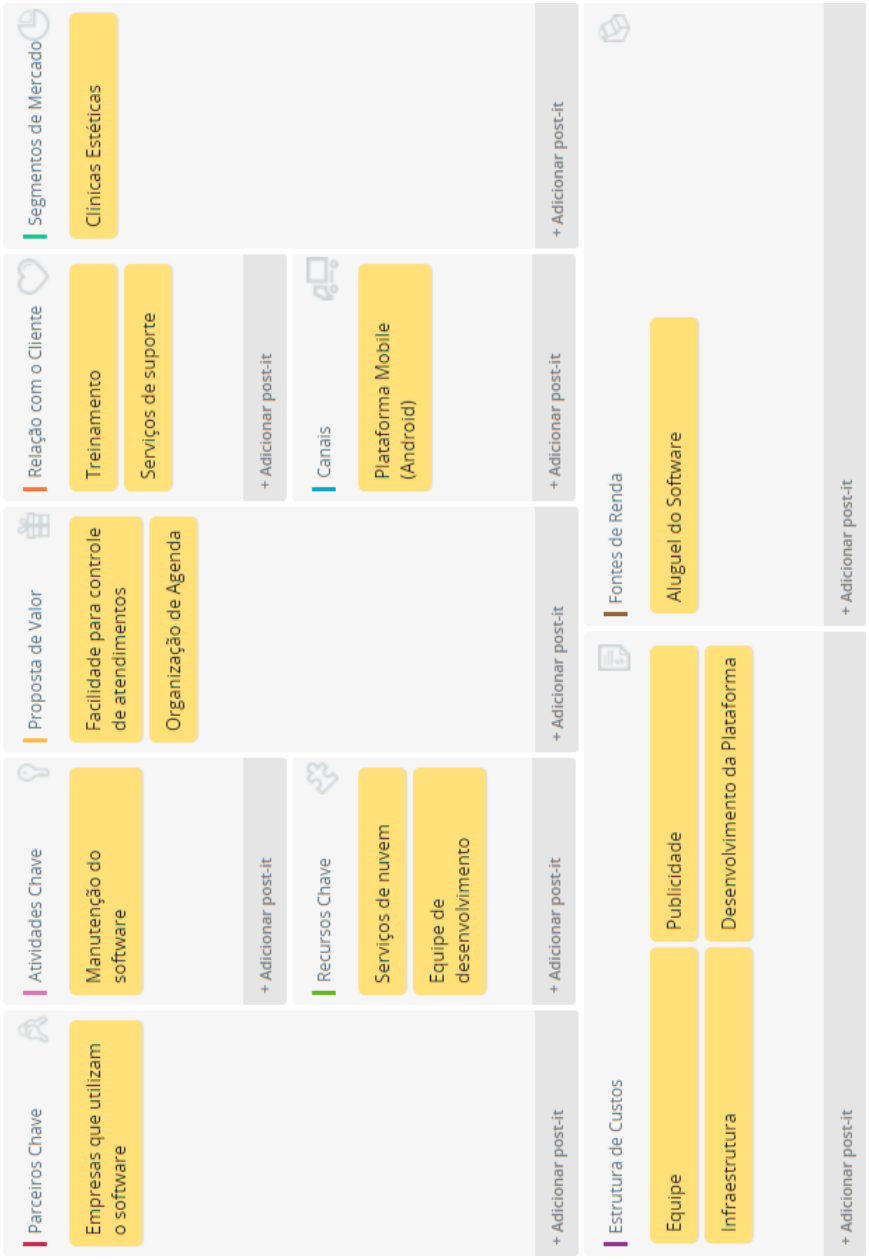
Na segunda etapa devem ser determinadas quais são as FONTES DE RENDA, ou seja, quais os meios de lucro que o software irá possuir, que nesse caso a única definida é o aluguel do software.

Na terceira etapa foram determinados os RECURSOS CHAVE, que são os recursos necessários para manter o funcionamento do software, que serão serviços de nuvem e uma equipe de desenvolvimento; as ATIVIDADES CHAVE, que são as atividades importantes para manter o software, que no caso será apenas a

manutenção do software; e também PARCEIROS CHAVE que irão auxiliar a entrega de valor do software, que serão apenas as empresas que utilizarão o software.

Por fim, é determinada a ESTRUTURA DE CUSTOS, ou seja, o que será preciso possuir para que se mantenha o software em funcionamento, e nesse caso, os custos serão referentes à equipe de desenvolvimento, publicidade, infraestrutura e o desenvolvimento da plataforma.

Figura 1 Canvas do projeto MarqFacil



Fonte: Autoria Própria

4 Resultados da Análise e Projeto

Nesta seção estão comentadas as atividades realizadas durante as etapas de Análise e Projeto do processo de desenvolvimento do sistema MarqFacil, com exemplos dos artefatos produzidos. A documentação completa está disponível no *GITHUB* (LEITE, 2020).

4.1 Elicitação de Requisitos

Essa foi a primeira atividade do projeto, pois para se construir um sistema é necessário entender o que o cliente deseja, extraindo informações sobre suas necessidades e desejos. Sommerville (2011) comenta que o levantamento e a análise de requisitos devem envolver diferentes tipos de pessoas dentro da organização. Ainda comenta que, geralmente, os requisitos de um sistema são classificados em três segmentos: funcionais, não funcionais (também chamados de requisitos de qualidade) e de domínio.

Para o desenvolvimento do sistema MarqFacil os requisitos foram divididos em duas partes, sendo elas os Requisitos Funcionais (uma parte da descrição desses requisitos pode ser visualizada na Tabela 1) e os Requisitos de Qualidade (que podem ser visualizados na Tabela 2).

Os requisitos foram decididos a partir de reuniões que foram feitas com o cliente, e que ajudaram o analista a entender quais as funcionalidades deveriam existir no software para atender as necessidades do cliente, com sucesso. Esta documentação dos requisitos do software foi uma das atividades do backlog da segunda *sprint*.

Tabela 1 Requisitos Funcionais

Requisitos Funcionais	Descrição
[RF 01] – Gerenciar empresa	O sistema deve possuir um ambiente para fazer o cadastro de empresas, no caso a clínica de estética, contendo a Razão Social, CPF / CNPJ, Nome Fantasia, Endereço completo (Rua, Bairro, Cidade, Complemento, Número, CEP, Estado). Os campos obrigatórios para o cadastro são: Razão Social, CPF / CNPJ, Nome Fantasia, Telefone, Email, Rua, Número, CEP, Bairro, Cidade e Estado. Após preenchidos esses campos, o sistema já deve permitir o cadastro de um usuário principal, que terá acesso total ao sistema. Para esse cadastro é necessário informar os seguintes campos: Nome Completo, Usuário, Senha, RG, CPF, Telefone, Celular, E-mail. Após cadastrada a empresa no sistema, é enviado um boleto para o profissional que cadastrou a empresa, para que, após seu pagamento, possa utilizar o sistema. Caso seja necessário fazer alguma alteração nos dados da empresa, só poderá ser feita pela empresa de software pelo módulo gerencial e a exclusão de alguma clínica só poderá ser feita pela empresa de software.
[RF 02] - Gerenciar usuários	O sistema deve permitir cadastrar pacientes e colaboradores em um mesmo ambiente, pois as informações de cadastro são as mesmas para ambos. Assim, devem ser informados: Nome Completo, RG, CPF, Data de Nascimento, Endereço completo (Rua, Bairro, Cidade, Complemento, Número, CEP, Estado), Telefone, Celular, E-mail, Usuário e Senha. Os campos obrigatórios são: Nome completo, RG, CPF, Celular, E-mail, Rua, Número, CEP, Bairro, Cidade, Estado. Para diferenciar entre paciente e colaborador, o sistema deve ter, no final, a opção de marcar se o cadastrado é um colaborador ou um paciente. Só deve ser possível cadastrar os pacientes e colaboradores caso o usuário administrador esteja logado no sistema. O usuário administrador é o único com permissão para alterar ou excluir um usuário.

...	
[RF 06] – Gerenciar avaliações	As avaliações só podem ser cadastradas pelo usuário administrador, e após o paciente passar por algum procedimento na clínica estética, em que o colaborador realizará a avaliação e passará as informações para o usuário administrador. Para a inclusão de uma avaliação, escolhe-se o paciente a quem ela pertence (previamente cadastrado) e insere uma descrição com as informações, uma imagem do paciente e o sistema deve incluir a data de inserção automaticamente. Os campos obrigatórios são apenas o paciente e uma descrição da avaliação.
[RF 07] – Agendar Horários	Para agendar um horário, o paciente, o colaborador e o procedimento a ser realizado devem estar previamente cadastrados. Devem ser preenchidos os seguintes campos: paciente, colaborador, horário e os procedimentos. Quando o usuário for cadastrar o atendimento e não informar o tempo de atendimento, o sistema dará a opção de somar o tempo de cada procedimento (informação já registrada no cadastro do procedimento) ou pegar o maior tempo de algum procedimento. O sistema deve preencher o campo status com AGENDADO, automaticamente. A mudança de status está especificada no requisito [RF 08].

Fonte: Autoria própria

Tabela 2 Requisitos de Qualidade

Requisitos de Qualidade	Descrição
[RQ 01] – Excluir dados apenas se não estiverem sendo utilizados	Qualquer exclusão deve ser verificada no sistema se irá ocasionar algum problema no futuro em outra parte do sistema. Se o dado que for excluído não tiver relacionado a nenhum outro dado no sistema pode ser excluído, caso contrário o sistema não irá permitir a exclusão.
[RQ 02] - Segurança	O sistema pode ser acessado por qualquer pessoa, mas apenas para visualização dos produtos e procedimentos das empresas cadastradas. Para a utilização das outras funcionalidades, a pessoa deve ser cadastrada, observando os seguintes níveis: • **Administrador** – tem acesso a todas as funcionalidades do sistema; • **Colaborador** – tem acesso às funcionalidades especificadas pelos requisitos funcionais [RF 08], [RF 09], [RF 10], [RF 12], [RF 13], [RF 14], [RF 15], [RF 17], [RF 18]; • **Paciente** - tem acesso às funcionalidades especificadas pelos requisitos funcionais [RF 09], [RF 10], [RF 11], [RF 12], [RF 13], [RF 14], [RF 15], [RF 17], [RF 18].
[RQ 03] - Usabilidade	O sistema deve ser simples e intuitivo em seu uso. Para isso é importante que as telas sigam um padrão de *layout,* os botões possuam nomes significativos e com cores bem informativas nos botões (verde e vermelho) e com cores neutras nas telas de todo o software.

Fonte: Autoria própria

4.2 Diagrama de Caso de Uso

Boch, Rumbaugh e Jacobson (2006) definem que os casos de uso captam o comportamento pretendido do sistema, sem a necessidade de especificar como é a implementação desse comportamento. Os autores também definem o diagrama de casos de uso como um conjunto de casos de uso, seus atores e relacionamentos.

Como já comentado anteriormente, foi utilizado o Scrum para o desenvolvimento do projeto, e a tarefa de criação do Diagrama de Casos de Uso completo do sistema (Figura 2) fez parte do *backlog* da segunda *sprint.* Durante essa atividade verificou-se a necessidade de criar dois módulos: Gerencial e MarqFacil.

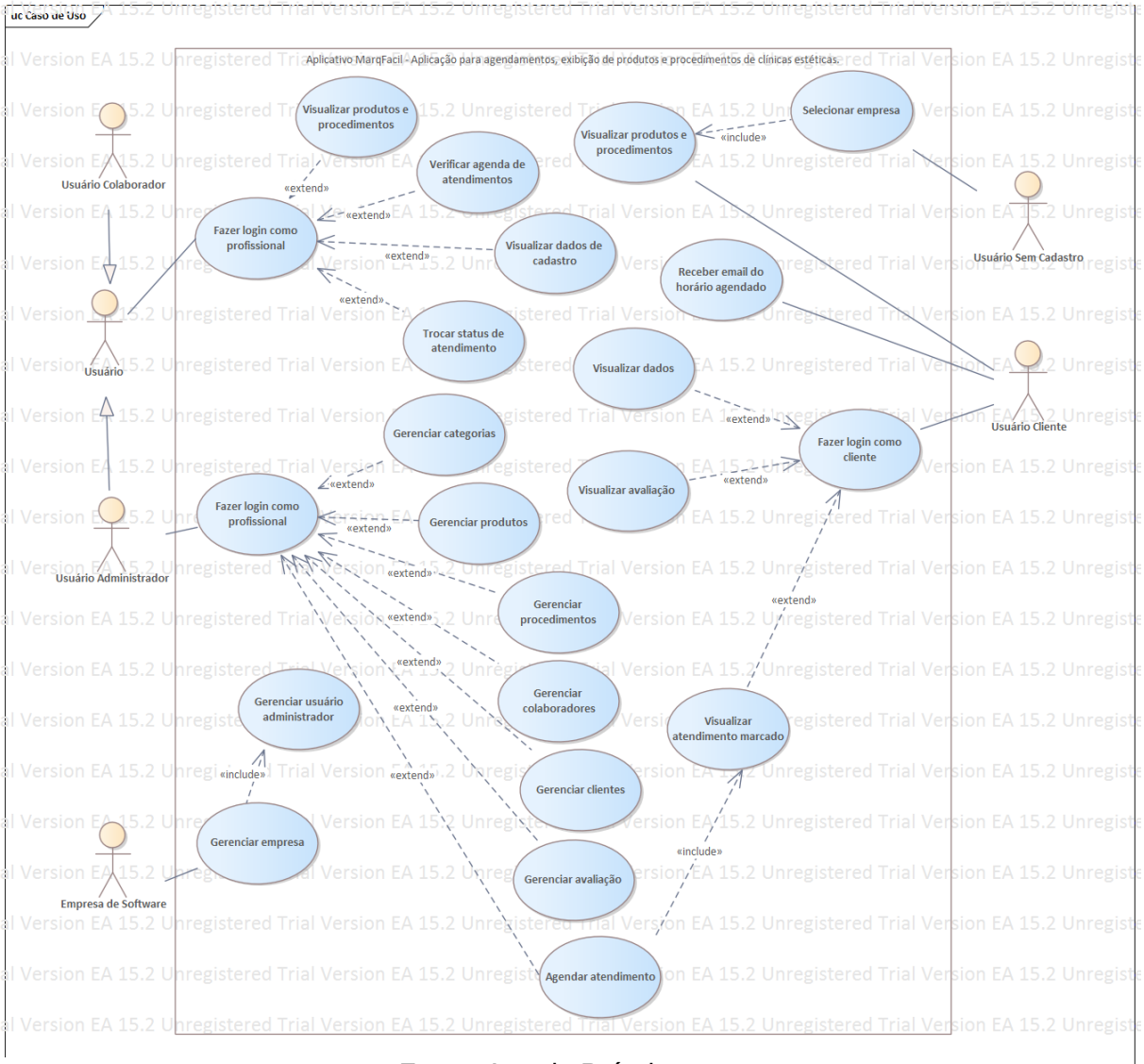
O diagrama de caso de uso foi desenvolvido seguindo as especificações passadas pelo cliente durante o levantamento de requisitos. Neste diagrama é feito um estudo de todos os requisitos do software, e a partir desses requisitos funcionais e de qualidade foram criadas todas as funcionalidades que constituem o software.

4.3 Modelo Relacional

De acordo com Elmasri e Navathe (2019), o modelo relacional representa o banco de dados como uma coleção de relações e estas podem ser representadas como tabelas de valores, sendo que todos os valores em uma coluna devem ser do mesmo tipo de dados. Os autores continuam comentando que cada linha da tabela se chama tupla, o cabeçalho da coluna é o atributo e a tabela em si é chamada de relação.

Considerando os requisitos solicitados pelo cliente para o sistema aqui apresentado, em um primeiro momento foi pensado criar 3 tabelas: Usuários, Atendimentos e Avaliações. Mas, levando em consideração as Formas Normais (ELMASRI; NAVATHE, 2019) e boas práticas para a definição de Banco de Dados, outras tabelas foram sendo criadas, como por exemplo, a Tabela ProcedimentosAtendimentos que é a tabela responsável por relacionar os procedimentos que estão ligados aos atendimentos. O diagrama completo está apresentado na Figura 3.

Figura 2 Diagrama de Casos de Uso Completo do Sistema



Fonte: Autoria Própria

A criação do banco de dados foi elaborada buscando a facilidade para futuras manutenções do sistema que envolvam modificações na estrutura das tabelas e seus relacionamentos. Algumas tabelas foram criadas de forma a permitir que o sistema filtre os dados para realizar pesquisas mais rápidas na base de dados, como por exemplo, quando o usuário pesquisar o atendimento, poderá filtrar pelos nomes do colaborador ou do paciente.

5 Resultados da Implementação

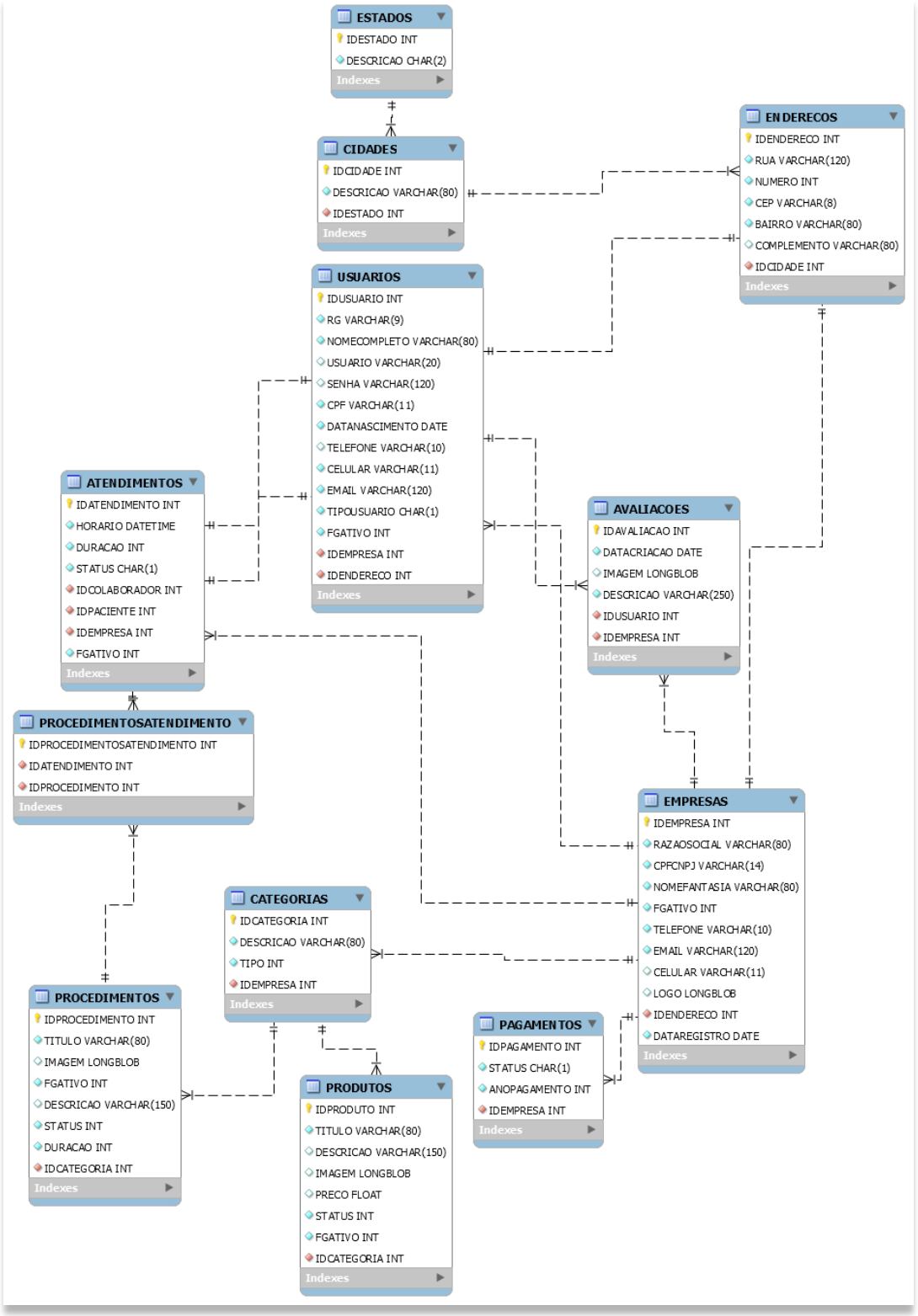
Nesta seção estão comentadas as partes que compõem o sistema MarqFacil, que é um sistema que possui dois módulos: o Gerencial e o MarqFacil.

5.1 Integração das Aplicações

Durante a execução de todo o projeto foram desenvolvidas duas aplicações *FrontEnd,* utilizando o React Native, que são os módulos: Gerencial e o MarqFacil; que irão ser utilizados pelo cliente. Também foi criada uma aplicação

BackEnd que foi desenvolvida com a utilização da linguagem Java, utilizando o Framework Spring Boot, e essa aplicação é a chamada API, a qual foi construída para que fosse possível executar as operações com o banco de dados.

Figura 3 Modelo Entidade-Relacionamento do Sistema MarqFacil



Fonte: Autoria Própria

A aplicação *BackEnd* foi criada seguindo o padrão de desenvolvimento das API's RESTful com os principais métodos HTTP's que são os seguintes: *get* (exibe os dados), *post* (envia os dados), *put* (altera os dados) e *delete* (remove os dados). Na Figura 4 pode-se identificar um método que foi criado na API para remoção de um usuário.

Figura 4 Método para remover um usuário

```
@ApiOperation(value = "Rota para remover um usuário.")
@DeleteMapping("/{id}")
public ResponseEntity<Response<String>> removerUsuario(@PathVariable("id") Integer idusuario) {
    Response<String> response = new Response<String>();
    try {
        log.info("Rota para remover um usuário.");
        this.usuarioService.removerUsuario(idusuario);
        return ResponseEntity.noContent().build();
    } catch (Exception ex) {
        response.getErrors().add("Erro ao excluir o dado.");
        return ResponseEntity.badRequest().body(response);
    }
}
```

Fonte: Autoria Própria

O código exibido na Figura 4 possui um método HTTP *delete* que é capaz de fazer a remoção a partir do id (identificador) do usuário. Caso o usuário exista e seja feita a remoção com sucesso, o retorno será uma resposta de status 204 (*No content*) que significa que ocorreu corretamente e não existe um conteúdo para retorno; caso ocorra erro, a API retorna o status 400 (*Bad request*) com a mensagem de erro: "Erro ao excluir os dados".

Esse método citado é um dos vários métodos que foi criado na API que pode ser integrado com as aplicações que estão no *FrontEnd;* essa integração funcionará a partir de rotas de acesso à API. Essas rotas de acesso são configuradas dentro da API, mas são chamadas a partir da aplicação do *FrontEnd*.

No desenvolvimento da aplicação *FrontEnd* foram criados arquivos que possuem o sufixo *Service,* que são os arquivos responsáveis pela comunicação da aplicação com a API. Na Figura 5 está apresentado um método criado que irá chamar a rota citada na Figura 4.

Figura 5 Método utilizado para chamar rota de remoção de usuário

```
// Rota para remover um usuário
removerUsuario(id) {
    return ApiService.delete(`usuarios/${id}`)
},
```

Fonte: Autoria Própria

Durante o desenvolvimento da aplicação BackEnd foi desenvolvido um método para envio de e-mail, este envio de e-mail será feito quando um método que utiliza uma anotação para ser executado a cada um segundo analisa os atendimentos e verifica se existe algum atendimento com um horário de uma hora de antecedência. Caso existam horários com este tempo determinado para ocorrer, é enviado um e-mail ao paciente para que ele se organize para que vá ao local de atendimento fazer os procedimentos estéticos. Esse método pode ser visualizado na Figura 6.

Figura 6 Método enviar email horário agendado

```
@Scheduled(fixedDelay = 1000)
public void executar() {
    // Recebe atendimentos
    LocalDateTime horario = LocalDateTime.now();
    List<EmailDataDto> datas = this.atendimentoService.findAtendimentos(horario);
    datas.forEach(item -> {
        try {
            // Puxar procedimentos do atendimento
            List<DataProcedimentoDto> procedimentos = this.atendimentoService
                    .findProcedimentoAtendimento(item.getIdatendimento());
            // Envio de email
            Util.enviaEmailHorarioAgendado(item, procedimentos);
        } catch (Exception e) {
            System.out.println(e.getMessage());
        }
    });
}
```

Fonte: Autoria Própria

O projeto de *FrontEnd* foi desenvolvido utilizando o framework React Native, e com a utilização deste framework é possível criar componentes para serem utilizados nas telas, e isso é uma das funcionalidades que o tornam uma tecnologia mais rentável.

Na Figura 7 pode ser visualizado um trecho de código desenvolvido na aplicação, que faz a criação de um campo *input* com os estilos padronizados e que é possível enviar propriedades para serem utilizados dentro do *input*. Este componente possui uma condição que é passada por propriedade para verificar se é um campo que irá utilizar máscara ou não. É um componente com funcionalidades dinâmicas para ser de fácil personalização.

Figura 7 Codificação Componente Input

```
import React, { Component } from 'react'
import { StyleSheet, View, Text, TextInput } from 'react-native';
import { TextInputMask } from 'react-native-masked-text'

class Input extends Component {
    render() {
        return (
            <View style={Styles.containerInput} >
                <Text style={Styles.textInputStyle} >{this.props.nome}</Text>
                {
                    !(this.props.mask) &&
                    <TextInput style={Styles.inputStyle} {...this.props}/>
                }
                {
                    (this.props.mask) &&
                    <TextInputMask style={Styles.inputStyle} {...this.props}/>
                }
            </View>
        )
    }
}
```

Fonte: Autoria Própria

5.2 Módulo Gerencial

Esse módulo é utilizado pela empresa de desenvolvimento com o objetivo de cadastrar as clínicas estéticas. Foi desenvolvido considerando os atributos de usabilidade, com componentes simples e intuitivos. As principais funcionalidades desse módulo estão apresentadas e comentadas a seguir. Um manual de utilização completo se encontra em (LEITE, 2020).

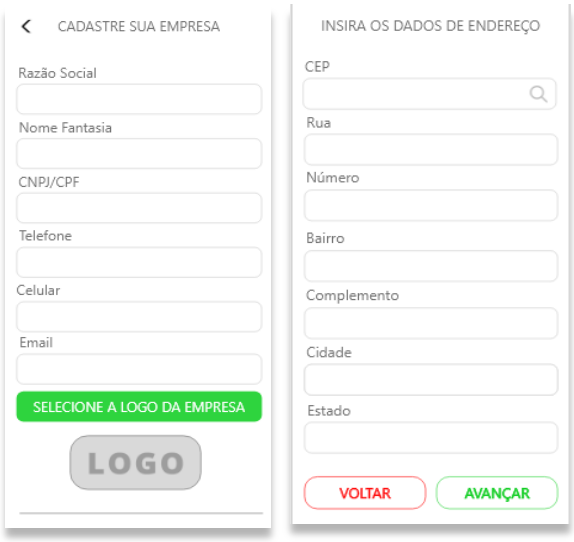
Na tela inicial, ilustrada na Figura 8, o usuário pode optar em cadastrar uma nova empresa no sistema ou verificar as empresas que já estão cadastradas. Pode ser observado que o *layout* é bem simples e fácil de ser utilizado, o que se estende para todas as telas do sistema, conforme requisito de qualidade [RQ 03] especificado na Tabela 2. As cores utilizadas no projeto foram cores mais neutras para que não se torne um sistema tão cansativo de ser utilizado, sem cores vibrantes, mantendo tons claros e com apenas alguns detalhes em destaques.

Figura 8 Tela inicial do Módulo Gerencial

Fonte: Autoria Própria

Ao clicar em CADASTRAR EMPRESAS, o usuário é levado para a área apresentada na Figura 9, na qual serão inseridos os dados cadastrais do cliente. Ao clicar no botão AVANÇAR, o sistema valida os dados: para o CPF e o CNPJ é feita uma validação se o número informado é válido e se foi inserida a quantidade correta de números (CPF com 11 caracteres e CNPJ com 14 caracteres); para o e-mail é verificado se foi informado de maneira correta; para o telefone é verificado se foi inserida a quantidade de 10 ou 11 caracteres; e para o endereço completo é verificado se todos os dados foram informados.

Figura 9 Telas para cadastramento da empresa



Fonte: Autoria Própria

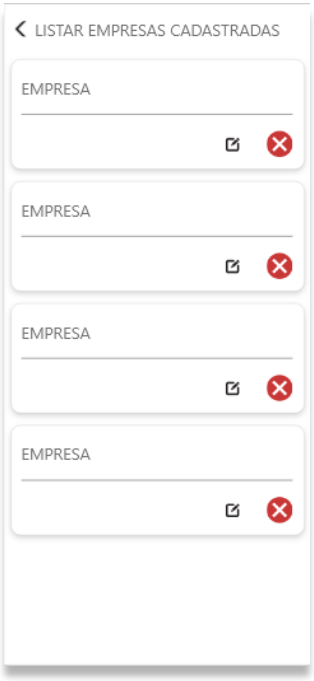
Caso tudo esteja validado, o usuário é redirecionando para a próxima etapa do cadastro para fazer o cadastro do usuário administrador, cujas telas estão apresentadas na Figura 10. Para esse cadastro devem ser informados os dados pessoais da pessoa que ficará como administrador do sistema. Ao clicar no botão CADASTRAR, o sistema cria a empresa com um usuário administrador, que será o que terá acesso a todas as funcionalidades do módulo MarqFacil.

Figura 10 Telas para cadastro do usuário principal

Fonte: Autoria Própria

Se na tela inicial (Figura 8) o usuário clicar em LISTAR EMPRESAS CADASTRADAS, ele é direcionado para uma tela com todas as empresas cadastradas (Figura 11) e pode optar em visualizar o usuário cadastrado na empresa selecionada (bastando clicar no nome da empresa), editar (clicando no ícone do lápis) ou excluir (clicando no botão vermelho) uma empresa. Se o usuário opta por excluir a empresa, o sistema pergunta se realmente o usuário deseja excluir a empresa e, caso clique em SIM, a empresa é excluída.

Figura 11 Tela listando as empresas cadastradas



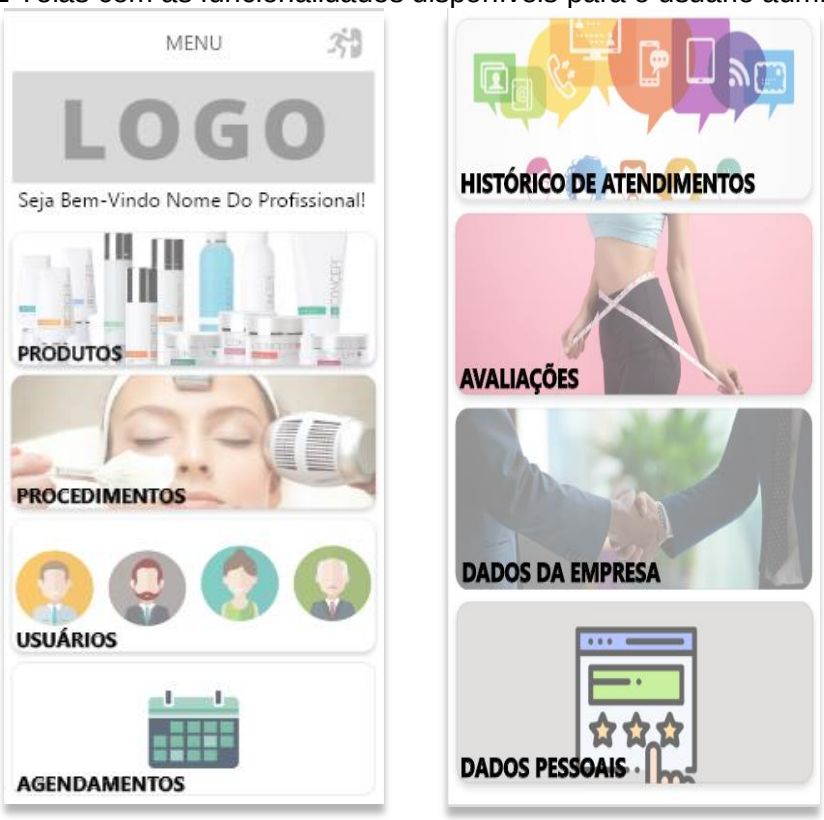
Fonte: Autoria Própria

5.3 Módulo MarqFacil

O módulo MarqFacil é o que será utilizado pelas clínicas estéticas, contendo as funcionalidades de agendamento, cadastro de avaliações, cadastro dos produtos e procedimentos, cadastro de usuários (Paciente e Colaborador), histórico de atendimentos, dados da empresa e dados pessoais.

Este módulo possui diferentes níveis de permissão para usuários: o usuário administrador tem permissão para utilizar todas as funcionalidades: gerenciamento de produtos, procedimentos, usuários, agendamentos, avaliações, dados da empresa, dados pessoais, além de visualizar o histórico de atendimentos (Figura 12). O usuário paciente tem permissão para visualizar produtos e procedimentos da clínica a qual está vinculado, seus agendamentos, avaliações, dados pessoais e seu histórico de agendamentos (Figura 13). O usuário colaborador tem permissão para visualizar os produtos e procedimentos oferecidos pela empresa, verificar os agendamentos que serão feitos por ele com a função de poder trocar o status do atendimento, histórico dos atendimentos e os dados pessoais (Figura 14).

Figura 12 Telas com as funcionalidades disponíveis para o usuário administrador



Fonte: Autoria Própria

Figura 13 Telas com as funcionalidades disponíveis para o usuário paciente

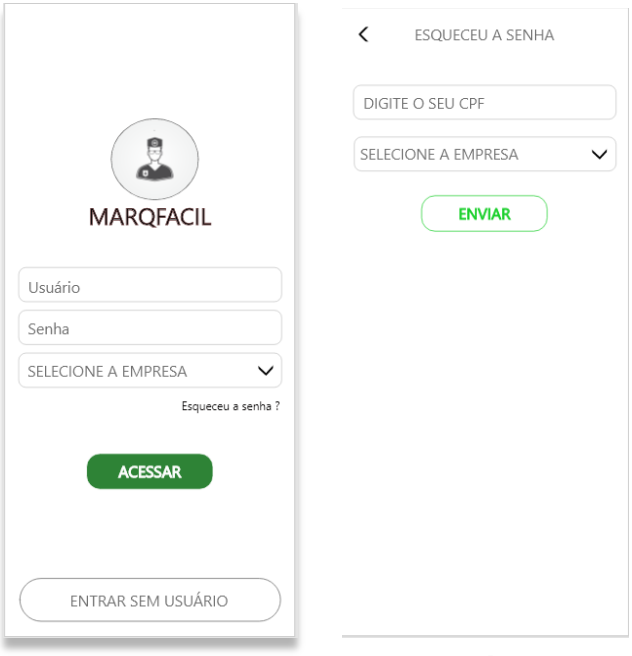
Fonte: Autoria Própria

Figura 14 Telas com as funcionalidades disponíveis para o usuário colaborador

Fonte: Autoria Própria

A tela inicial do sistema é a tela de Login (Figura 15a), que é composta pelos campos usuário, senha e o campo para selecionar a clínica estética a que pertence o usuário. Nessa tela também existe a opção ESQUECEU A SENHA, que direciona o usuário para a tela de recuperação de senha (Figura 15b), na qual o usuário deve inserir o CPF e uma opção para selecionar a empresa que o usuário é cadastrado. Caso o usuário esteja mesmo cadastrado, o sistema envia um e-mail com a nova senha.

Figura 15 (a) Tela inicial do Módulo MarqFacil e (b) Tela de recuperação de senha

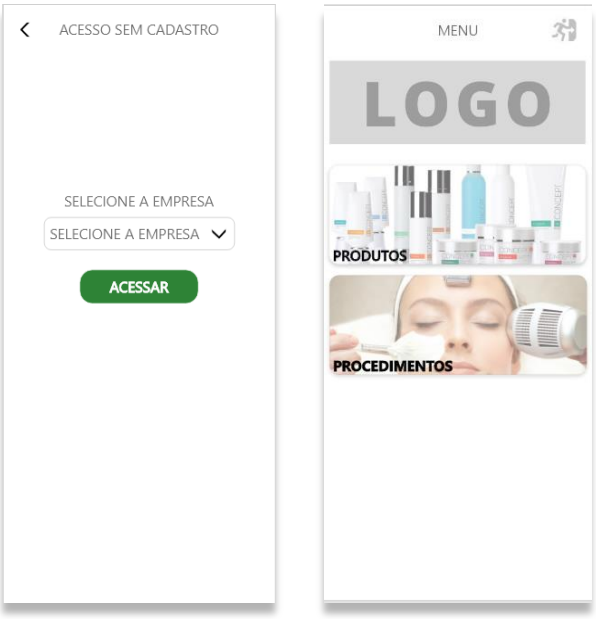


(a) (b)

Fonte: Autoria Própria

As informações das clínicas cadastradas podem ser vistas por qualquer pessoa, seja cadastrada ou não. Se a pessoa não tiver cadastro, na tela inicial (Figura 15a), basta clicar na opção ENTRAR SEM USUÁRIO, e será direcionada para uma tela na qual existe a opção para selecionar a clínica estética que deseja visualizar (Figura 16a); após essa seleção, os produtos e procedimentos dessa empresa são apresentados (Figura 16b).

Figura 16 (a) Tela de escolha da empresa que se deseja visualizar as informações e (b) Tela de apresentação de produtos e procedimentos



(a) (b)

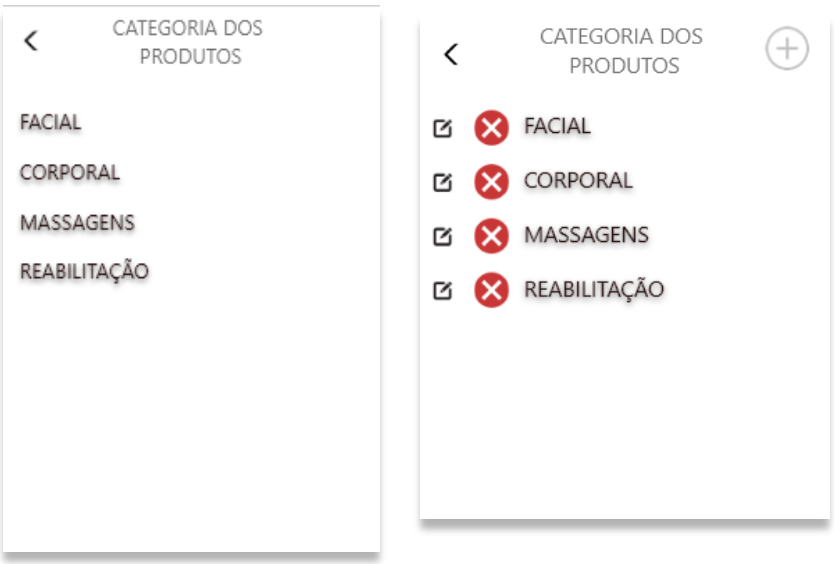
Fonte: Autoria Própria

Os usuários cadastrados, após fazerem o login na tela inicial, serão direcionados à tela de Menu, que é customizada para cada tipo de usuário: Administrador (Figura 12), Paciente (Figura 14) e Colaborador (Figura 15).

Todos os tipos de usuários cadastrados podem visualizar as categorias dos produtos (Figura 17a), a partir da opção PRODUTO da tela de Menu. Caso o usuário seja o administrador, aparecerá também as opções de adicionar uma nova categoria (clicando no ícone de +, no canto superior direito), editar (clicando no ícone do lápis, na frente de cada produto) ou excluir (clicando no ícone de X, na frente de cada produto), como pode ser visto na Figura 17b.

Ao selecionar uma categoria na lista, o sistema irá para a tela com a lista de produtos daquela categoria (Figura 18a). No caso do usuário ser o administrador, o sistema exibe se o produto está ativado ou não e permite editar ou excluir o produto, além de permitir também a inclusão de um novo produto (Figura 18b). Os ícones para inclusão, exclusão e alteração são padronizados, ou seja, são os mesmos já comentados para a categoria. Quando selecionado algum produto, o sistema direciona o usuário para a tela de detalhes do produto, que exibe a imagem do produto (caso possua), o nome do produto, a descrição e o valor (Figura 18c).

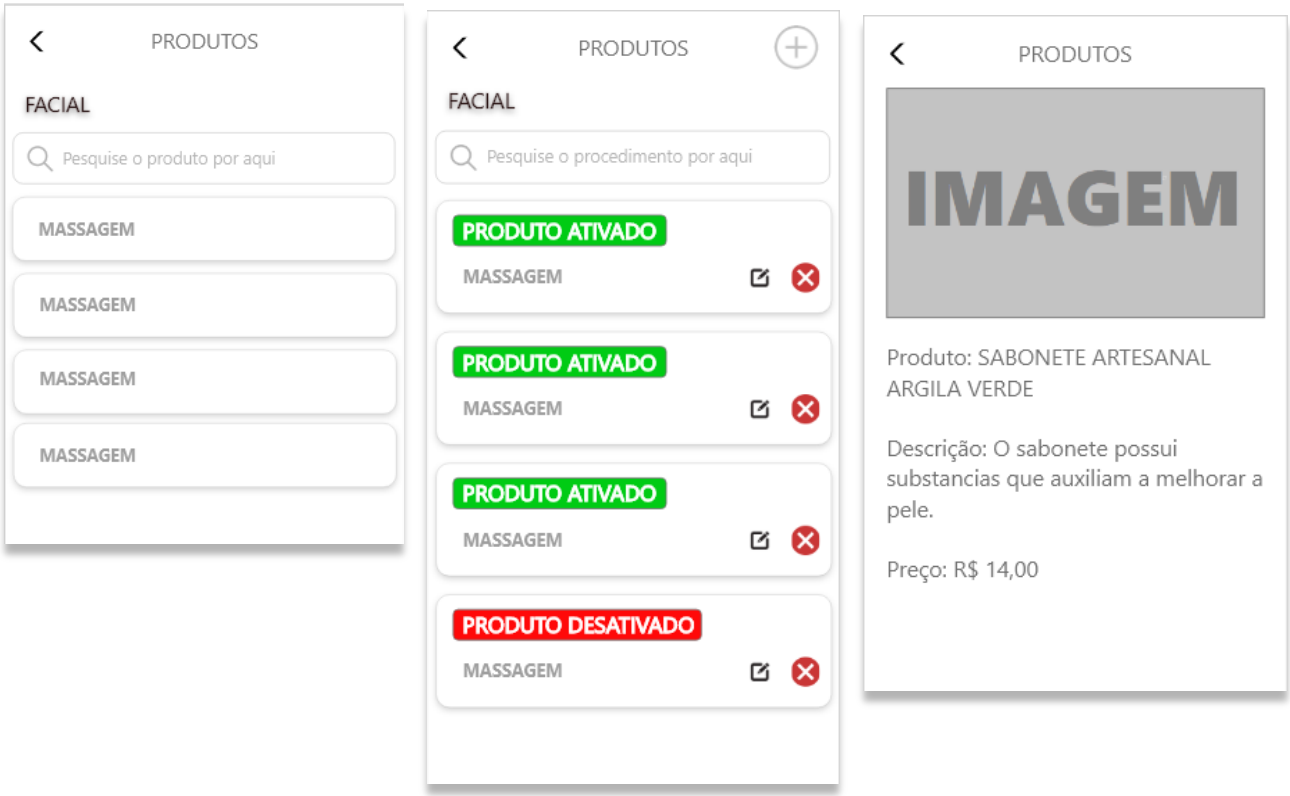
Figura 17 Telas de categorias de produtos (a) paciente e colaborador; (b) administrador



(a) (b)

Fonte: Autoria Própria

Figura 18 Telas de visualização de produtos (a) paciente e colaborador; (b) administrador; (c) Tela de detalhes dos produtos



(a) (b) (c)

Fonte: Autoria Própria

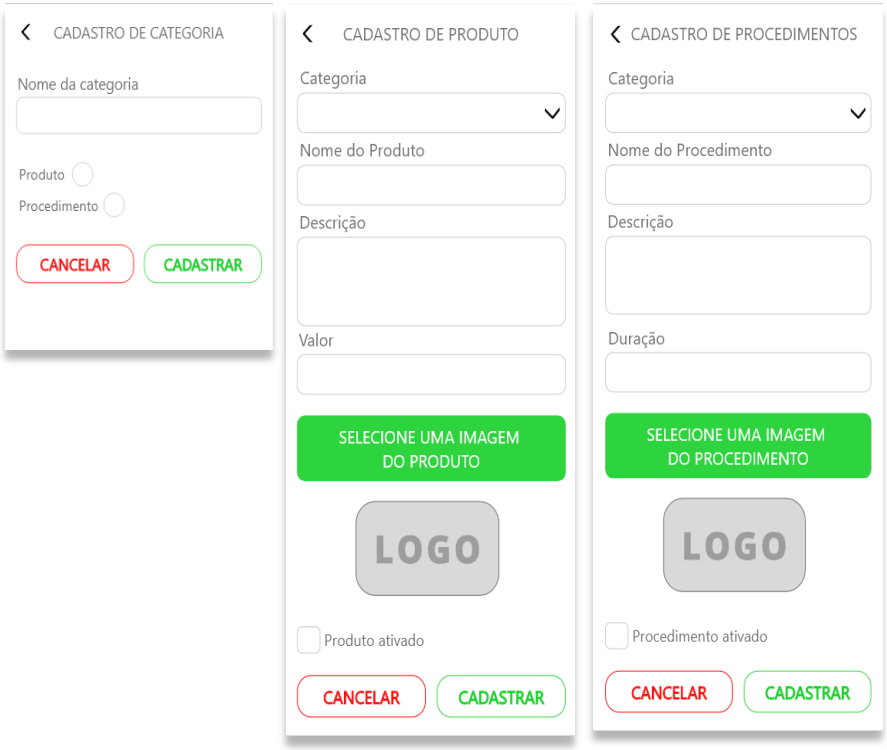
Buscando a facilidade de utilização do módulo, o processo para o

gerenciamento de Procedimentos é o mesmo do processo de gerenciamento de Produto, que já foi explicado. As telas são padronizadas, mudando apenas as informações de Produtos para Procedimentos e podem ser visualizadas no GITHUB (LEITE, 2020).

Como já dito, a inclusão de Categoria é de responsabilidade apenas do usuário administrador e ocorre pressionando o ícone de mais (+) por dois meios: pela tela de Lista de Categorias dos Produtos e, seguindo a padronização, pela tela de Lista de Categorias dos Procedimentos. Para a inclusão de uma Categoria é necessário digitar seu nome e especificar se é de Produto ou Procedimento, conforme pode ser observado na Figura 19a.

Ainda buscando a padronização de processos, assim como explicado para a inclusão de Categoria, a inclusão de Produtos e de Procedimentos ocorre pressionando o ícone de mais (+) na tela de Lista de Produtos ou na Lista de Procedimentos, respectivamente. Conforme pode ser observado na Figura 19b, para a inclusão de um novo produto é necessário preencher os seguintes campos: categoria, nome do produto, descrição, valor, imagem e status. De forma similar, para a inclusão de um novo procedimento é necessário preencher os campos: categoria, nome do procedimento, descrição, duração, imagem e status, conforme pode ser visto na Figura 19c.

Figura 19 Telas de inclusões de categoria (a); produtos (b); e procedimentos (c)



(a) (b) (c)

Fonte: Autoria Própria

Como dito anteriormente, além de gerenciar categorias, produtos e procedimentos, o usuário administrador também pode gerenciar os usuários do sistema, bastando clicar na opção USUÁRIOS, na tela de menu, sendo direcionado à tela na qual são exibidos os nomes dos usuários (Figura 20a). Para inserir um novo

usuário, basta clicar no ícone (+), e preencher os campos: Nome Completo, RG, CPF, Data de Nascimento, Telefone, Celular, E-mail, Tipo Usuário, Endereço completo, Usuário e Senha. O cadastro de usuário possui formatação nos campos para que seja mais fácil inserir os dados, com um fácil entendimento do que deve ser informado em cada um (Figura 20b).

Como pode ser observado na Figura 20a, o administrador também pode buscar usuários pelo nome, editar ou excluir um usuário e, quando pressionado o *card* de um usuário específico, o sistema direciona o administrador para uma tela com os dados do usuário (Figura 20c).

Figura 20 (a) Telas de usuários; (b) Tela cadastro de usuários; (c) Tela de detalhes de usuário

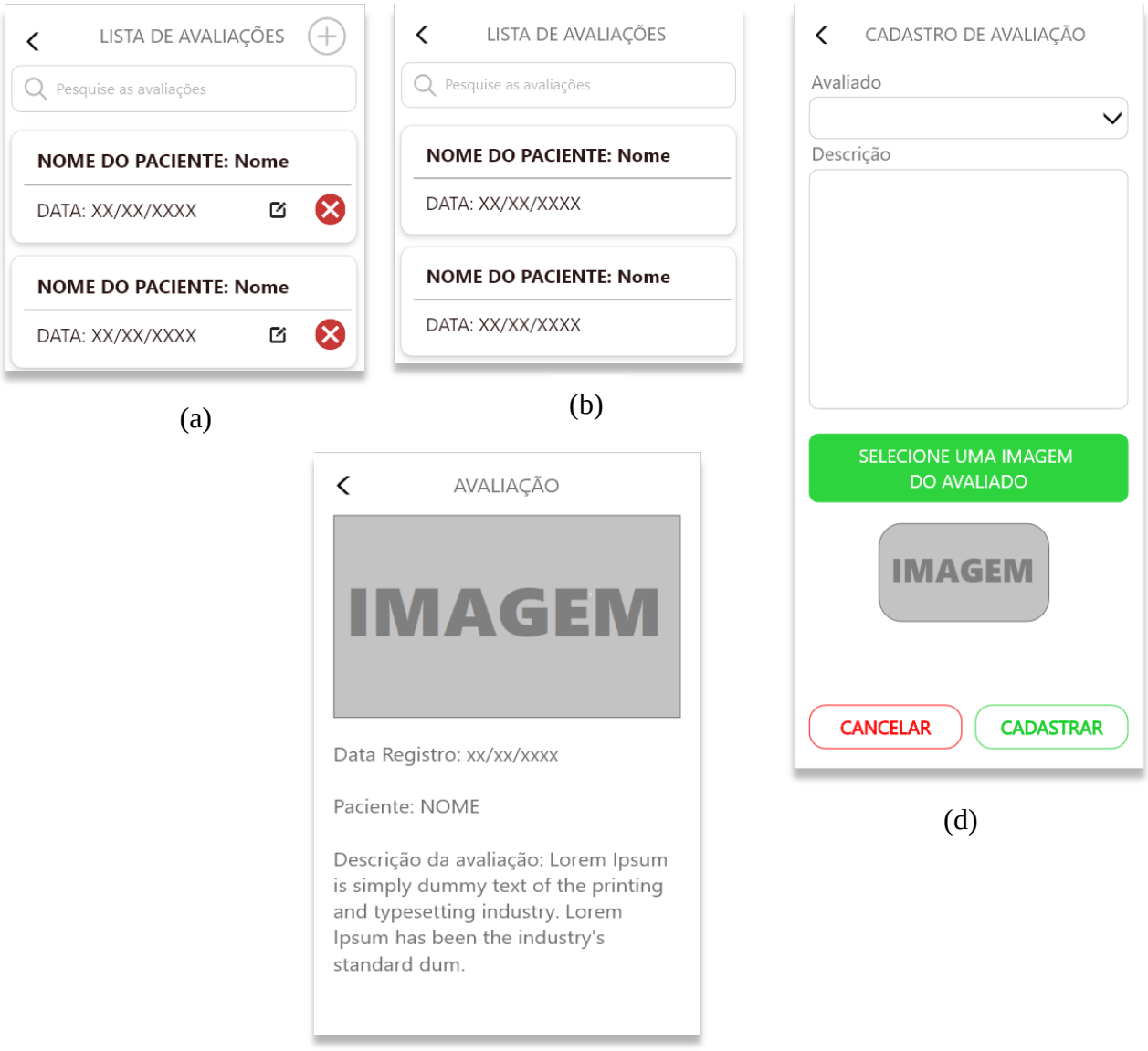
(a) (b) (c)

Fonte: Autoria Própria

O sistema permite o gerenciamento de Avaliações de pacientes, realizado pelo usuário administrador. A tela de lista de avaliações (Figura 21a), pode ser acessada no menu, porém apenas os usuários paciente e administrador terão acesso a ela, pois o usuário paciente pode visualizar suas avaliações e, quando pressionado o *card* de uma avaliação específica, o sistema mostra os seus detalhes: a imagem, a data do registro, o nome do paciente e a descrição da avaliação (Figura 21c).

O usuário administrador contará com a lista de todas as avaliações (Figura 21b) cadastradas no sistema e com as opções de adicionar uma nova avaliação, editar ou excluir e com as opções já citadas no usuário paciente. Como em todos os gerenciamentos de cadastros do sistema, a tela de cadastro é acessada a partir do ícone de mais (+) existente na tela de lista de avaliações. Para cadastrar uma avaliação é necessário preencher os campos: paciente avaliado, uma descrição e uma imagem referente a avaliação, os campos são bem informativos com o que deve ser informado em cada um (Figura 21d).

Figura 21 Telas de Avaliações (a) administrador; (b) paciente; (c) detalhes da avaliação; (d) cadastro de avaliação



Fonte: Autoria Própria

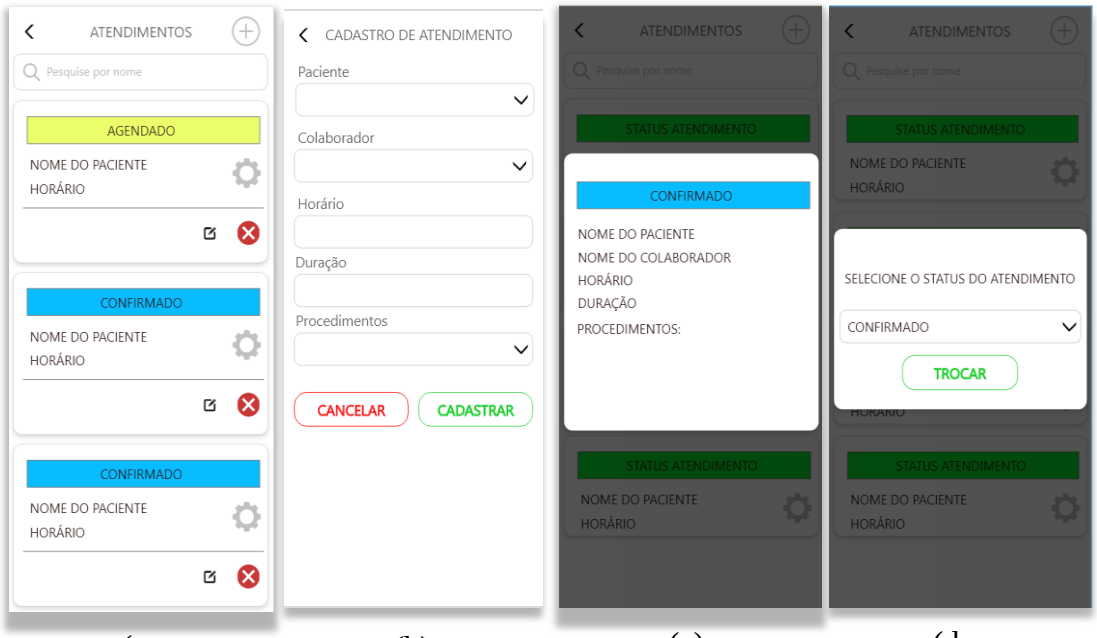
Uma das funcionalidades mais importantes do sistema é o Agendamento de Atendimentos, que pode ser realizado apenas pelo usuário administrador. Para realizar o agendamento, basta clicar na opção AGENDAMENTOS, na tela de menu que o sistema direciona para a Lista de Atendimentos, que exibe todos os atendimentos com status de AGENDADO ou CONFIRMADO (Figura 22a).

Nessa tela existe o ícone de mais (+) que, quando pressionado, direciona o usuário para a tela de Cadastro de Atendimento (Figura 22b), na qual o usuário preenche os campos: Paciente, Colaborador, Horário, Duração e Procedimentos. No campo horário deve ser informada a data e hora de atendimento; a duração é um campo que pode ser informado ou não, pois, caso não seja informado, o sistema irá calcular a duração automaticamente.

Se o usuário administrador, paciente ou colaborador apenas clicar em algum dos *cards* de atendimento da lista de atendimentos, o sistema abre uma tela

com os detalhes do atendimento, exibindo: nome do paciente, nome do colaborador, horário, duração, os procedimentos que serão feitos e o status do atendimento (Figura 22c). Se o usuário administrador ou colaborador na lista de atendimentos, clicar no ícone de uma engrenagem, aparecerá uma tela com a opção de troca do status do atendimento, podendo selecionar as opções de atendimento e, ao pressionar o botão, altera o status do atendimento (Figura 22d).

Figura 22 Telas de Agendamento (a) Tela lista de atendimentos; (b) Tela cadastro de atendimento; (c) Tela de detalhes do atendimento; (d) Tela de troca de status

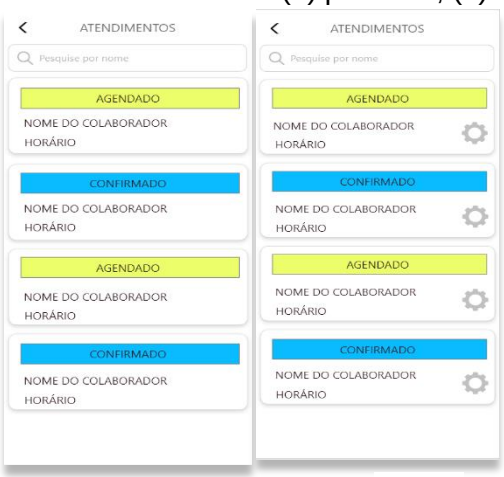


(a) (b) (c) (d

Fonte: Autoria Própria

O paciente pode, apenas, visualizar a lista dos seus atendimentos e ver os detalhes dos atendimentos, clicando sobre o *card* que ele marcou (Figura 23a). O Colaborador pode trocar o status do atendimento, além de ver detalhes dos atendimentos e visualizar a lista de todos os atendimentos que irá fazer (Figura 23b).

Figura 23 Tela de Atendimentos (a) paciente; (b) colaborador



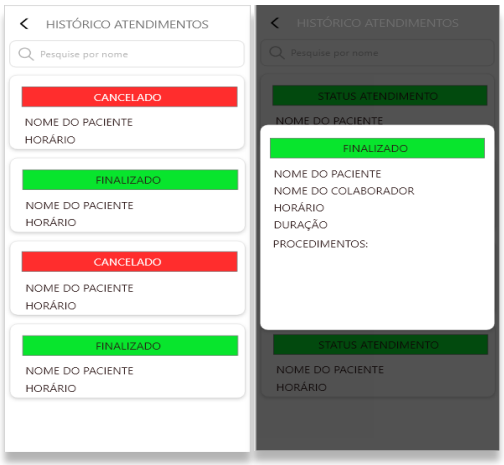
(a) (b)

Fonte: Autoria Própria

O usuário administrador irá conseguir visualizar todos os atendimentos marcados com as opções de adicionar novos atendimentos, trocar status e com as operações de editar ou excluir algum atendimento. Todos usuários possuem a opção de pesquisar algum atendimento pelo nome do paciente ou colaborador.

No menu existe a opção HISTÓRICO DE ATENDIMENTOS que, ao ser selecionada, o sistema direciona para a lista de todos os atendimentos (Figura 24a) com os status de “CANCELADO” ou “FINALIZADO”, com opção de pesquisar por nome os atendimentos e, quando pressionado sobre algum dos *cards* de atendimento, o sistema exibe os detalhes do atendimento (Figura 24b), com o status do atendimento, nome do paciente, nome do colaborador, horário, duração e os procedimentos feitos.

Figura 24 (a) Tela do Histórico de Atendimentos; (b) Tela de detalhes do Atendimento

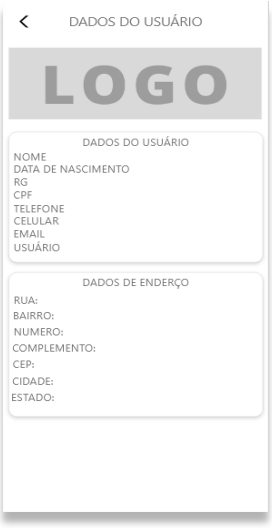


(a) (b)

Fonte: Autoria Própria

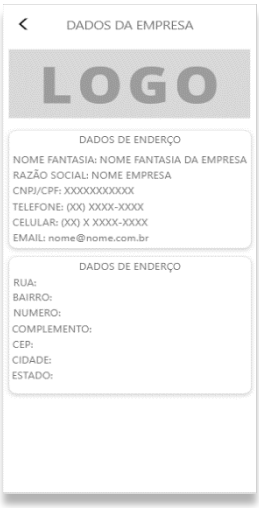
As duas últimas funcionalidades do sistema são a Visualização dos Dados Pessoais e a Visualização dos Dados da Empresa, ambas funcionalidades sendo executadas a partir do menu principal. Quando qualquer usuário cadastrado clica na opção de DADOS PESSOAIS, no menu, o sistema o direciona para a tela com os dados do usuário logado no sistema (Figura 25), sendo exibidos: a logo da empresa que o usuário está cadastrado, o nome do usuário, a data de nascimento, o rg, o cpf, o telefone, o celular, o email, o usuário e os dados de endereço. Quando o usuário administrador clica na opção DADOS DA EMPRESA, o sistema o direciona para a tela com as informações da empresa (Figura 26): Logo, Nome Fantasia, Razão Social, CNPJ/CPF, Telefone, Celular, E-mail e os dados de endereço. Importante ressaltar que o usuário só consegue visualizar os dados cadastrados, pois, qualquer mudança nesses dados só pode ser feita pelo módulo gerencial.

Figura 25 Tela de Dados Pessoais



Fonte: Autoria Própria

Figura 26 Tela de Dados da Empresa



Fonte: Autoria Própria

6 Conclusão

O projeto teve como objetivo criar um MVP para desenvolver um software para auxiliar no gerenciamento de agenda e controle de avaliações de pacientes de clínicas estéticas, e auxiliar na divulgação dos serviços e produtos oferecidos por essas clínicas.

O MVP que foi apresentado no artigo foi completamente desenvolvido, utilizando técnicas da engenharia de software e metodologia ágil para organização do projeto, de forma a desenvolver um sistema de qualidade, porém não foi implantado no cliente.

O módulo foi desenvolvido utilizando o framework React Native no *FrontEnd*, utilizou-se a linguagem de programação Java com o uso do Framework Spring Boot no *BackEnd* e o banco de dados utilizado na aplicação foi o MySQL.

Durante o desenvolvimento do software, foi necessário aprender a utilizar a plataforma Enterprise Architect para a especificação dos diagramas, mas tudo foi concluído sem muitos problemas.

No futuro, espera-se modificar o software para ser utilizado em plataforma WEB tornando mais abrangente o sistema, contando com todas as funcionalidades do módulo, com a implantação da funcionalidade do pagamento com a utilização de boletos e mais algumas novas funcionalidades voltadas para emissão de relatórios.

Referências

ANLEY, Chris. *Advanced SQL injection in SQL server applications*. 2002.

BECKER, L. O que é React Native? Organica Natural Marketing 2020. Disponível em: <https://www.organicadigital.com/blog/o-que-e-react-native/>. Acesso em 11 mar 2020.

BISSI, Wilson. Metodologia de desenvolvimento ágil. Campo Digital, v. 2, n. 1, 2007.

BOCH, G.; RUMBAUGH, J.; JACOBSON, I. UML – Guia do Usuário. Editora: Elsevier, 2006.

BORBA, Tamila J; THIVES, Fabiana M. Uma reflexão sobre a influência da estética na auto estima, auto-motivação e bem estar do ser humano. 2011.

BRANDEN, Nathaniel. Como aumentar sua auto estima: Aprenda a Acreditar em Si Mesmo e a Viver com Confiança e Otimismo. Rio de Janeiro. Sentante, 2009.

CUNHA, Francisco José Aragão Pedroza. A gestão da informação nos hospitais: importância do prontuário eletrônico na integração de sistemas de informação em saúde, 2005. Dissertação (mestrado). Universidade Federal da Bahia, Salvador. Disponível em: < https://repositorio.ufba.br/ri/bitstream/ri/8174/1/Disserta%c3%a7%c3%a3o_Francisco%20Pedroza.pdf>. Acesso em 04 jul 2020.

DAL MORO, Tharcis; DORNELES, Carina; REBONATTO, Marcelo Trindade. Web services WS-* versus Web Services REST. Revista Eletrônica de Iniciação Científica em Computação, v. 11, n. 1, 2011.

ELMASRI, Ramez; NAVATHE, Shamkant B. Sistemas de Banco de Dados. 7 ed. São Paulo: Pearson Universidades, 2019.

FEITOSA, Ailton Luiz Gonçalves. A integração entre sistemas legislativos, terminologia e web semântica na organização e representação da informação legislativa. 2005. Tese (Doutorado em Ciência da Informação). Universidade de Brasília, Brasília, 2005. Disponível em: < https://repositorio.unb.br/handle/10482/34606>. Acesso em 05 jul 2020.

FRANÇA, Ilka Cavalcante et al. Eficácia da técnica de massagem modeladora para redução de adiposidades e do fibro edema gelóide. V. 4, n. 2, p. 23-30, 2016.

INVENTTI. Metodologia Ágil: Scrum X Kanban. Inventti Tecnologia. Disponível em: <https://www.inventti.com.br/scrum-x-kanban/>. Acesso em 20 mar 2020.

ISAC JUNIOR. JSX. Brasil JS. Disponível em: <https://braziljs.org/artigos/jsx/>. Acesso em 10 mar 2020.

JOHNSON, Rod et al. *The spring framework-reference documentation*. Interface, v. 21, p.27, 2004.

LEITE, José Hamilton Martins. Documentação do Projeto TCC. Disponível em: <https://github.com/joses166/TCC-2020-UniFACEF>. Acesso em 25 ago. 2020.

MARIOTTI, Flavio S. Engenharia de Software Magazine - Kanban no desenvolvimento de projetos de software. 2012. 45 ed. 7 p.

MENDES, Juliana Veiga; ESCRIVÃO FILHO, Edmundo. Sistemas integrados de gestão ERP em pequenas empresas: um confronto entre o referencial teórico e a prática empresarial. Gestão & Produção, v. 9, n. 3, p. 277-296, 2002. Disponível em:< https://www.scielo.br/pdf/gp/v9n3/14570.pdf>. Acesso em 25 jul 2020.

MEYER, M. A história do Android. Oficina da Net. Disponível em: <https://www.oficinadanet.com.br/post/13939-a-historia-do-android>. Acesso em 15 mar 2020.

MILANI, André. MySQL-guia do programador. Novatec Editora, 2007.

OSTERWALDER, Alexander; PIGNEUR, Yves. Business model canvas. Self published. Last, 2010.

RIBEIRO FILHO, Carlos Caldas. Aproximações entre teologia e estética. Uma introdução em perspectiva da teologia protestante. 2016. V. 11 N. 19, p. 128-143

RICCIO, Edson Luiz. Efeitos da tecnologia de informação na contabilidade: estudo de casos de implementação de sistemas empresariais integrados-ERP, 2001. Tese (Livre-Docência em Contabilidade e Atuária). Universidade de São Paulo, São Paulo: FEA/USP. Disponível em :< https://teses.usp.br/teses/disponiveis/livredocencia/12/tde-06122005-101802/publico//riccio.pdf>. Acesso em 04 jul 2020.

SAUDATE, Alexandre. REST: Construa API's inteligentes de maneira simples. Editora Casa do Código, 2014.

SILVA, T. R. B.; MERCADO, N. F. Criolipólise e sua eficácia no tratamento da gordura localizada: Revisão bibliográfica. 2015. v. 3.

SOMMERVILLE, Ian. Engenharia de Software. 9° Edição. 2011.

WALLS, Craig. *Spring Boot in action*. Manning Publications, 2016.

WARCHOLINSKI, M. 10 *Famous Apps Built with React Native*. Brainhub. Disponível em: <https://brainhub.eu/blog/react-native-apps/>. Acesso em 10 mar 2020.