

SoftHair - Aplicativo de Gestão de Salões de Beleza e Clínicas de Estética

Cesar E. Silva¹, Luiz Guilherme S. Oliveira²

¹Instituto Federal do Paraná (IFPR)

Caixa Postal 780 – 85860-000 – Foz do Iguaçu – PR – Brazil

Resumo. *A área da beleza vem crescendo a cada dia, com o aumento no número de clientes e também quantidade de produtos vendidos movimentando a economia desse setor. O Brasil atualmente é o quarto no ranking mundial em consumo de produtos de higiene pessoal, perfumaria e cosméticos (FORBES). Só por este dado é possível entender o grande potencial que o segmento tem. Mesmo diante da forte crise que o Brasil enfrentou, a média anual de crescimento desse setor nos últimos 10 anos foi 4,1%. Em 2019 uma pesquisa encomendada pela Beauty Fair (Feira Internacional de Beleza Profissional) em que foram registrados cerca de 500 mil salões de beleza em todo o Brasil. Um grande fator que tem contribuído para um número tão expressivo é o aumento significativo de franquias no setor, sejam de salões de beleza, esmalterias ou design de sobrancelhas, por exemplo. Na verdade, o número de salões de beleza no país é muito maior, já que quase 50% dos salões ainda estão na informalidade. E como estamos numa era digital, onde os processos acontecem de forma ágil, ferramentas surgem para facilitar a vida das pessoas, inclusive na área da estética e serviços relacionados na área da beleza. No App SoftHair o tema proposto é que o prestador que proverá os serviços de beleza, terá as funcionalidades de controlar a sua agenda de clientes, dos serviços prestados e gestão de agendamento e atendimentos, que por fim agregará agilidade e segurança nas rotinas do usuário ao sistema.*

1. Introdução

O setor de beleza no Brasil é um dos mais promissores. São vários aspectos que colaboram para esse fato: inclusão das classes “D” e “E” decorrente do aumento de renda, inserção da mulher no mercado de trabalho, lançamento constante de novos produtos, elevação da expectativa de vida, entre outros.

A população apresenta alta demanda por produtos e serviços de qualidade e procedimentos específicos que contribuam para a elevação da autoestima e do bem estar e atendam às necessidades de higiene pessoal.

Mesmo diante de cenários econômicos de crise nos anos recentes, o Brasil está em 4º lugar no ranking mundial com relação ao mercado de beleza e cuidados pessoais, de acordo com a revista Forbes. Segundo o IBGE, o brasileiro gasta mais com beleza do que

com comida. Então, este é um dos melhores ramos para se investir atualmente. (SALOMÃO, 2020).

Este trabalho tem como objetivo o desenvolvimento de um aplicativo *mobile* para gestão de salões de beleza e clínicas de estética, destinado a controlar e organizar os serviços de rotina em tal ambiente. Este público foi escolhido visto o crescente número de microempreendedores individuais e também de micro empresas.

Os benefícios de informatizar as atividades relacionadas a esse nicho de mercado é fundamental na gestão desse tipo de empreendimento. As facilidades são altamente positivas, automatizando os processos, se reúnem as informações e rotinas do estabelecimento, como fluxo de agendamentos aos clientes, cadastro de funcionários, controle e gestão de serviços, levantamento de vendas e cálculo de comissão.

Este aplicativo *mobile* em fase de desenvolvimento é uma continuação de um projeto em *web* anterior, que foi desenvolvido com a proposta de resolver as dificuldades em controlar as rotinas de um salão de beleza. As duas modalidades combinadas dos sistemas implantados, agregarão segurança e praticidade na gestão dos serviços prestados.

2. Escopo

O salão de beleza utilizado como modelo no desenvolvimento do sistema *web* teve como objetivo liquidar com as antigas agendas de papel, nas quais haviam muitas rasuras e letras mal escritas. Este projeto de *software* de um aplicativo *mobile* tem como interesse integrar ao sistema *web*, funcionalidades e rotinas específicas deste mercado da beleza, a fim de resolver problemas relacionados à falta de um sistema informatizado, com o intuito de trazer robustez e segurança nos serviços prestados.

3. Metas

Para entregar todas as funcionalidades do aplicativo, seguiu-se o planejamento do projeto, atentando-se ao cronograma. Dividindo-se as tarefas entre os membros da equipe e efetuando-se o versionamento de código para garantir agilidade e consistência na construção do *App*. Abaixo segue a lista das principais funções do *software*.

- Manter usuários para acesso ao Sistema;
- Manter atividades;
- Controle de agendamento do Cliente;

4. Requisitos e Regras de Negócio

4.1. Requisitos Funcionais

Tabela 1. Requisitos Funcionais

Código	Requisito	Descrição do requisito funcional
RF01	Cadastrar prestador	O cadastro dos prestadores no sistema deverá ser efetuado com nome, <i>e-mail</i> e senha. Esse cadastro será realizado pelo administrador.
RF02	Cadastrar serviços	Cadastrar os tipos de serviços realizados pelo aplicativo. O sistema permitirá a manutenção de inclusão, alteração, consulta e exclusão de dados, essas funcionalidades estarão a cargo do administrador.
RF03	Cadastrar clientes	Cadastrar o cliente no aplicativo. A manutenção dos dados pelo sistema permitirá a inclusão, exclusão, consulta e alteração, a cargo do administrador e do prestador de serviço.
RF04	Cadastrar agendamento	Administrador e prestador podem registrar um agendamento.
RF05	Cadastrar atendimento	O atendimento é um agendamento com alguns campos adicionais como valor dos procedimentos, valor dos produtos, descrição detalhada dos procedimentos para consulta futura.
RF06	Validação de dados do cadastro	O cadastro de usuários deverá ser validado pelo banco de dados, os campos nome e email serão únicos, evitando duplicidade.

4.2. Requisitos Não Funcionais

Tabela 2. Requisitos Não-Funcionais

Código	Descrição do requisito não-funcional
RNF01	O <i>framework React Native</i> será utilizado no desenvolvimento do aplicativo.
RNF02	O sistema será desenvolvido em linguagem <i>JavaScript</i>
RNF03	O sistema utilizará o banco de dados <i>PostgreSQL</i> .
RNF04	O <i>Knex.js</i> será utilizado na manipulação de <i>Queries</i> no banco de dados.
RNF05	O <i>framework Express.js</i> será utilizado no desenvolvimento das requisições HTTP na API <i>Rest</i> .
RNF06	O servidor <i>web</i> utilizado será o <i>Node.js</i> .
RNF07	O aplicativo será implementado na plataforma <i>Android</i> na versão 10.
RNF08	O servidor <i>web</i> fará uso do utilitário de interface <i>nodemon.js</i> .
RNF09	O ferramenta <i>Postman</i> utilizada para realizar os testes das requisições HTTP
RNF10	Validação de <i>token JWT (JSON WEB TOKEN)</i> será implementado no <i>backend</i> com o <i>Node.js</i> , para autenticação de requisições entre as duas partes servidor <i>web</i> e o aplicativo, a cada solicitação irá incluir o <i>JWT</i> permitindo que prestador continue acessando o sistema com o <i>token</i> válido, caso o <i>token</i> tenha expirado retorna a tela de <i>login</i> .

4.3. Regras de Negócio

Tabela 3. Regras de Negócio

Código	Descrição regra de negócio
RN01	Para o aplicativo ter acesso ao servidor <i>web</i> e para a manipulação no banco de dados , terá como protocolo de segurança a implementação de geração de <i>token</i> , a fim de barrar esta eventualidade.
RN02	A alteração de senha de acesso ao aplicativo somente será realizada pelo prestador
RN03	Somente será permitido iniciar um serviço com agendamento criado
RN04	O agendamento se tornará um atendimento mediante a alteração do <i>status</i> .
RN05	O valor dos serviços prestados serão vinculados ao atendimento caso ele se concretize.
RN06	O agendamento que não se torna um atendimento tem seu <i>status</i> alterado para inativo.
RN07	O prestador de serviço terá um único cliente por agendamento.
RN08	O prestador de serviço terá um atendimento único por horário.
RN09	O agendamento terá uma verificação prévia na data e hora antes da inserção, para evitar duplicidade.
RN10	Os campos de cadastro tal como o nome deverá ser obrigatoriamente preenchido.
RN11	O campo e-mail deverá ser único.

5. Metodologia

Inicialmente, o desenvolvimento do aplicativo *App SoftHair* foi idealizado com foco em um projeto advindo de uma versão *web*, pois envolve o atendimento de demandas as quais o segmento necessitam, com o intuito de contemplar agilidade e segurança nas rotinas e atividades de serviços relacionados a salões de beleza a implantação de um *software* na área *mobile*.

Assim, a metodologia do projeto envolveu em basicamente três etapas:

(1) Desenvolvimento e testes da API *Rest*, que faz a comunicação por meio de requisições HTTP das operações lógicas do aplicativo, onde o aplicativo que é parte do *front end(interface)*, a qual o usuário interage com este servidor *backend*; (2) levantamento das rotinas e atividades a qual o aplicativo envolveriam, para suprir as dificuldades no segmento da beleza que poderão agregar benefícios para os usuários deste *software*. (3) Implementação do banco de dados e as suas relações entre as entidades levantadas no escopo inicial do projeto.

As tecnologias computacionais adotadas foram: linguagem de programação *Javascript* com uso do ambiente de execução em *Node.js* para implementação da API *Rest*, no lado *backend* da aplicação, agregando a API foi usado o método *JWT(JSON Web Token)* que permite a autenticação entre as requisições entre as duas aplicações, para os testes das chamadas às HTTP e agilidade na integração destes foi utilizada a ferramenta *Postman*. Para a implementação do aplicativo no lado *frontend* a linguagem de programação *JavaScript* com o *framework React Native* na plataforma *Android*, o *MySQL* utilizado para armazenamento dos dados, servidor local *Apache2*, e o ambiente de desenvolvimento o *VSCode*. As telas do aplicativo foram implementados no *software* de prototipagem *Builder X*. Para controle de versionamento de código na plataforma *GitHub*.

6. Tecnologias utilizadas na API *Rest*.

6.1 *Javascript*

A linguagem de programação *JavaScript* possibilita a implementação de recursos complexos em páginas web. Um *site* que não está mostrando simplesmente informações estáticas, e sim apresentando conteúdo que se atualiza em um intervalo de tempo, mapas interativos ou gráficos 3D animados, entre outros itens muito enriquecedores que tornam um *website* interessante de ser navegado. Nesta página web o *JavaScript* certamente está presente.

O *JavaScript* foi criado como um complemento para o navegador da *Netscape*. Por certo período foi rotulado como uma linguagem ruim, bagunçada e lenta. Com uma evolução significativa ela se tornou organizada e rápida. Segundo a *RedMonk*, uma empresa de

Uma aplicação nativa ao invés de uma aplicação com um navegador, oferece melhor performance do que uma aplicação híbrida. As aplicações híbridas dependem do navegador padrão do sistema, com a possibilidade de criar um código que o navegador daquele dispositivo não suporte. Em uma aplicação nativa, por ser independente de navegadores, não há preocupação se haverá suporte para as funcionalidades.

6.2. Node.js

O *Node.js* se caracteriza como um ambiente de execução *JavaScript*. Com ele, é possível criar aplicações sem depender do *browser* para isso. Com alta capacidade de escalabilidade, boa flexibilidade, e produtividade. O *Node.js* é uma plataforma muito versátil e que pode ser usada em inúmeros cenários. Seu gerenciador de pacotes o *npm* (*Node Package Manager*) é classificado como o maior repositório de *softwares* disponível, sendo que um desses pacotes mais famoso é um *framework* voltado a desenvolvimento web, chamado *Express.js* entre outros *packages* que auxiliam no desenvolvimento como o *nodemon* e *Knex*, destes citados foram usados no desenvolvimento do projeto.

6.3. Estrutura da API *Rest*

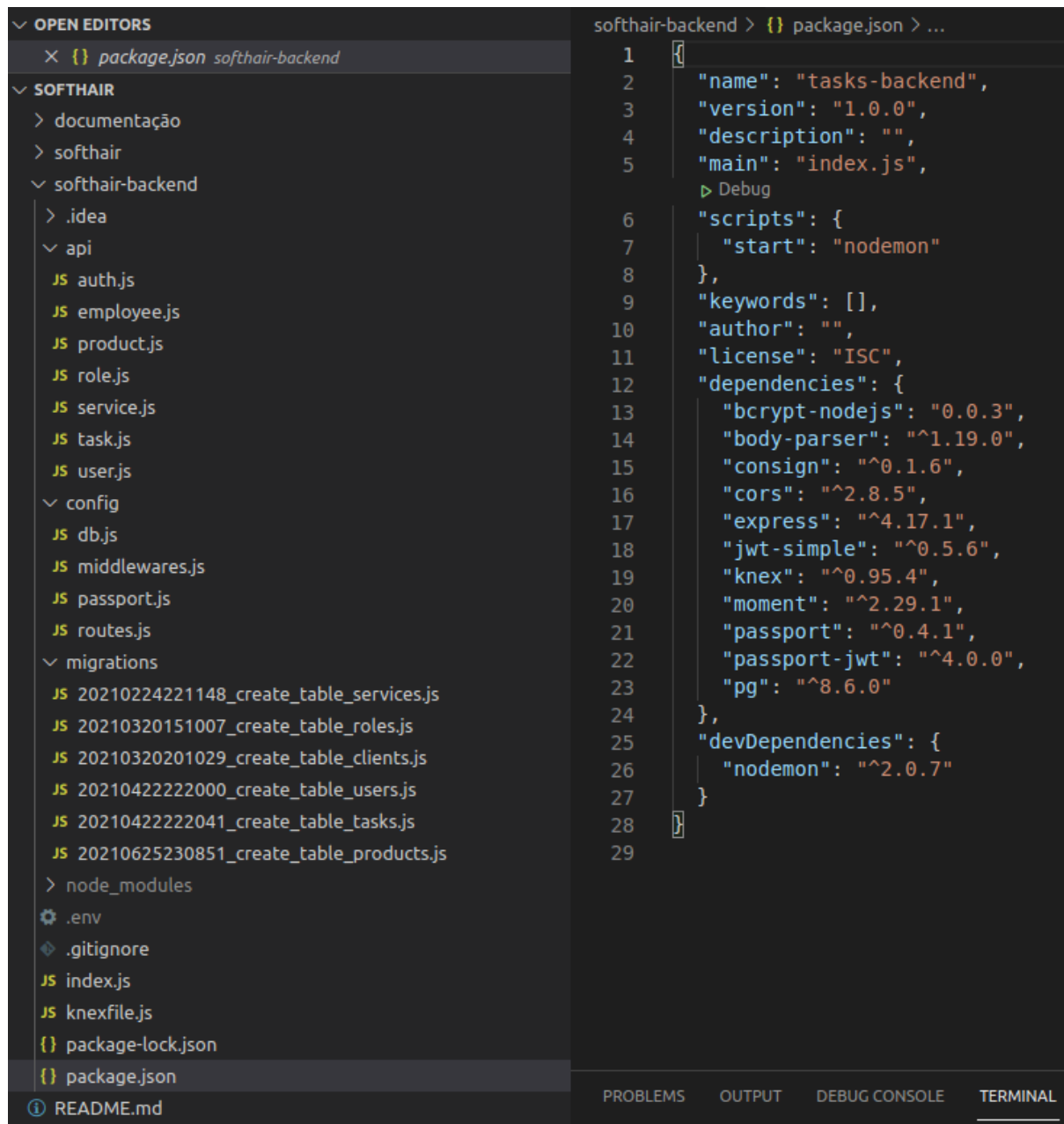


Figura 2. Tela da estrutura de diretórios do projeto da API.

A Figura 2 apresenta como está organizado o projeto no lado servidor conhecido também como *backend*, detalha-se algumas subpastas e arquivos de *scripts*, entre estes o arquivo em destaque é o *package.json* que apresenta as bibliotecas e suas versões utilizadas no desenvolvimento da API, o projeto está disponível através do link, “<https://github.com/luizsett7/softhair>”.

6.4. Testes de comunicação da API pela ferramenta *Postman*

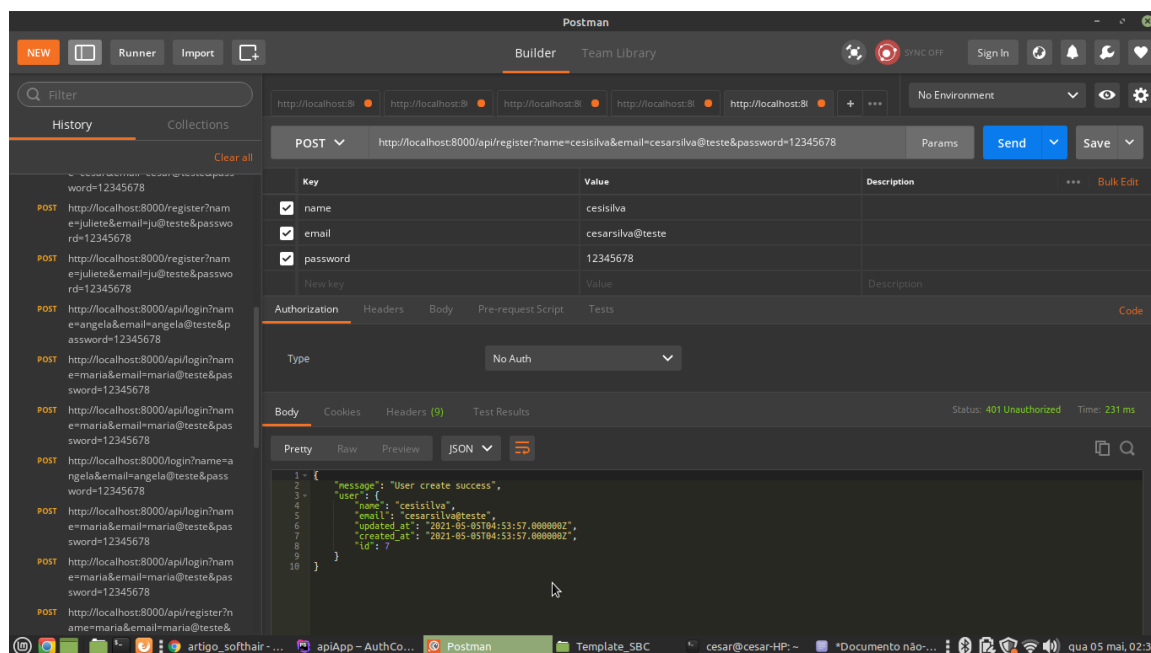


Figura 3. Tela do *Postman* com os resultados das chamadas HTTP.

Com a utilização da ferramenta *Postman*, pode ser realizada os testes das requisições da API, mesmo antes de implementar a interação do aplicativo *front end* com o lado *backend*, passando parâmetros nos campos solicitados e no seu retorno imprime os dados que foram definidos na modelagem, assim as chamadas aos métodos da API podem ser corrigidas e testadas sem necessitar passar pelo aplicativo.

6.5. *JWT (JSON Web Token)*

É um método RCT 7519 padrão da indústria para realizar autenticação entre duas partes por meio de um *token* assinado que autentica uma requisição HTTP. Esse *token* é um código que armazena objetos *JSON* com os dados que permitem a autenticação das requisições.

Através de uma requisição HTTP ao *endpoint* de autenticação da API. Nela o usuário envia, no corpo da requisição dados como *e-mail* e senha.

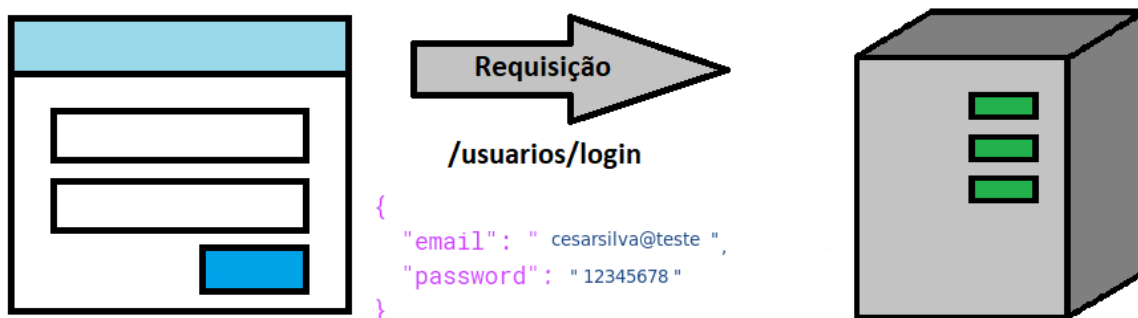


Figura 4. Usuário enviando requisição com dados de autenticação.

Uma vez que os dados enviados pelo usuário tenham sido autenticados no servidor, este criará um *token JWT* assinado com um segredo interno da API e enviará este *token* de volta ao usuário.

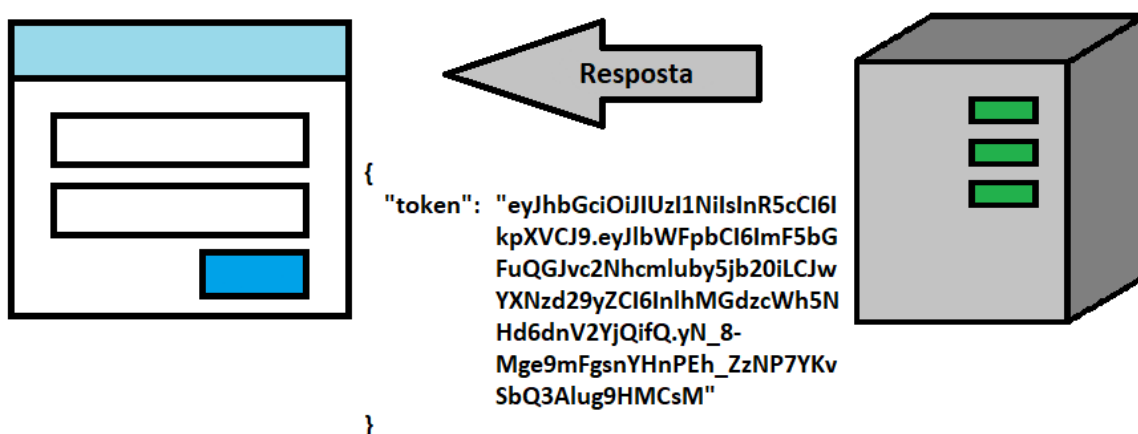


Figura 5. Servidor envia o *token* assinado para o usuário.

Provido com o *token* autenticado, o usuário possui acesso aos *endpoints* da aplicação que antes eram restritos.

6.6. Express.js

O *Express* é um *framework* para aplicativo da *web* do *Node.js*, leve e flexível que fornece um conjunto robusto de recursos para aplicativos *web* e *mobile*.

Com uma miríade de métodos utilitários HTTP e *middleware* a seu dispor, criar uma API robusta é rápido e fácil.

6.7. Knex.js

É um módulo do *Node.js* para fazer a criação de banco de dados e manipulação dos dados, porém com ele podemos manter um padrão de *Querys* no qual, se mudar o tipo do banco de dados ainda assim a aplicação continuará funcionando.

6.8. Nodemon.js

É uma ferramenta que ajuda a desenvolver aplicativos baseados em *Node.js*, reiniciando automaticamente o aplicativo quando mudanças de arquivo no diretório são detectadas.

6.9. Código-fonte

O código-fonte escrito no decorrer da implementação deste projeto está disponível no *GitHub*, um sistema de controle de versões, utilizado no desenvolvimento de software. Endereço do projeto: <https://github.com/luizsett7/softhair>.

7. Especificações do aplicativo *mobile*

7.1. React Native

O *React Native* é um *framework* baseado no *React*, desenvolvido pelo *Facebook*, que permite o desenvolvimento de aplicações *mobile*, para várias plataformas como por exemplo *Android* e *iOS*, utilizando apenas *Javascript*.

Desenvolver *apps* para dispositivos móveis era complexo, pois além de ter que aprender as linguagens *Objective-C* (*iOS*) e *Java* (*Android*), o desenvolvedor não aproveitava o código de uma plataforma para outra, fazendo com que as empresas tivessem equipes distintas de desenvolvimento para cada sistema operacional, tornando o projeto lento e caro. Com a utilização do *React Native*, o código pode ser reaproveitado em até 100% entre as plataformas, podendo fazer com que o custo e a duração do projeto caiam significativamente.

7.2. Prototipação

Para a etapa de construção das telas utilizou-se da ferramenta *BuilderX* que possibilita criar o *design* das interfaces e também conta com o recurso de gerar componentes que podem ser integrados facilmente ao *app*. Um exemplo de componente pode ser visualizado na Figura 6.

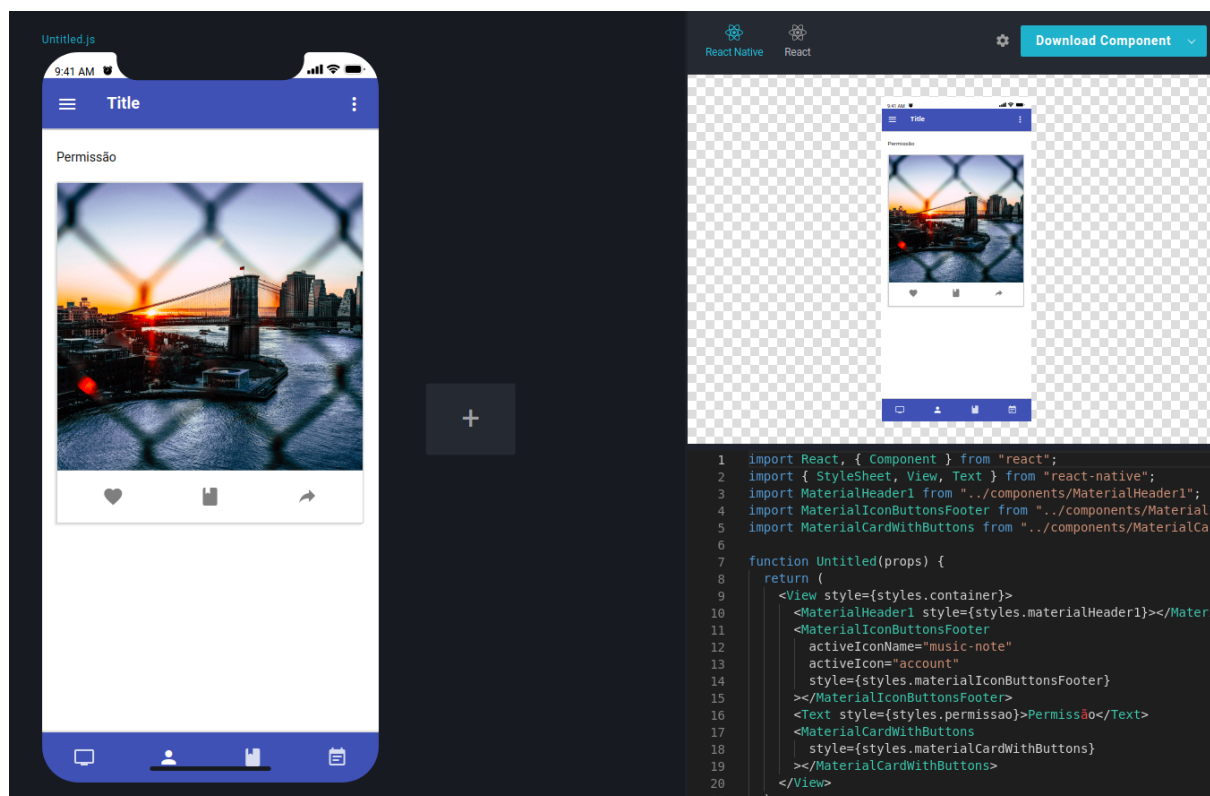


Figura 6. Tela de criação de interfaces do BuilderX.

8. Resultados Obtidos

Apresenta-se nesta seção as telas de interação do usuário com o sistema. Na Figura 7 observa-se a tela de login com os campos obrigatórios *e-mail* e senha e também com a opção de criar uma nova conta.

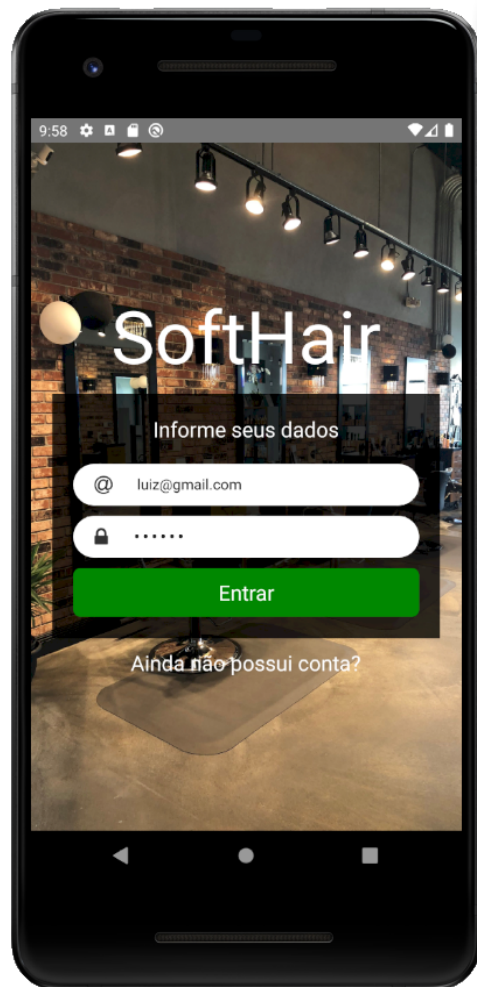


Figura 7. Tela de *Login*.

Conforme mencionado anteriormente no capítulo de API, o *login* visto na Figura 7 utiliza-se do *JWT* para autenticar o usuário e permitir que ele realize as funções do sistema.

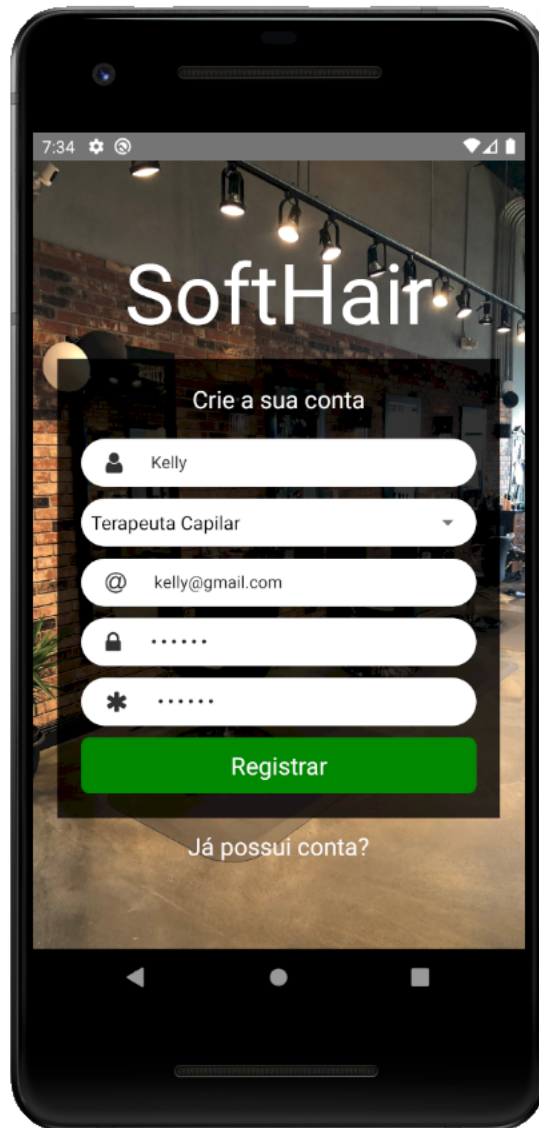


Figura 8. Tela de criação de conta.

Na Figura 8 observa-se os campos necessários para criação de uma nova conta. Segundo a regra de negócio 11 o campo de e-mail deverá ser único sendo definida no banco de dados pelo comando *table.string('email').notNull().unique()*. Com o objetivo de evitar que o campo nome ficasse em branco utilizou-se uma validação (regra de negócio 10) conforme trecho de código a seguir.

```
if(!login.nome || !login.nome.trim()) { // Verifica se existe um nome

    Alert.alert('Dados inválidos', 'Nome não informado!') // Caso não exista exibe um alerta

    return

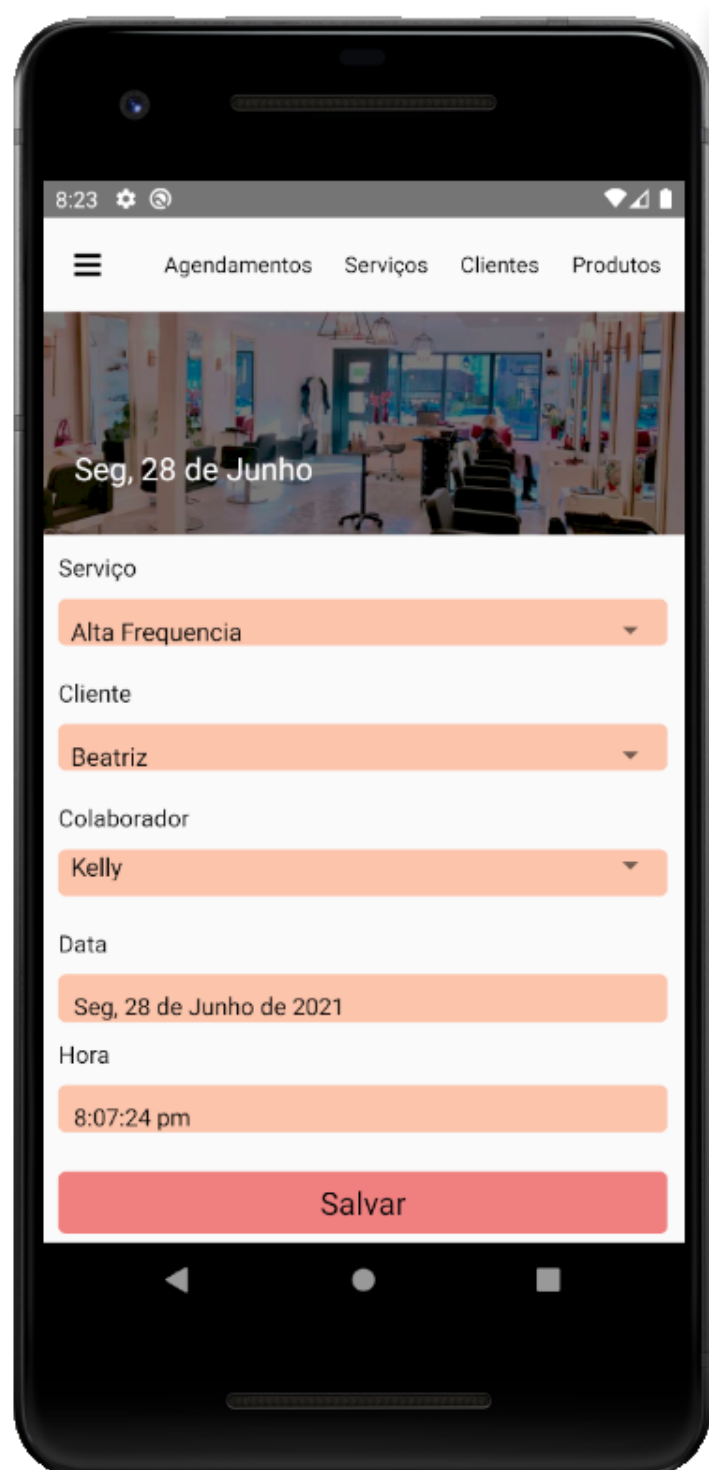
}
```

A Figura 9 demonstra a tela de lista de agendamentos que concede as seguintes funcionalidades de gerenciamento, inserção, visualização, atualização e remoção.



Figura 9. Tela de Lista de Agendamentos.

Na Figura 10 observa-se a tela de inserção de agendamento. Este cadastro possui uma validação na data e hora para evitar agendamentos no mesmo horário, permitindo somente agendamentos no intervalo de trinta minutos, conforme regra de negócio 9.



The image shows a smartphone screen with a mobile application interface. At the top, the status bar shows the time 8:23 and various icons. Below the status bar is a navigation bar with a hamburger menu icon and four tabs: 'Agendamentos', 'Serviços', 'Clientes', and 'Produtos'. The main content area features a background image of a modern interior space. Overlaid on this is a form for creating a new appointment. The form includes a date picker set to 'Seg, 28 de Junho', a service dropdown menu set to 'Alta Frequencia', a client dropdown menu set to 'Beatriz', and a collaborator dropdown menu set to 'Kelly'. Below these are fields for 'Data' (set to 'Seg, 28 de Junho de 2021') and 'Hora' (set to '8:07:24 pm'). A large red 'Salvar' button is at the bottom of the form.

8:23

Agendamentos Serviços Clientes Produtos

Seg, 28 de Junho

Serviço

Alta Frequencia

Cliente

Beatriz

Colaborador

Kelly

Data

Seg, 28 de Junho de 2021

Hora

8:07:24 pm

Salvar

Figura 10. Tela de Cadastro de Agendamento.

Na Figura 11 revela-se a tela de catálogo de produtos.

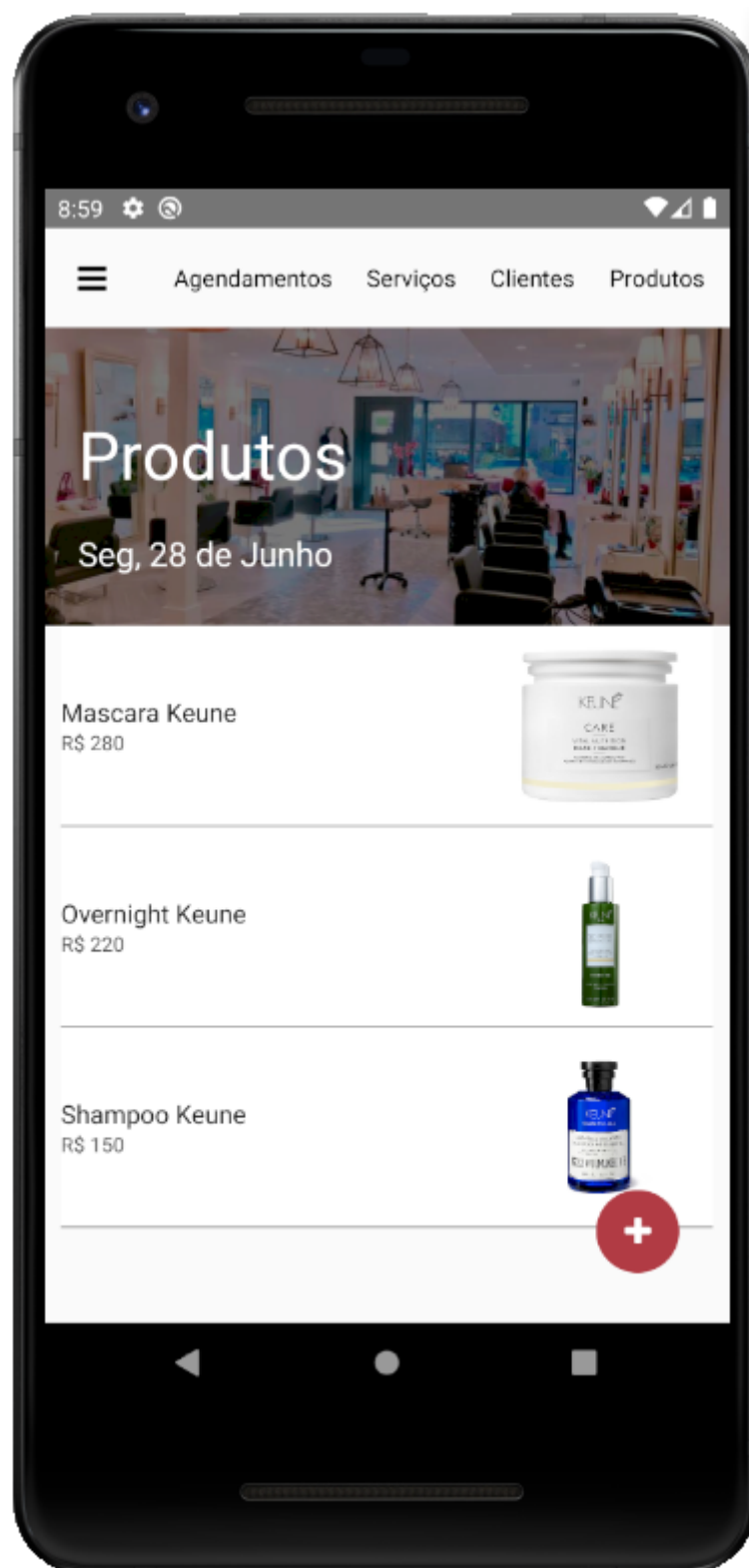


Figura 11. Tela de Catálogo de Produtos.

9. REFERÊNCIAS

BECKER, L. O que é React Native, Rio Grande do Sul, 13 mai. 2021.

Disponível em: <<https://www.organicadigital.com/blog/o-que-e-react-native/>>. Acesso em: 03 jun. 2021.

BuilderX The design tool for Designers and Developers, 2021. Disponível em: <<https://builderx.io/learn>>. Acesso em: 23 jun. 2021.

Express.js Framework, 2021. Disponível em: <<http://expressjs.com/>>. Acesso em: 03 abr. 2021.

FLANAGAN, David. JavaScript: o guia definitivo. Tradução J.E.N. Tortello. 6. ed. Porto Alegre: Bookman, 2013.

FITZPATRICK, K. RedMonk Top 20 Languages Over Time, Portland, 02 mar. 2021.

Disponível em:

<<https://redmonk.com/kfitzpatrick/2021/03/02/redmonk-top-20-languages-over-time-january-2021/>>. Acesso em: 28 abr. 2021.

GitHub code hosting platform, 2008.

Disponível em: <<https://docs.github.com/pt/get-started>>. Acesso em 12 mai. 2021.

HANASHIRO, A. O que se pode fazer com JavaScript hoje em dia? São Paulo, 2018.

Disponível em:

<<https://www.treinaweb.com.br/blog/o-que-se-pode-fazer-com-javascript-hoje-em-dia>>. Acesso em: 26 abr. 2021.

JWT-Auth1.0.5v Json Web Token, 2020.

Disponível em: <<https://jwt-auth.readthedocs.io/en/develop/>>. Acesso em: 08 mai. 2021.

Knex.js Query Builder, 2021. Disponível em: <<http://knexjs.org/>>. Acesso em: 12 jun. 2021.

Node.js Javascript runtime, 2009. Disponível em: <<https://nodejs.org/en/docs/>>. Acesso em: 02 abr. 2021.

Nodemon.js Interface utility, 2021.

Disponível em: <<https://www.npmjs.com/package/nodemon>>. Acesso em: 12 jun. 2021.

O que é JavaScript?, 2021. Disponível em:

<https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript>

Acesso em: 27 abr. 2021.

Postman API Platform, 2012. Disponível em: <<https://learning.postman.com/docs/>>.

Acesso em: 06 abr. 2021.

React Native Framework, 2015. Disponível em: <<https://reactnative.dev/docs/getting-started>>.

Acesso em: 29 mar. 2021.

RIGUES, R. JavaScript se consolida como a linguagem de programação mais popular, São Paulo, 22 out 2020.

Disponível em:

<<https://olhardigital.com.br/2020/10/22/noticias/javascript-se-consolida-como-a-linguagem-d-e-programacao-mais-popular/>>. Acesso em: 27 abr. 2021.

SALOMÃO, R. M. O promissor Mercado de Salões de Beleza do Brasil, Belo Horizonte, 28 jan. 2020.

Disponível em: <<https://buyco.com.br/blog/mercado/mercado-de-saloes-de-beleza>>. Acesso em: 8 mar. 2021.

WEBER, M. O Brasil é o quarto maior mercado de beleza e cuidados pessoais do mundo, São Paulo, 04 jul. 2020. Disponível em:

<<https://www.forbes.com.br/principal/2020/07/brasil-e-o-quarto-maior-mercado-de-beleza-e-cuidados-pessoais-do-mundo>>.