

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

Cesar E. Silva<sup>1</sup>, Luiz Guilherme S. Oliveira<sup>2</sup>

<sup>1</sup>Instituto Federal do Paraná

Caixa Postal 15.064 – 91.501-970 – Foz do Iguaçu – PR – Brazil

**Abstract.** *The beauty area is growing every day, with the increase in the number of customers and also the quantity of products sold, moving the economy of this sector. Brazil is currently the fourth in the world ranking in consumption of personal hygiene products, perfumes and cosmetics (FORBES). Only by this data is it possible to understand the great potential that the segment has. Even in the face of the severe crisis that Brazil faced, the annual average growth of this sector in the last 10 years was 4.1%. In 2019 a survey was commissioned by Beauty Fair (International Professional Beauty Fair) in which around 500 thousand beauty salons were registered all over Brazil. A major factor that has contributed to such an expressive number is the significant increase in franchises in the sector, be it beauty salons, enamels or eyebrow design, for example. In fact, the number of beauty salons in the country is much higher, since almost 50% of the salons are still informal. And as we are in a digital age, where processes take place in an agile way, tools appear to make people's lives easier, including in the area of aesthetics and related services in the area of beauty. In AppSoftHair the proposed theme is that the provider that provides beauty services, will have the functionality to control their customer schedule, the services provided and scheduling and attendance management, which will ultimately add agility and security to the user's routines to the system.*

**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 em fase de elaboração se deve a uma continuidade de um projeto em versão web que foi desenvolvido com a proposta de resolver as dificuldades em controlar as rotinas de um salão de beleza, com as duas modalidades dos sistemas implantados, isto agregará segurança e praticidade na gestão dos serviços prestados.

## 2. Cenário Atual

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, 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 de funções principais do *software*.

- Cadastro de Usuários para acesso ao Sistema;
- Cadastro de 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	Definir acesso e permissões	Administrador concede acesso a determinados recursos do sistema para uma entidade cadastrada.
RF03	Cadastrar serviços	Cadastrar os tipos de serviços realizados pelo <i>App</i> . O sistema permitirá a manutenção de inclusão, alteração,

		consulta e exclusão de dados, essas funcionalidades estarão a cargo do administrador.
RF04	Cadastrar clientes	Esse cadastro será atrelado ao tipo de serviço, ao prestador que irá executar a atividade. A manutenção dos dados pelo sistema permitirá a inclusão, exclusão, consulta e alteração, a cargo do administrador e do prestador de serviço.
RF05	Cadastrar agendamento	Uma atividade pode ser agendada para ser realizada por um prestador. Apenas usuários podem registrar um agendamento.
RF06	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.

## 4.2. Requisitos Não Funcionais

**Tabela 2. Requisitos Não-Funcionais**

Código	Descrição do requisito não-funcional
RNF01	O <i>framework Expo</i> será utilizado no desenvolvimento do aplicativo
RNF02	O sistema utilizará banco de dados <i>MySQL</i>
RNF03	O sistema utilizará API <i>REST</i> para a comunicação entre servidor <i>web</i> e o aplicativo

RNF04	O sistema será desenvolvido em linguagem <i>JavaScript</i>
RNF05	O aplicativo será implementado na plataforma <i>Android</i> na versão 10
RNF06	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
RNF07	Através de validação de <i>token</i> , uma vez que o prestador estiver conectado, cada solicitação irá incluir o <i>JWT</i> , permitindo que o prestador continue acessando o sistema que foram liberados com tal <i>token</i> .

### 4.3. Regras de Negócio

**Tabela 3. Requisitos Não-Funcionais**

Código	Descrição regra de negócio
RN01	O acesso ao servidor <i>web</i> , para evitar manipulação de dados no banco de dados, terá como protocolo de segurança a implementação de geração de <i>token</i> , a fim de barrar esta eventualidade.
RN02	Somente o administrador poderá manipular dados de serviços e prestadores
RN03	A alteração de senha de acesso ao aplicativo somente será realizada pelo prestador
RN04	Somente será permitido iniciar um serviço com agendamento criado
RN05	O agendamento se tornará um atendimento mediante a alteração do <i>status</i> .

RN06	O valor dos serviços prestados serão vinculados ao atendimento caso ele se concretize.
RN07	O agendamento que não se torna um atendimento tem seu status alterado para inativo.
RN08	O prestador de serviço terá um único cliente por atendimento.
RN09	O prestador de serviço terá um atendimento único por horário.

## 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 de *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 Expo 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 versão de código do *GitHub*.

## 6. Resultados e especificações da 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.

O *JavaScript* não se restringe apenas aos navegadores. Essa linguagem possibilita a criação de aplicações nativas. O *React Native* o qual será explicado a seguir e o *NativeScript* são as ferramentas mais importantes para gerar aplicativos para dispositivos móveis. Essas bibliotecas permitem implementar telas com XML, e estilizar com a linguagem CSS, sendo convertidas para telas nativas de cada plataforma, como Android e iOS. As ações são escritas com JavaScript.

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. React Native

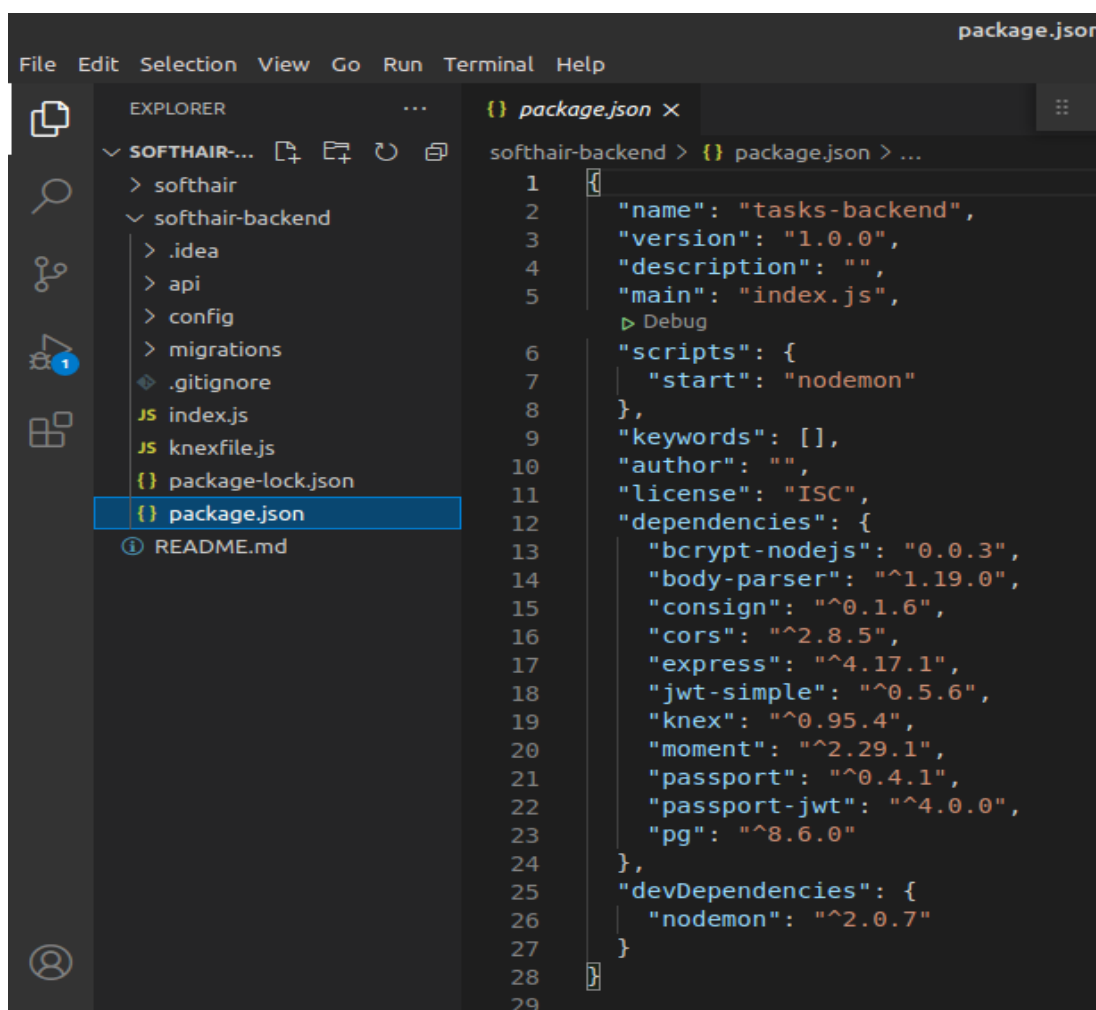
O *React Native* é um *framework* baseado no *React*, desenvolvido pelo *Facebook*, que permite o desenvolvimento de aplicações *mobile*, para várias plataformas como por exemplo *Android* e *iOS*, utilizando apenas *Javascript*.

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.

### 6.3. 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.

#### 6.4. Estrutura da API Rest



The screenshot shows an IDE interface with a dark theme. On the left, the 'EXPLORER' sidebar displays the project structure. The root directory is 'SOFTHAIR-...', which contains a subdirectory 'softhair'. Inside 'softhair', there is a subdirectory 'softhair-backend'. The 'softhair-backend' directory contains several files and subdirectories: '.idea', 'api', 'config', 'migrations', '.gitignore', 'index.js', 'knexfile.js', 'package-lock.json', 'package.json' (highlighted with a blue selection bar), and 'README.md'. On the right, the main editor area shows the content of 'package.json'. The file is titled 'package.json' in the top right corner. The content is a JSON object with the following properties: 'name' (tasks-backend), 'version' (1.0.0), 'description' (empty string), 'main' (index.js), 'scripts' (start: nodemon), 'keywords' (empty array), 'author' (empty string), 'license' (ISC), 'dependencies' (bcrypt-nodejs, body-parser, consign, cors, express, jwt-simple, knex, moment, passport, passport-jwt, pg), and 'devDependencies' (nodemon). The JSON is formatted with line numbers 1 through 29 on the left side of the editor.

```
1 {
2   "name": "tasks-backend",
3   "version": "1.0.0",
4   "description": "",
5   "main": "index.js",
6   "scripts": {
7     "start": "nodemon"
8   },
9   "keywords": [],
10  "author": "",
11  "license": "ISC",
12  "dependencies": {
13    "bcrypt-nodejs": "0.0.3",
14    "body-parser": "^1.19.0",
15    "consign": "^0.1.6",
16    "cors": "^2.8.5",
17    "express": "^4.17.1",
18    "jwt-simple": "^0.5.6",
19    "knex": "^0.95.4",
20    "moment": "^2.29.1",
21    "passport": "^0.4.1",
22    "passport-jwt": "^4.0.0",
23    "pg": "^8.6.0"
24  },
25  "devDependencies": {
26    "nodemon": "^2.0.7"
27  }
28
29 }
```

Figura 1. Tela da estrutura de diretórios do projeto da API.



A figura 1 apresenta como está organizado o projeto no lado *backend*, neste apresenta algumas subpastas e arquivos de *scripts*, entre estes o mais detalhado *package.json*, apresenta as bibliotecas e suas versões utilizadas no desenvolvimento da API.

## 6.5. Testes de comunicação da API pela ferramenta *Postman*

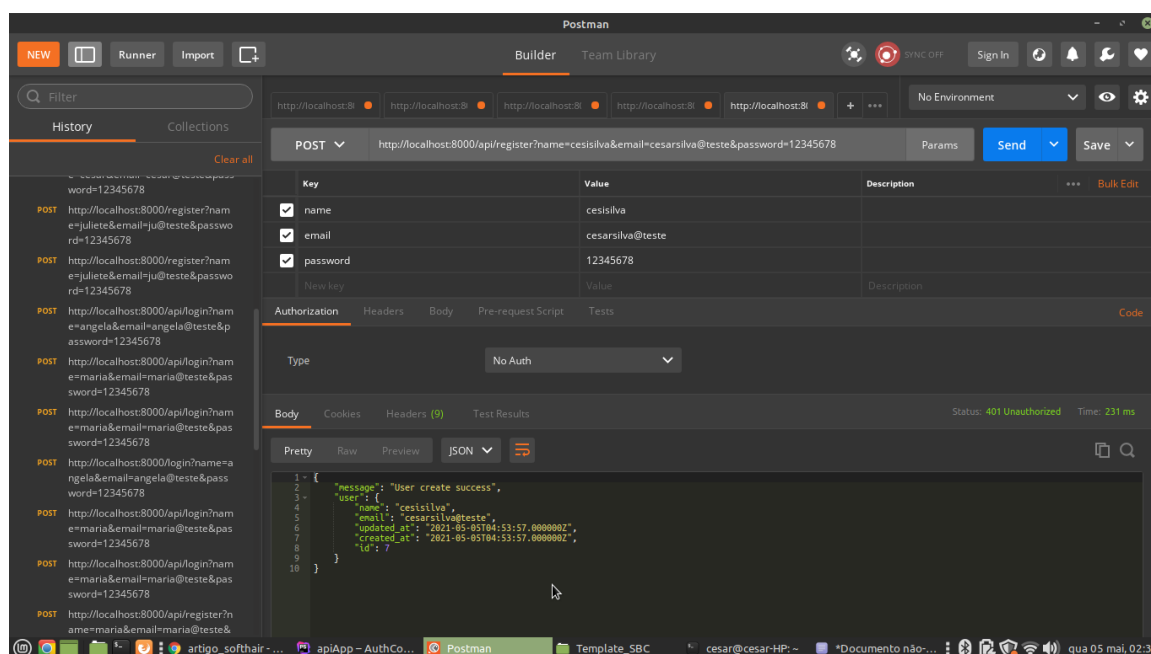


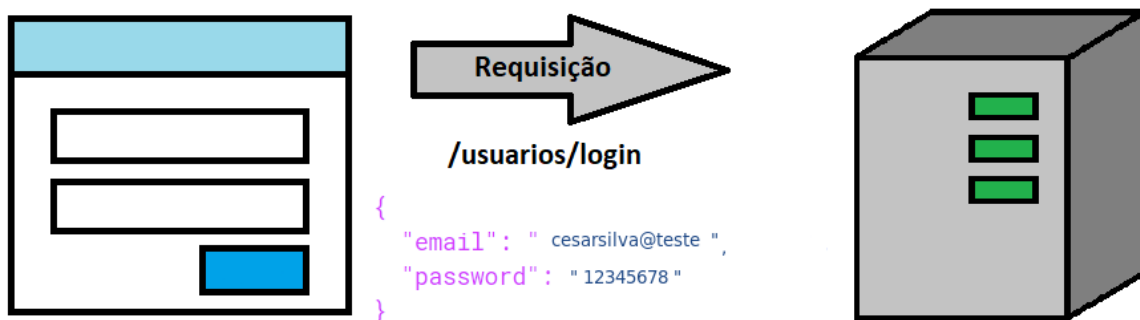
Figura 2. Tela do *Postman* com os resultados das chamadas HTTP.

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.

## 6.6. *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 3. 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 4. 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.

## 6.7. 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.

## 6.8. 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.

## 6.9. 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.

# 7. Especificações do aplicativo *mobile*

## 7.1. Expo CLI

É uma estrutura e uma plataforma para aplicações *React* universais. É um conjunto de ferramentas e serviços criados em torno de plataformas *React Native*, que ajudam a desenvolver, construir, implantar e iterar rapidamente em aplicativos *iOS*, *Android* e *web* a partir da mesma base de código *JavaScript* e *TypeScript*.

## 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 5.

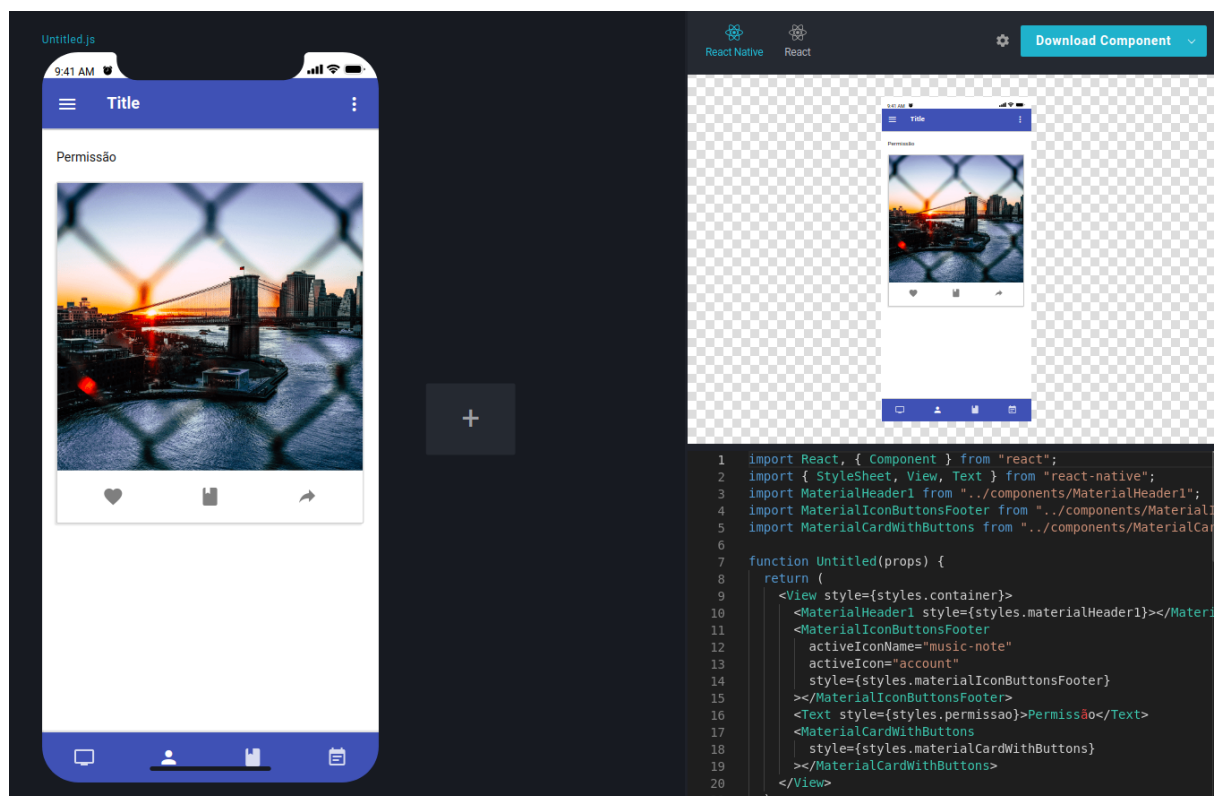
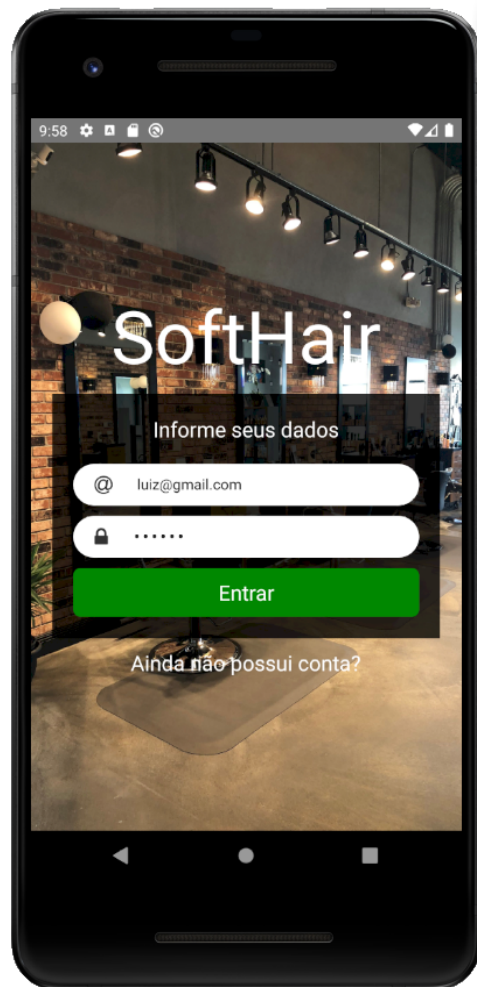


Figura 5. Tela de criação de interfaces do BuilderX.

## 8. Resultados Obtidos

Demonstra-se neste capítulo as telas de interação do usuário com o sistema. Na Figura 6 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 6. Tela de *Login*.**

Conforme mencionado anteriormente no capítulo de API, o *login* visto na Figura 6 utiliza-se do *JWT* para autenticar o usuário e permitir que ele realize as funções do sistema.



**Figura 7. Tela de criação de conta.**

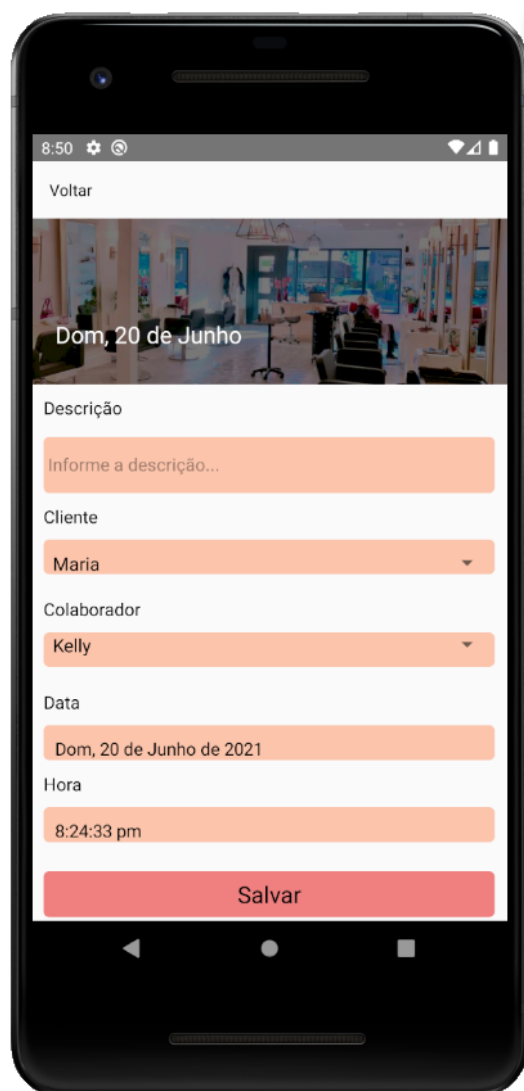
Na Figura 7 observa-se os campos necessários para criação de uma nova conta.

A Figura 8 demonstra a tela de lista de agendamentos que concede as seguintes funcionalidades de gerenciamento, inserção, visualização, atualização e remoção.



**Figura 8. Tela de Lista de Agendamentos.**

Na Figura 9 observa-se a tela de manipulação de atendimento, o agendamento se torna um atendimento e permite que nele sejam acrescentados mais informações como produtos vendidos, serviços que foram realizados além do previsto e uma descrição técnica de como foi realizado o atendimento.



**Figura 9. Tela de Cadastro de Agendamento.**



## 9. Referências

About Node.js (2021), <https://nodejs.org/en/about/>.

BuilderX Introduction (2021), <https://builderx.io/learn>.

David Flanagan. (2012), JavaScript: O Guia Definitivo, Bookman, São Paulo.

Express.js Framework (2021), <http://expressjs.com/>

Expo SDK 40.v Framework documentation (2020), <https://docs.expo.io/>.

JWT-Auth 1.0.5v Json Web Token (2020), <https://jwt-auth.readthedocs.io/en/develop/>.

Knex.js Query Builder(2021), <http://knexjs.org/>

Nodemon.js (2021), <https://www.npmjs.com/package/nodemon>

O que é JavaScript? (2021)

[https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First\\_steps/What\\_is\\_JavaScript](https://developer.mozilla.org/pt-BR/docs/Learn/JavaScript/First_steps/What_is_JavaScript)

O que é React Native. (2021), <https://www.organicadigital.com/blog/o-que-e-react-native/>.

O que se pode fazer com JavaScript hoje em dia? (2018),

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

Postman API Platform version 5.5.5 (2020), <https://learning.postman.com/docs/>.

SALOMÃO, R. M. (2020) “O promissor Mercado de Salões de Beleza do Brasil”.

<https://buyco.com.br/blog/mercado/mercado-de-saloes-de-beleza>.

WEBER, M. (2020) “Brasil é o quarto maior mercado de beleza e cuidados pessoais do mundo”.

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