

**DOCUMENTAÇÃO DO PROJETO MOBILE**

**CURSO: DESENVOLVIMENTO DE SISTEMAS III**

**NOME DO PROJETO: L’Scent Parfum**

**ALUNOS/ EQUIPE:**

**ELIVAN MAIA AZEVEDO (07)**

**LUCAS MATHEUS DA SILVA MOURA (20)**

**PROFESSOR**

**EDUARDO ALVES ALMEIDA**

**RUSSAS- CE**

**JUNHO, 2025**

1. **VISÃO GERAL DO PROJETO**

L’Scent Parfum tem como objetivo facilitar a gestão de vendas e estoque de uma loja de cosméticos, permitindo o controle eficiente de produtos, categorias, cupons de desconto e lembretes internos. O sistema resolve o problema da desorganização e da dificuldade em administrar múltiplas informações em comércios de beleza, oferecendo uma plataforma web prática, segura e funcional. O público-alvo são lojistas e administradores do ramo de cosméticos que buscam mais agilidade, controle e organização no gerenciamento do seu negócio.

1. **FUNCIONALIDADES IMPLEMENTADAS**

* Funcionalidades atuais:
* Tela de login;
* Tela de registro;
* Dashboard;
* Tela de listagem de produtos, categorias, cupons e lembretes;
* Tela de criação de produtos, categorias, cupons e lembretes;
* Tela de edição de produtos, categorias, cupons e lembretes;
* Deletar produtos, categorias, cupons e lembretes;
* Integração com API;
* Banco Supabase;
* Autenticação;
* Integração completa com banco;
* Cadastro com validação.

1. **COMO EXECUTAR O PROJETO**

O projeto L’Scent Parfum utiliza APIs desenvolvidas em Laravel como backend, conectadas a um banco de dados Supabase, onde são armazenadas as informações dos clientes, produtos e pedidos. As rotas da API foram testadas utilizando o Postman para garantir que todos os endpoints funcionam corretamente. Além disso, foi empregado o ngrok para gerar uma URL pública e segura, permitindo que a aplicação local seja acessada pela internet durante o desenvolvimento e testes.

* 1. **FlutterFlow**

Para desenvolver o frontend do L’Scent Parfum, foi utilizada uma plataforma visual baseada em Flutter, que possibilita criar interfaces de forma rápida e intuitiva, sem a necessidade de codificação manual. Essa ferramenta permitiu integrar o design da aplicação com a API desenvolvida em Laravel, utilizando as rotas expostas por meio do ngrok.

* **Link do projeto no FlutterFlow:** https://app.flutterflow.io/share/projeto-elivan-e-lucas-52yrx3
* **Telas disponíveis:** Login, registro, dashboard, listagem de produtos, categorias, cupons e lembretes, criação de produtos, categorias, cupons e lembretes, atualização produtos, categorias, cupons e lembretes.
  1. **API Backend**

O backend da API do projeto L’Scent Parfum foi criado utilizando o framework Laravel, conhecido por sua robustez e ampla utilização. Essa API gerencia as solicitações dos usuários, controla a autenticação e faz a conexão com o banco de dados Supabase.

* **Clone o repositório:** git clone https://github.com/1Lucas-Matheus/API-LScentParfum.git
* **Instalar as dependências:** Depois de clonar o repositório usando o comando git clone, é preciso instalar as dependências para que o projeto funcione corretamente. Projetos que utilizam Node.js possuem um arquivo chamado package.json, que lista todas as bibliotecas necessárias. Essa etapa é fundamental antes de executar o projeto, pois assegura que todos os recursos e ferramentas estejam devidamente configurados no ambiente.
* **Comando:** composer install
* **Executar a API:** Depois de instalar as dependências e preparar o ambiente, a API pode ser executada usando o servidor local do Laravel. Com isso, as rotas passam a responder às solicitações. Uma vez em funcionamento, a API ficará acessível em um endereço local, que pode ser compartilhado na internet através do ngrok.
* **Comando para o servidor local:** php artisan serve
* **AuthToken do ngrok:** ngrok config authtoken 2vGVf8bzOI4WXq2afsxqefvkP18\_3cg836Waw1H46mWER6ZcM.
* **Link da API via ngrok:** https://mainly-awake-minnow.ngrok-free.app.

1. **Estrutura do Banco de Dados (Supabase)**

**Tabela: users (usuários)**

* **Campos:**
* id: Chave primária que identifica cada usuário de forma única;
* name: Nome do usuário;
* email: Endereço de e-mail do usuário (deve ser único);
* email\_verified\_at: Momento em que o e-mail foi confirmado (pode estar em branco);
* password: Senha armazenada de forma segura e criptografada;
* remember\_token: Código utilizado para manter o login automático;
* created\_at / updated\_at: Registros das datas de criação e última modificação do usuário.

**Tabela: categories (categorias)**

* **Campos:**
* id: Código único que identifica a categoria;
* name: Nome da categoria (não pode se repetir);
* created\_at / updated\_at: Datas que indicam quando a categoria foi criada e quando foi alterada pela última vez.

**Tabela: products (produtos)**

* **Campos:**
* id: Identificação única do produto;
* name: Nome do produto (deve ser exclusivo);
* quantity: Quantidade disponível no estoque (valor inteiro);
* price: Valor do produto, com até 6 dígitos no total, sendo 2 casas decimais;
* category\_id: Relacionamento com a tabela de categorias (chave estrangeira);
* promo: Campo opcional que pode indicar um valor promocional (inteiro ou nulo);
* created\_at / updated\_at: Registros das datas de criação e atualização do produto;
* Chave estrangeira: O campo category\_id faz referência ao id da tabela categories, com exclusão em cascata.

**Tabela: coupons (cupons)**

* **Campos:**
* id: Identificação exclusiva do cupom;
* key: Código do cupom (string de até 12 caracteres, exclusivo);
* value: Valor do desconto representado pelo cupom (inteiro e único);
* created\_at / updated\_at: Datas de criação e modificação do registro.

**Tabela: reminders (lembretes)**

* **Campos:**
* id: Código único que identifica o lembrete;
* reminder: Descrição do lembrete (texto);
* date: Data agendada para o lembrete;
* created\_at / updated\_at: Datas de criação e última edição do lembrete.
* **Link do projeto Supabase:** https://supabase.com/dashboard/project/elgfqxfsalecpwrkmtyg

1. **Integração App + API**

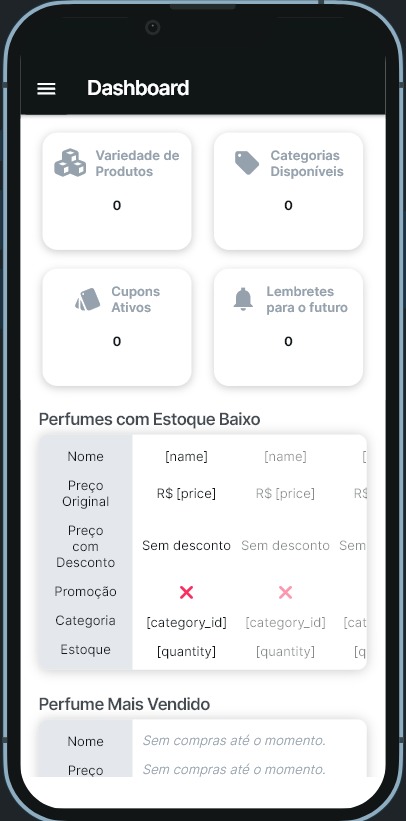
A integração do app com a API foi realizada por meio de chamadas REST em JSON configuradas no FlutterFlow. O app consome os endpoints da API Laravel usando requisições HTTP, com autenticação via token Bearer recebido no login. O token é enviado no cabeçalho Authorization em todas as requisições para garantir segurança. Endpoints Principais Utilizados:

* **Autenticação:**
* POST /login: Autentica usuário e retorna token;
* POST /register: Registra novo usuário;
* POST /logout: Encerra sessão e invalida token.
* **Usuários:**
* GET /users/{id}: Obtém dados do usuário autenticado.
* **Categorias:**
* GET /categories: Lista categorias;
* POST /categories: Cria categoria;
* PUT /categories/{id}: Atualiza categoria;
* DELETE /categories/{id}: Exclui categoria.
* **Produtos:**
* GET /products: Lista produtos;
* POST /products: Cria produto;
* PUT /products/{id}: Atualiza produto;
* DELETE /products/{id}: Exclui produto.
* **Cupons:**
* GET /coupons: Lista cupons;
* POST /coupons: Cria cupom;
* PUT /coupons/{id}: Atualiza cupom;
* DELETE /coupons/{id}: Exclui cupom.
* **Lembretes:**
* GET /reminders: Lista lembretes;
* POST /reminders: Cria lembrete;
* PUT /reminders/{id}: Atualiza lembrete;
* DELETE /reminders/{id}: Exclui lembrete.

Esses endpoints foram integrados diretamente ao FlutterFlow utilizando requisições HTTP configuradas na própria ferramenta. Para assegurar a proteção da API, foram implementadas as seguintes medidas de segurança:

* Autenticação por token: Após efetuar o login, o usuário recebe um token de acesso, que precisa ser incluído em todas as requisições que exigem autenticação.
* Middleware de autenticação no Laravel: Responsável por proteger rotas mais sensíveis, como aquelas relacionadas a informações do usuário e aos produtos, garantindo que apenas usuários autorizados tenham acesso.

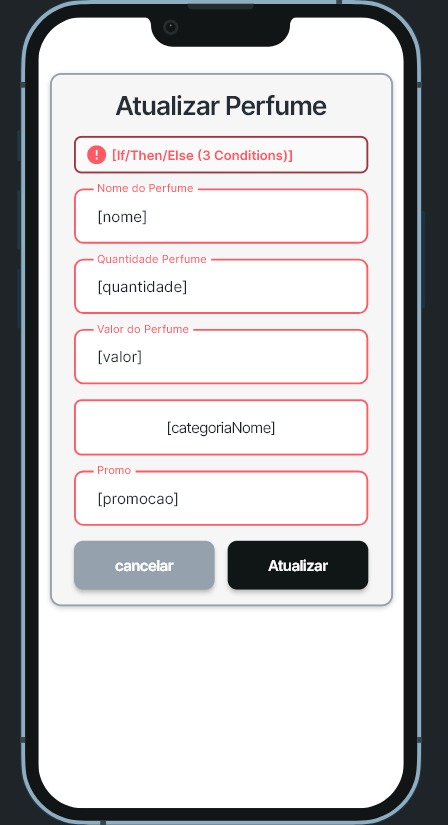
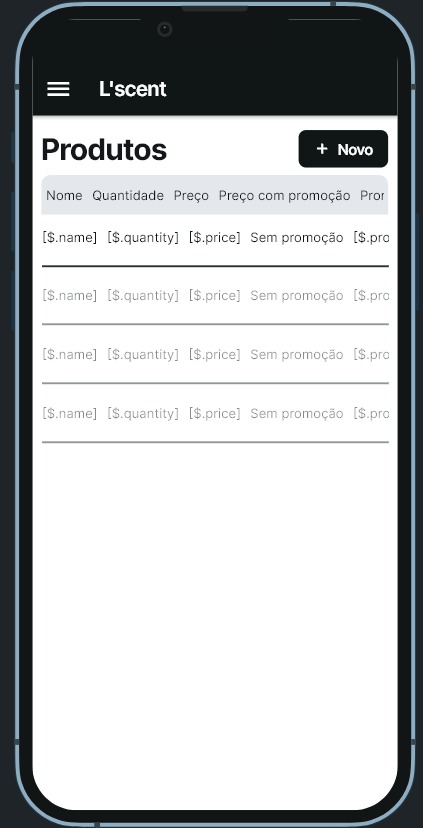
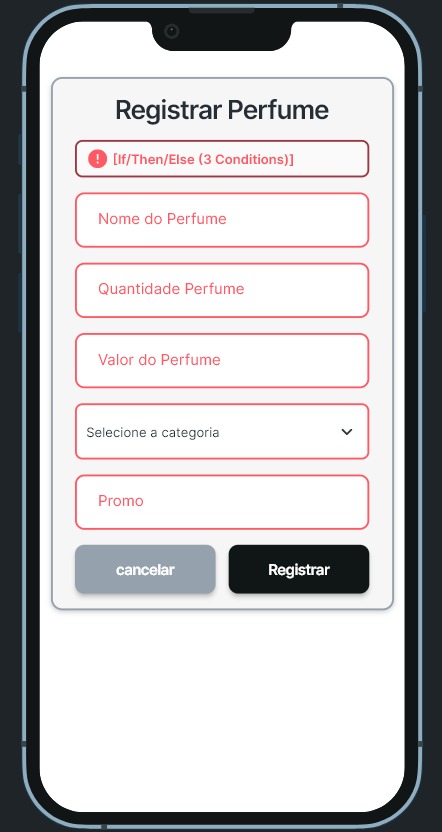
1. **PRINTS DO APP**



Tela de cadastro.

Tela de login.

Dashboard.



Cadastro de produtos.

Edição de produtos.

Read dos produtos.

1. **Exportação de Rotas (Postman)**

No processo de criação da API do projeto PetCharm, foi montada uma coleção no Postman com todas as rotas utilizadas para testes, incluindo cadastro de usuários, autenticação, exibição de serviços e agendamentos. A coleção foi organizada de forma estruturada e, ao final, exportada no formato JSON com o nome:

* LSCent Perfum.postman\_collection.json

Esse arquivo contém todas as informações essenciais para realizar testes na API, como os caminhos das rotas (URLs), os métodos HTTP usados (como GET e POST), os headers das requisições e exemplos de dados enviados em cada operação. Com ele, qualquer desenvolvedor pode importar a coleção no Postman e testar os recursos da API de maneira rápida, sem precisar configurar os testes manualmente. Esse arquivo também pode ser salvo no repositório do projeto no GitHub, facilitando a organização, documentação e o trabalho colaborativo entre os membros da equipe ou desenvolvedores externos.