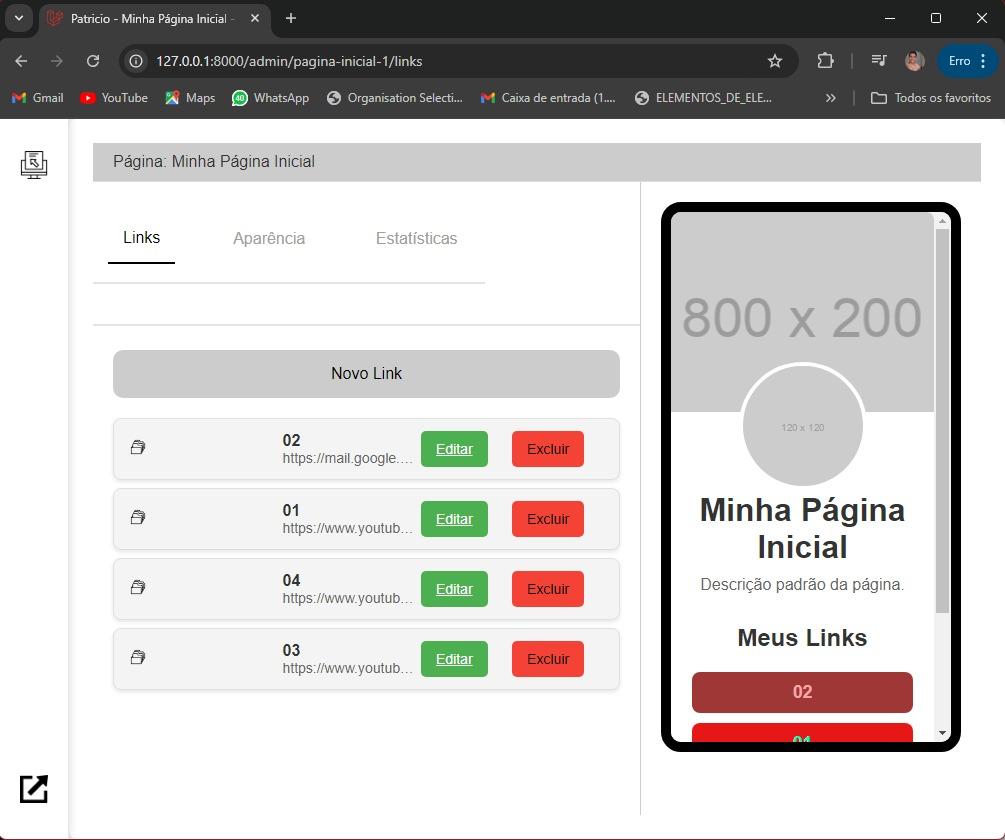
## 



Linktree

28/09/2024

**─**

Matheus Patricio da S e Silva

projeto - CRUD em Laravel

## Sumário:

[Sumário: 2](#_lg6ku2fb12l7)

[**1. Visão Geral 3**](#_45tbmp1q5mrp)

[**2. Objetivos 4**](#_3at9u9s4e0vp)

[**3. Especificações 4**](#_4p7xi5bvhxdr)

[**4. Documentação das Rotas 5**](#_7v98aechho4o)

[**4.1 Rotas Públicas 5**](#_g47gbfouj425)

[Rota Principal 5](#_a5eitc4zb6z0)

[**4.2 Rotas de Administração 5**](#_307e7tfquy38)

[Grupo de Rotas de Administração 5](#_jc8xi99kb2pc)

[**4.2.1 Autenticação 5**](#_yqc72z7h3c6g)

[**4.2.2 Rotas Protegidas 6**](#_b66ygcntocf5)

[Gerenciamento de Páginas 6](#_m0fgr631njd0)

[Gerenciamento de Links 6](#_1ydwbhjagz4l)

[**4.3 Páginas Dinâmicas 7**](#_pz46ewufjgv2)

[**5. Documentação do Login 8**](#_28rshnp03ckp)

[Objetivo 8](#_hhmhedy0gx0i)

[Especificações 8](#_v5vack8iqq7r)

[**5.1 loginAction 9**](#_rida0xsz2riu)

[Funcionalidade 9](#_5bnlvi8ut474)

[Importância 9](#_8tsors65th96)

[Implementação 10](#_59717rlkeo23)

[**5.2 Medidas de Segurança na Autenticação 11**](#_xmduhtd7ifdm)

[Tratamento de Credenciais Incorretas 11](#_d9xhvgy13bc1)

[**6. Registro de Usuários 13**](#_b873qzu7abnl)

[**6.1 Método register 13**](#_1bu5e6y6yfv1)

[Descrição: 13](#_qwj6uozftezc)

[Explicação do Processo de Registro 13](#_7k0fhwc331nu)

[**6.2 Método registerAction 15**](#_7w4a3y35rvkn)

[● Explicação do registerAction 16](#_d8p2y51e60ua)

[Código: 16](#_s619yjo1owue)

[**7. Estrutura da Página e Funcionalidades Principais: 18**](#_lsml0s70td7i)

[**7.1 Organização Visual e Usabilidade 18**](#_xxj9cfw8sthb)

[**7.2 Documentação das Funcionalidades 20**](#_cwgovtbnwo4u)

[**7.3 Script de Ordenação com Sortable.js 21**](#_z177lzr99262)

[**8. Métodos de Edição de Links no AdminController 23**](#_byxx3bislkuj)

[**8.1 Métodos de Edição de Links 23**](#_2af411wghuyl)

[8.2 editLink($slug, $linkid) 23](#_a49fr0rntr1)

[8.3 editLinkAction($slug, $linkid, Request $request) 23](#_grvdr2a384e4)

[**8.4. Exemplos de Configuração de Cores 24**](#_3b5m7dghj0e)

[**8.5. Campos Disponíveis para Edição 24**](#_pwpj0ur49u5)

[**9. Método delLink - Funcionalidade Excluir Link 27**](#_hfxoru2696v)

[Funcionamento 28](#_q7xo5nk649se)

[**10. Controladores 29**](#_opsmo13ez6d1)

[**10.1 AdminController 29**](#_boj83a83d53d)

[**10.2 PageController 38**](#_9b26lxix6yd9)

[Método index 39](#_qppp349e6wuv)

[Método createPage 39](#_uznyli98rww8)

[**11.Modelos Blade e Estrutura Visual 42**](#_qfxak1oo6ac2)

[**11.1Index.blade.php 42**](#_if83m1o8wawn)

[11.2 Visão Geral 42](#_y8r9j7rj2k9h)

[Observações Importantes 43](#_nv52n6bne60s)

[Dependências 43](#_5zgqgyf65yz)

[**11.3 page\_editlink.blade.php 44**](#_z8s2yffbmim3)

[page\_editlink.blade.php 44](#_s41tc3mo3xhw)

[Visão Geral 44](#_6lnw25oeji9v)

[**11.4 page\_links.blade.php 45**](#_bf2owy7dy5ee)

[page\_links.blade.php 45](#_kxdgzqozywrr)

[Visão Geral 45](#_nr8zwiqs9d2p)

[Estrutura 46](#_2szn796m1t94)

[**11.5 page.blade.php 47**](#_g9zzxa16op62)

[Visão Geral 47](#_af76shu68kxy)

[11.6 Menu de Navegação Lateral 47](#_gwelirqa4nvp)

[**11.7 template.blade.php 48**](#_r13ipf1eya42)

[Visão Geral 48](#_d5xq7cdi0dw4)

[**11.8 notfound.blade.php 49**](#_ujiidc6ma9kr)

[Visão Geral 49](#_n9z2j19p18qm)

## 

# 1. Visão Geral

Este projeto é um sistema CRUD (Create, Read, Update, Delete) desenvolvido em Laravel, permitindo que usuários gerenciem links de maneira simples e intuitiva, semelhante ao Linktree.

# 2. Objetivos

* O objetivo deste projeto é desenvolver uma aplicação web que funcione como um serviço de gerenciamento de links, semelhante ao Linktree. A aplicação permitirá que os usuários criem, visualizem, editem e excluam links, proporcionando uma interface intuitiva e responsiva. O foco é facilitar a organização e o compartilhamento de links em um único local, melhorando a acessibilidade e a experiência do usuário.

# 3. Especificações

**Plataforma**: Aplicação web acessível via navegadores.

**Tecnologia**:

* Backend: Laravel (PHP)
* Frontend: Blade (motor de templates do Laravel), HTML, CSS (Bootstrap)
* Banco de Dados: MySQL

**Funcionalidades**:

* Adição de links com título e URL.
* Exibição de todos os links em uma lista.
* Edição de links existentes.
* Exclusão de links.

**Requisitos**:

* PHP 8.0 ou superior.
* Composer instalado.
* Servidor web (por exemplo, Apache ou Nginx) para execução.

**Interface**: Responsiva, utilizando Bootstrap para uma melhor experiência em dispositivos móveis e desktops.

# 4. Documentação das Rotas

Esta documentação descreve as rotas da aplicação Laravel, incluindo rotas principais e rotas administrativas. As rotas são organizadas em dois grupos: rotas públicas e rotas de administração.

### **4.1 Rotas Públicas**

### **Rota Principal**

* **Método:** GET
* **URL:** /
* **Controlador:** HomeController@index
* **Descrição:** Renderiza a página inicial da aplicação.

### **4.2 Rotas de Administração**

### **Grupo de Rotas de Administração**

* **Prefixo:** /admin

### **4.2.1 Autenticação**

* **Rota de Login**
  + **Método:** GET
  + **URL:** /admin/login
  + **Controlador:** AdminController@login
  + **Nome da Rota:** login
  + **Descrição:** Exibe o formulário de login para administradores.
* **Rota de Ação de Login**
  + **Método:** POST
  + **URL:** /admin/login
  + **Controlador:** AdminController@loginAction
  + **Descrição:** Processa o envio do formulário de login.
* **Rota de Registro**
  + **Método:** GET
  + **URL:** /admin/register
  + **Controlador:** AdminController@register
  + **Nome da Rota:** register
  + **Descrição:** Exibe o formulário de registro para novos administradores.
* **Rota de Ação de Registro**
  + **Método:** POST
  + **URL:** /admin/register
  + **Controlador:** AdminController@registerAction
  + **Descrição:** Processa o envio do formulário de registro.
* **Rota de Logout**
  + **Método:** GET
  + **URL:** /admin/logout
  + **Controlador:** AdminController@logout
  + **Descrição:** Realiza o logout do administrador.

### **4.2.2 Rotas Protegidas**

* **Rota do Painel Administrativo**
  + **Método:** GET
  + **URL:** /admin/
  + **Controlador:** AdminController@index
  + **Middleware:** auth
  + **Descrição:** Exibe o painel administrativo. Esta rota requer autenticação.

#### **Gerenciamento de Páginas**

* **Links da Página**
  + **Método:** GET
  + **URL:** /admin/{slug}/links
  + **Controlador:** AdminController@pageLinks
  + **Descrição:** Exibe os links da página especificada pelo slug.
* **Design da Página**
  + **Método:** GET
  + **URL:** /admin/{slug}/design
  + **Controlador:** AdminController@pageDesign
  + **Descrição:** Exibe a página de design da página especificada pelo slug.
* **Estatísticas da Página**
  + **Método:** GET
  + **URL:** /admin/{slug}/stats
  + **Controlador:** AdminController@pageStats
  + **Descrição:** Exibe as estatísticas da página especificada pelo slug.

#### **Gerenciamento de Links**

* **Atualização da Ordem dos Links**
  + **Método:** GET
  + **URL:** /admin/linkorder/{linkid}/{pos}
  + **Controlador:** AdminController@linkOrderUpdate
  + **Descrição:** Atualiza a ordem dos links.
* **Criar Novo Link**
  + **Método:** GET
  + **URL:** /admin/{slug}/newlink
  + **Controlador:** AdminController@newLink
  + **Descrição:** Exibe o formulário para criar um novo link.
* **Ação de Criar Novo Link**
  + **Método:** POST
  + **URL:** /admin/{slug}/newlink
  + **Controlador:** AdminController@newLinkAction
  + **Descrição:** Processa o envio do formulário para criar um novo link.
* **Editar Link**
  + **Método:** GET
  + **URL:** /admin/{slug}/editlink/{linkid}
  + **Controlador:** AdminController@editLink
  + **Descrição:** Exibe o formulário para editar o link especificado.
* **Ação de Editar Link**
  + **Método:** POST
  + **URL:** /admin/{slug}/editlink/{linkid}
  + **Controlador:** AdminController@editLinkAction
  + **Descrição:** Processa o envio do formulário para editar o link.
* **Deletar Link**
  + **Método:** DELETE
  + **URL:** /admin/{slug}/dellink/{linkid}
  + **Controlador:** AdminController@delLink
  + **Descrição:** Deleta o link especificado.

### **4.3 Páginas Dinâmicas**

* **Método:** GET
* **URL:** /{slug}
* **Controlador:** PageController@index
* **Descrição:** Renderiza uma página dinâmica com base no slug. Garante que o slug não comece com admin.



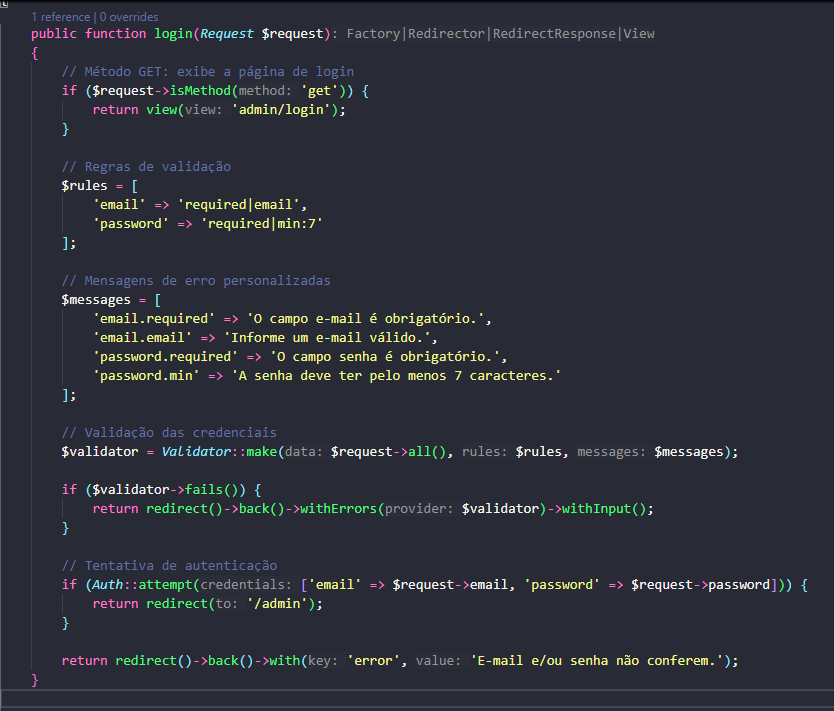
# 5. Documentação do Login

#### **Objetivo**

O objetivo desta seção é descrever o funcionamento do sistema de login da aplicação, detalhando a lógica implementada no código e a URL de acesso para os usuários.

#### **Especificações**

* **URL de Acesso**: O endpoint para acessar a página de login é /admin/login.
* **Métodos**: O sistema suporta dois métodos HTTP para login: GET e POST. O método GET é utilizado para exibir o formulário de login, enquanto o POST é utilizado para processar as credenciais fornecidas pelo usuário.



### **5.1 loginAction**

### **Funcionalidade**

O método loginAction é responsável por gerenciar a autenticação de usuários no sistema. Ele processa as credenciais (e-mail e senha) fornecidas pelo usuário e tenta autenticar o acesso ao painel administrativo.

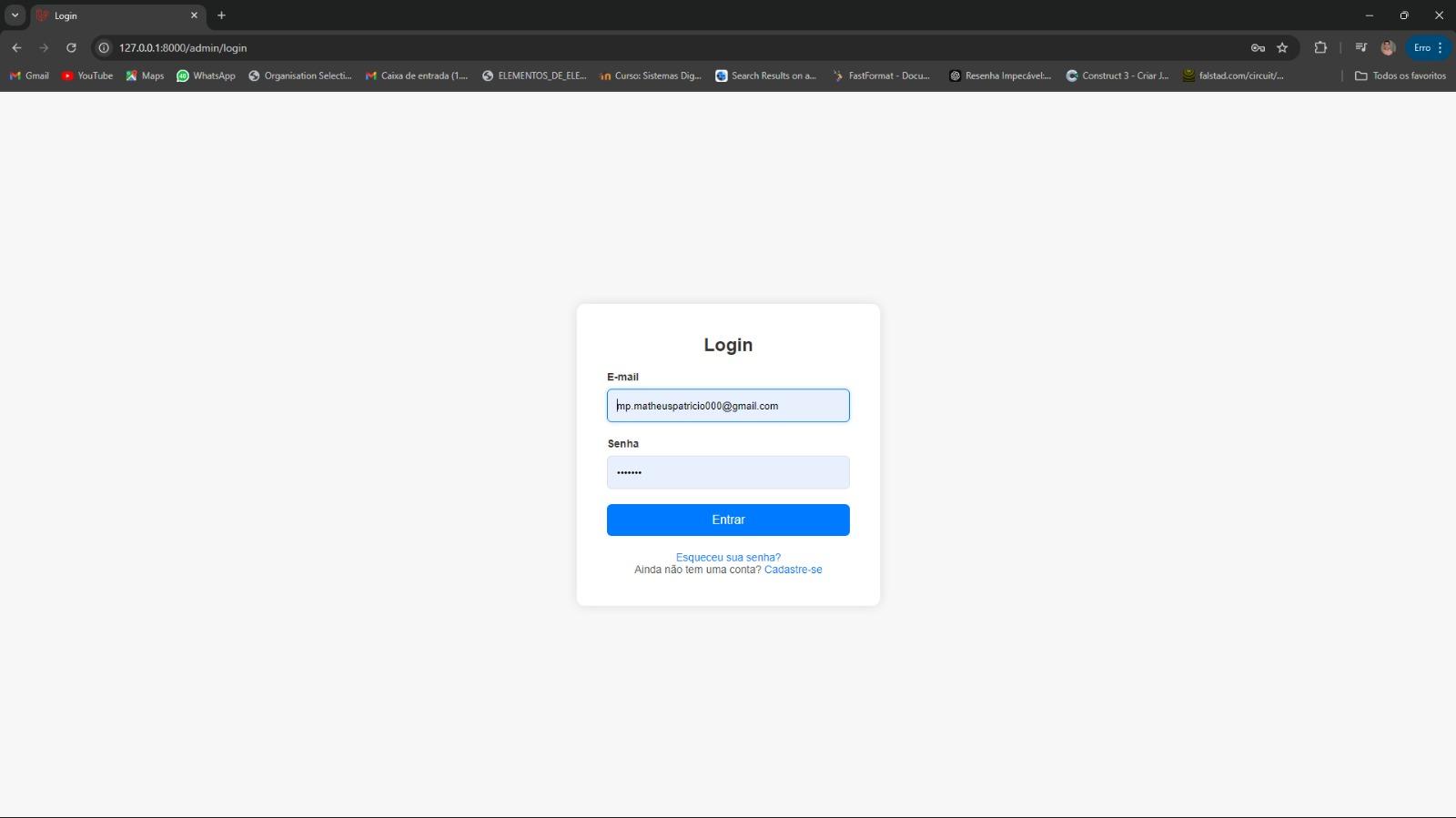
### **Importância**

* **Segurança**: Garante que apenas usuários autenticados tenham acesso à área administrativa, protegendo informações sensíveis.
* **Validação de Credenciais**: Valida as credenciais do usuário e fornece feedback adequado em caso de falhas, como e-mail ou senha incorretos.
* **Redirecionamento**: Após a autenticação bem-sucedida, redireciona o usuário para o painel administrativo, melhorando a experiência do usuário.

### **Implementação**

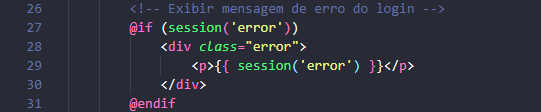
1. **Coleta de Credenciais**: Recebe as credenciais do usuário a partir da requisição.
2. **Autenticação**: Utiliza o método Auth::attempt para validar as credenciais.
3. **Redirecionamento**:
   * Se a autenticação for bem-sucedida, o usuário é redirecionado para /admin.
   * Se falhar, o usuário é redirecionado de volta para a página de login com uma mensagem de erro.

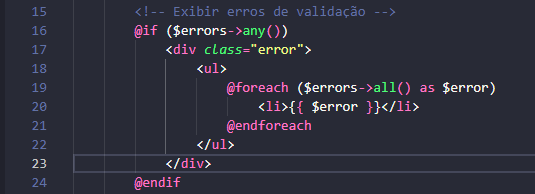




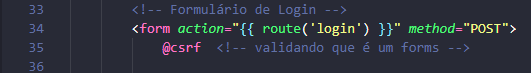
### **5.2 Medidas de Segurança na Autenticação**

### **Tratamento de Credenciais Incorretas**

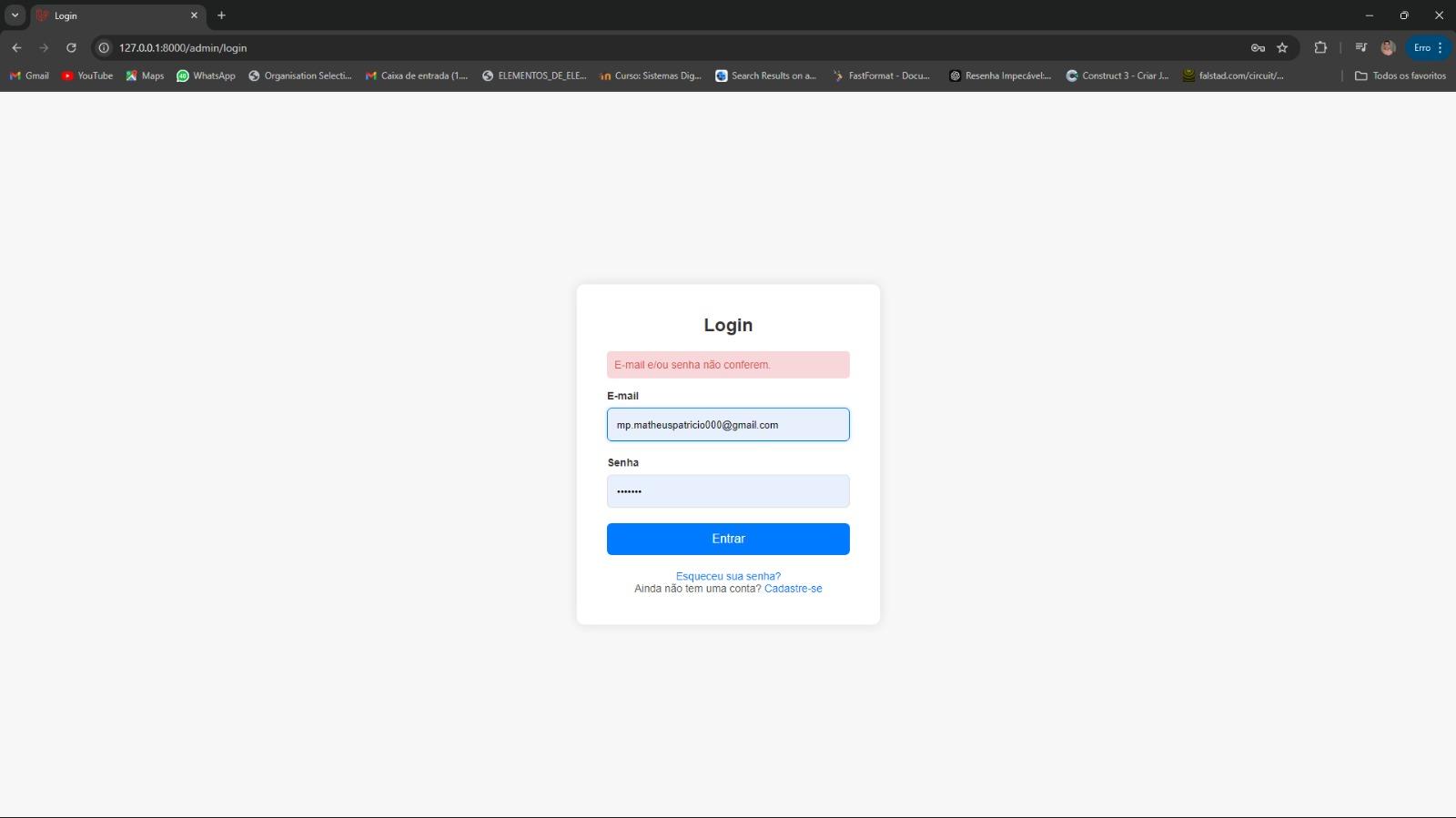
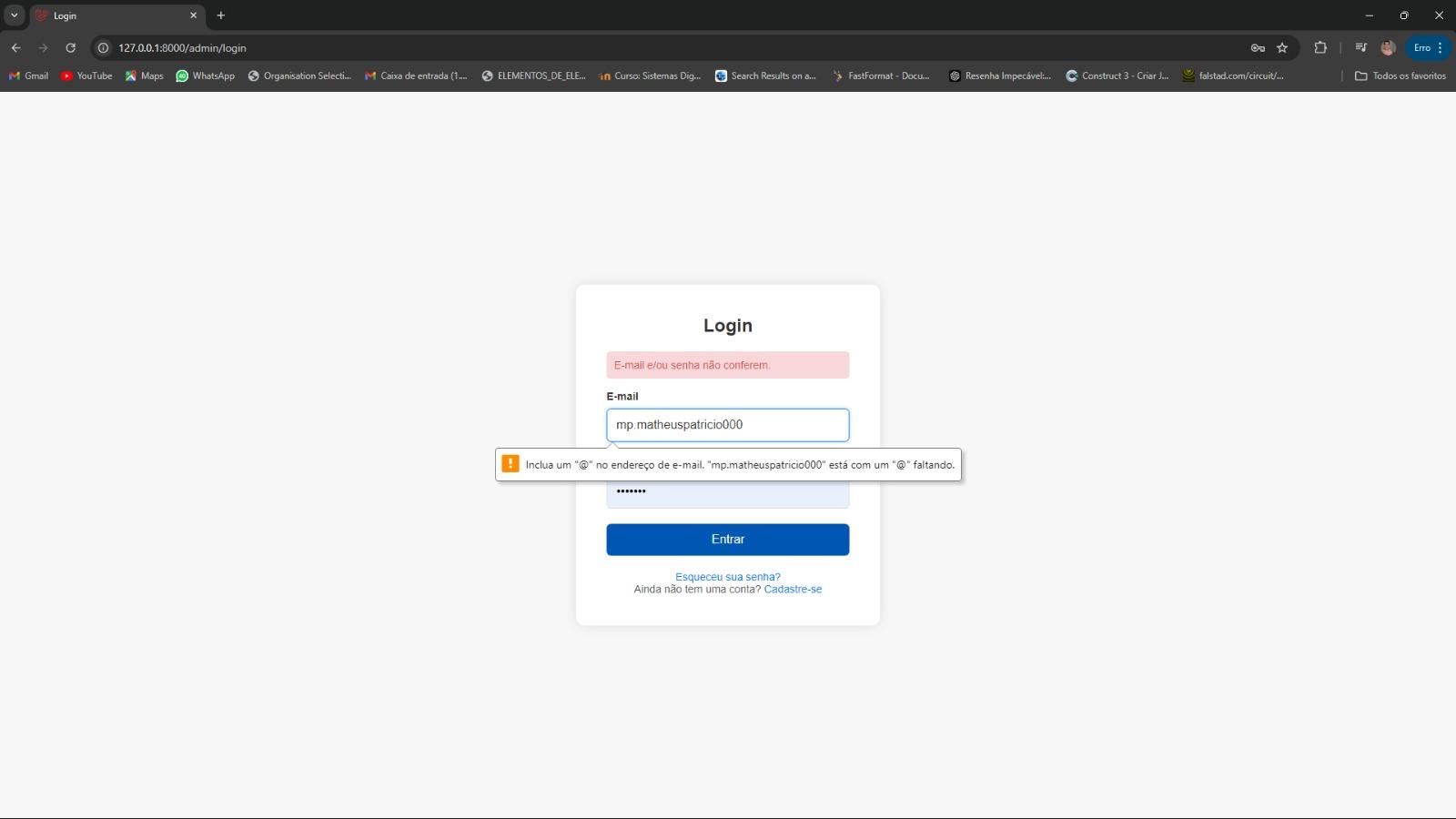
* **Mensagens de Erro**:
  + Mensagens de erro são exibidas ao usuário quando o login falha.
* **Validação de Entrada**:
  + O sistema verifica se os campos obrigatórios (e-mail e senha) estão preenchidos, evitando dados vazios.



* **Proteção contra Força Bruta**:
  + Implementação de bloqueio temporário após várias tentativas de login incorretas.
* **Token CSRF**:
  + Uso de token CSRF (@csrf) no formulário para proteger contra ataques de Cross-Site Request Forgery.



* **Armazenamento Seguro de Senhas**:
  + Senhas são armazenadas usando técnicas de hash, como o bcrypt.
* **Redirecionamento Seguro**:
  + Após o login bem-sucedido, o usuário é redirecionado para uma página segura.



# 6. Registro de Usuários

### **6.1 Método register**

Este método exibe a página de registro para requisições do tipo GET e processa o registro do usuário para requisições do tipo POST.

#### **Descrição:**

* **GET**: Exibe a view admin/register com o formulário de registro.
* **POST**: Realiza a validação dos dados e, em caso de sucesso, cria um novo usuário no banco de dados, logando-o automaticamente após o registro.

### **Explicação do Processo de Registro**

* **Exibição do Formulário de Registro**:
  + Caso a requisição seja do tipo **GET**, o método retorna a view admin/register, onde o formulário de registro é exibido para o usuário.
* **Validação de Dados**:

O método define regras e mensagens de erro personalizadas para validar os campos de entrada:

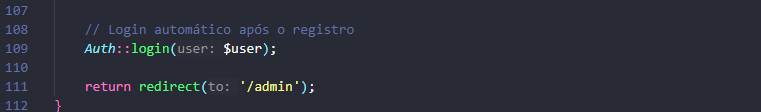
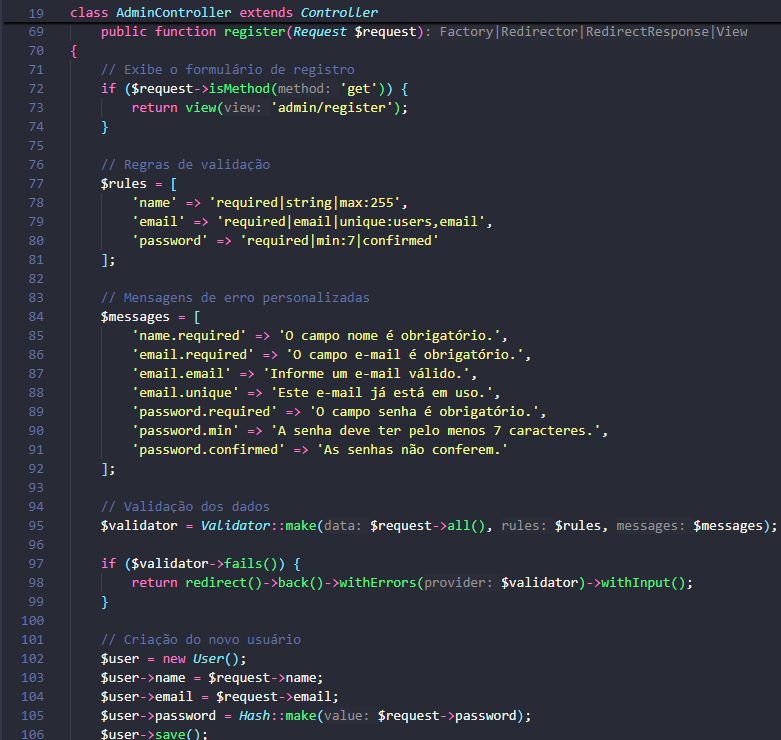
* + - name: Obrigatório, tipo string e com um tamanho máximo de 255 caracteres.
    - email: Obrigatório, em formato de e-mail e único na tabela users.
    - password: Obrigatório, com um tamanho mínimo de 7 caracteres, e que deve ser confirmado.

Em caso de erro de validação, o usuário é redirecionado de volta ao formulário com os erros exibidos.

**Criação do Usuário**:

* + - Se os dados são válidos, cria-se uma instância do modelo User:
    - name, email e password são atribuídos aos atributos do usuário.
    - A senha é criptografada utilizando Hash::make antes de ser salva.

Após a criação do usuário, o sistema faz o login automático e redireciona para a página /admin.

**Código:**

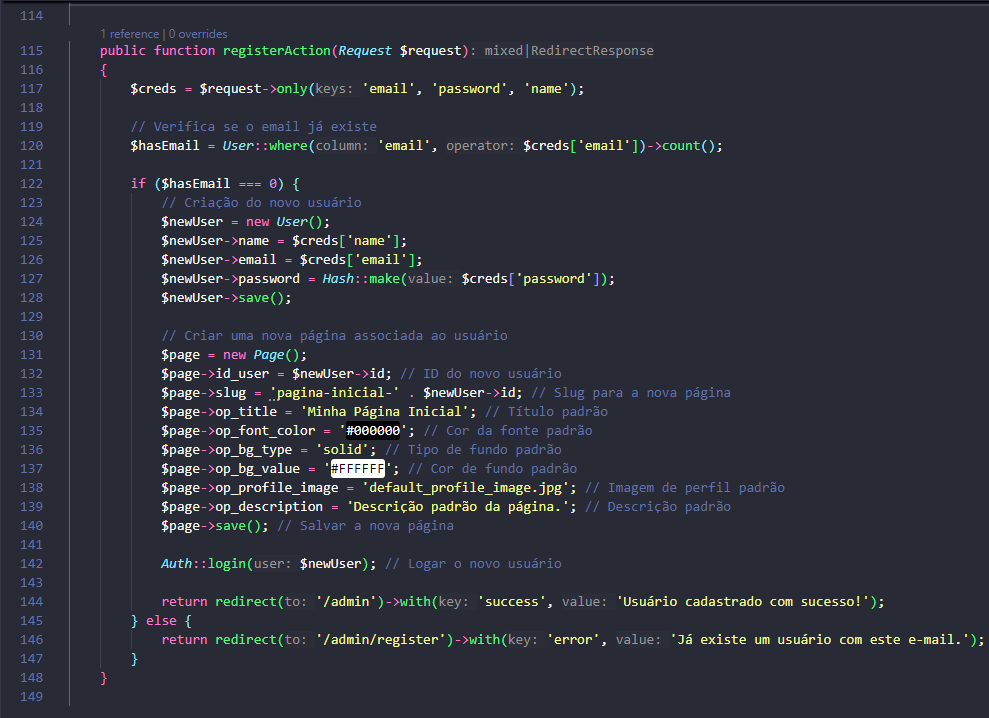
### **6.2 Método registerAction**

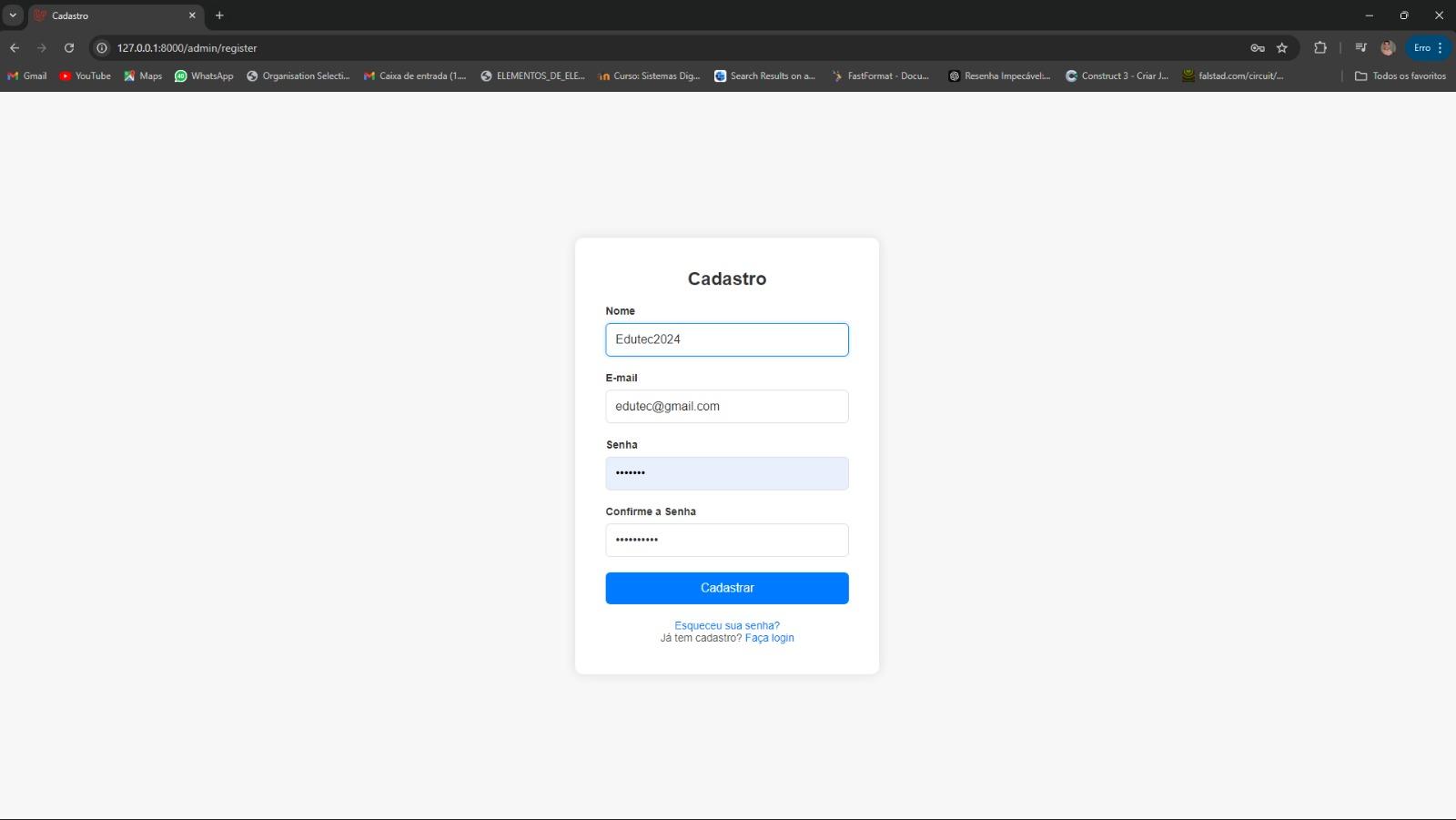
Este método alternativo de registro verifica se o e-mail já existe antes de cadastrar um novo usuário. Além disso, ele cria uma página associada ao novo usuário.

#### **Explicação do registerAction**

* **Verificação de Existência de E-mail**:
  + O método registerAction primeiro verifica se o e-mail fornecido já existe na tabela users.
  + Caso o e-mail já exista, o usuário é redirecionado para a página de registro com uma mensagem de erro.
* **Criação do Usuário e Página Associada**:
  + Se o e-mail é único, o método cria um novo usuário e uma **página personalizada** para ele.
  + Esta página associada recebe:
    - slug: Gerado automaticamente com o ID do usuário.
    - Configurações padrão, como op\_title, op\_font\_color, op\_bg\_type, op\_bg\_value, op\_profile\_image e op\_description.
  + Após salvar a página, o usuário é logado automaticamente.
* **Redirecionamento Final**:
  + Após o cadastro e login, o sistema redireciona o usuário para /admin com uma mensagem de sucesso.

#### **Código:**



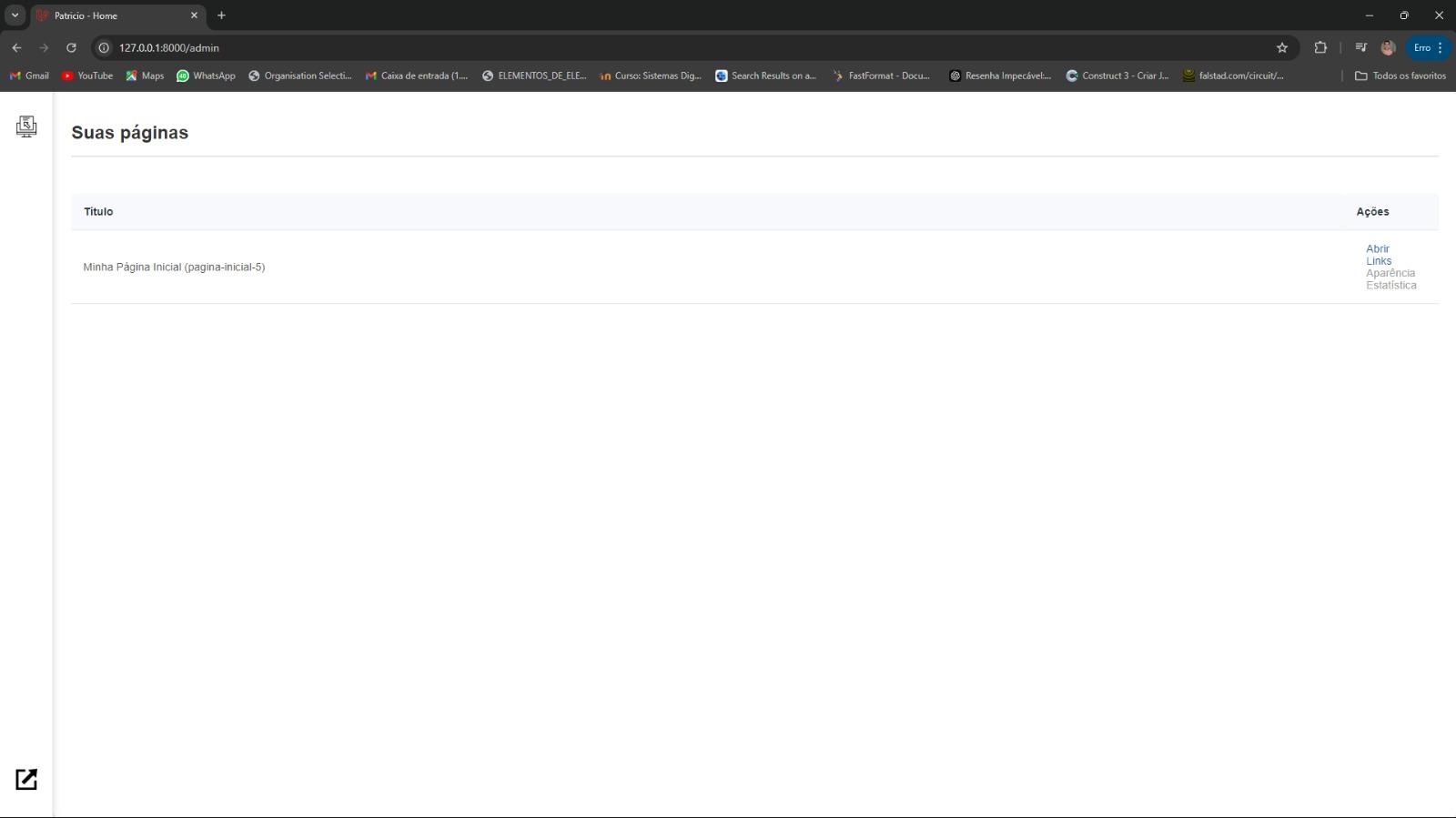


# 7. Estrutura da Página e Funcionalidades Principais:

* **Página Principal do Usuário**
  + Ao fazer login, o usuário é redirecionado para o painel principal (/admin).
  + Aqui, ele pode visualizar informações principais, incluindo suas páginas associadas, com destaque para o layout organizado e limpo.
* **Gerenciamento de Links**
  + A página inclui a opção para o usuário visualizar, adicionar, editar e excluir links associados a uma página específica, acessada através do slug da página.
  + Funções como pageLinks listam todos os links para a página correspondente do usuário, enquanto newLink e editLink permitem que ele adicione ou modifique links específicos.
* **Botões de Navegação e Ações**
  + **Logout**: Um botão de logout permite que o usuário encerre a sessão e seja redirecionado para a tela de login.
  + **Botões Aparência e Estatísticas**: Embora desabilitados, esses botões são exibidos em tom cinza claro para indicar futuras funcionalidades. Eles já têm a estrutura de lógica implementada, mas as interfaces associadas a eles ainda não foram totalmente desenvolvidas.
  + **Aparência**: Planejado para permitir que o usuário customize o design de suas páginas (cores, estilos de fundo, entre outros).
  + **Estatísticas**: Dará ao usuário a capacidade de monitorar e analisar dados de visualização de suas páginas, possibilitando uma gestão mais estratégica.

### **7.1 Organização Visual e Usabilidade**

* A página foi desenhada para oferecer uma interface organizada e funcional, seguindo o padrão visual de um painel administrativo. O layout também apresenta uma boa hierarquia visual, com botões intuitivos e uma área de navegação que guia o usuário nas principais ações de forma simples e eficaz.
* A escolha por exibir botões desabilitados sugere ao usuário as atualizações que estarão disponíveis em breve, indicando que o sistema é expansível e moderno.





### **7.2 Documentação das Funcionalidades**

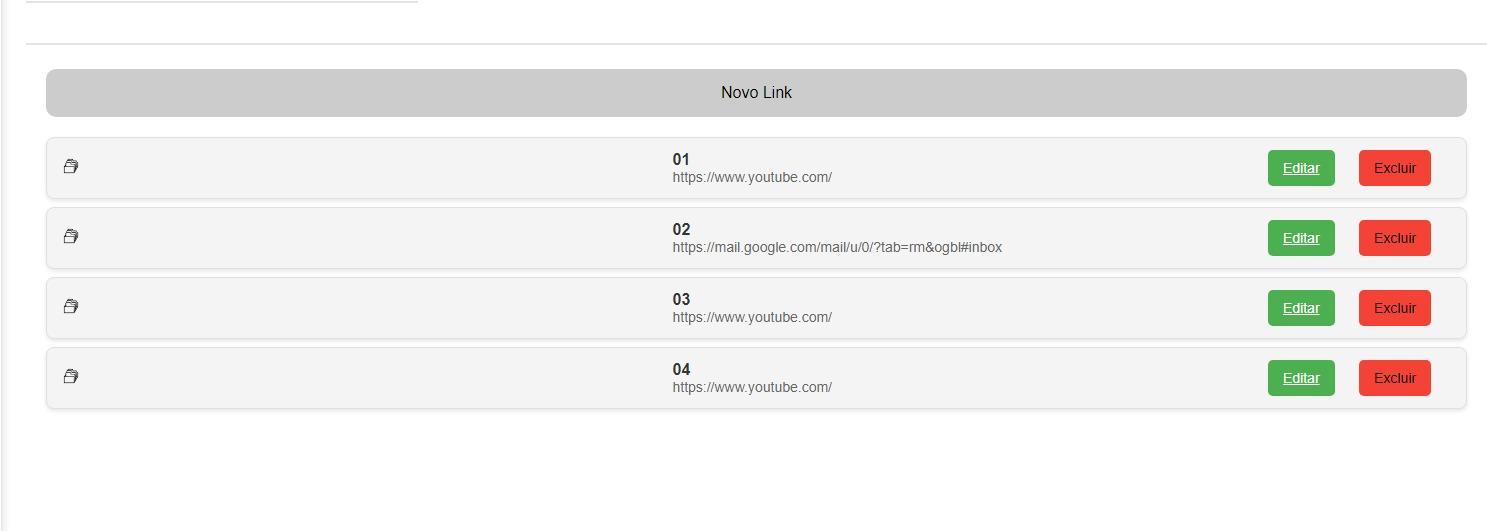
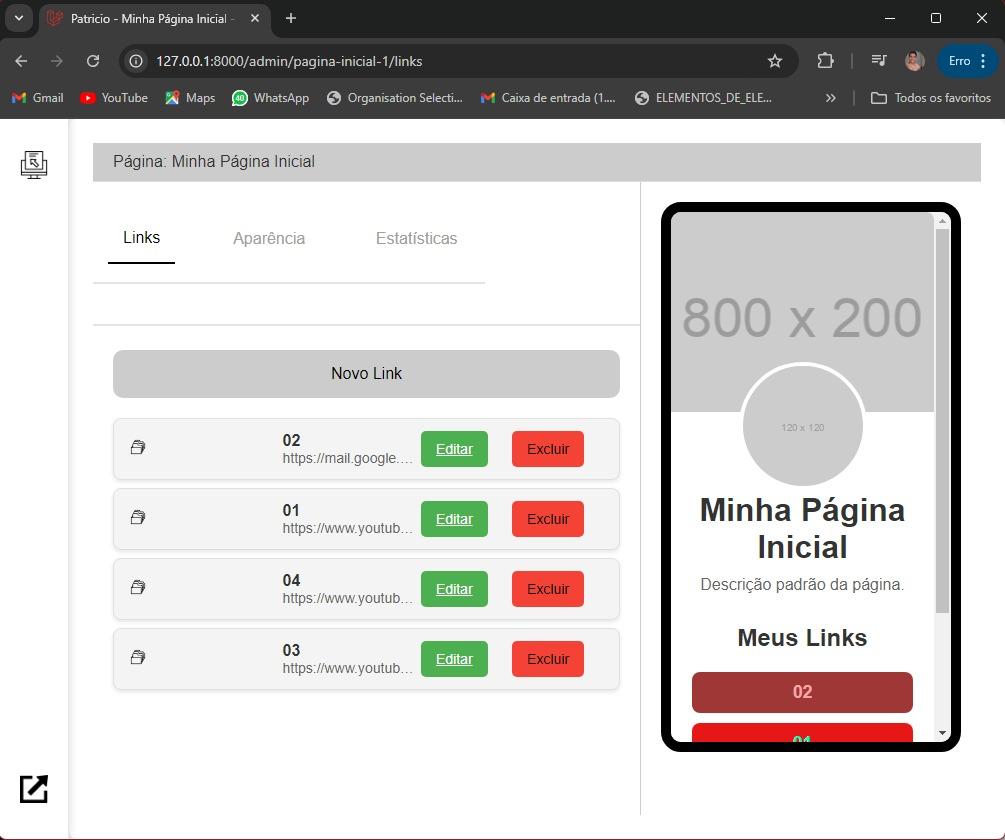
* **Botão "Novo Link"**:
  + Localizado no topo da lista, este botão permite criar um novo link para a página do usuário. Ao ser clicado, direciona para a página de criação de um link novo.
* **Lista de Links**:
  + Cada link é representado com as seguintes informações:
    - **Título** e **URL** do link.
    - Ícone para arrastar e reordenar.
    - Botões de ação:
      * **Editar**: Abre o formulário para edição completa do link selecionado.
      * **Excluir**: Remove o link com confirmação de exclusão.
* **Reordenamento de Links**:
  + A funcionalidade de arrastar e soltar (*drag-and-drop*) é utilizada para alterar a ordem dos links diretamente na interface.
  + **Persistência da Ordem**: Embora a biblioteca atualize visualmente a ordem, a aplicação garante que a nova ordem seja refletida no banco de dados através de uma requisição assíncrona (AJAX). A rota /admin/linkorder recebe o novo índice de cada link e ajusta as posições no banco.
* **Validação e Segurança**:
  + Cada alteração de ordem é protegida com um *token* CSRF para garantir a segurança da requisição.
  + Ao término do reordenamento, a página é recarregada para refletir a nova ordem visual.

### **7.3 Script de Ordenação com *Sortable.js***

O script Sortable fornece um efeito de animação ao arrastar os itens, facilitando a interação. A função onEnd garante que:

* Cada alteração de ordem envia uma solicitação para atualizar a posição do link no servidor.
* Em caso de sucesso, a interface é recarregada para confirmar visualmente a alteração.

Esse sistema foi desenvolvido para garantir que a ordem dos links seja mantida conforme estabelecida pelo usuário, evitando discrepâncias entre o que é exibido e o que está salvo no banco de dados.



# 8. Métodos de Edição de Links no AdminController

A parte de edição de links no AdminController permite ao usuário alterar as configurações dos links associados a uma página específica, incluindo cor de fundo, cor do texto, tipo de borda, entre outros. Essa funcionalidade utiliza métodos HTTP GET e POST para exibir e processar o formulário de edição dos links.

### **8.1 Métodos de Edição de Links**

### **8.2 editLink($slug, $linkid)**

Exibe o formulário de edição para um link específico associado a uma página.

* **Parâmetros**:
  + $slug: Identificador único da página.
  + $linkid: ID do link a ser editado.
* **Lógica**:
  + Verifica se a página e o link associados existem para o usuário autenticado.
  + Exibe a view admin/page\_editlink, passando os dados da página e do link.

### **8.3 editLinkAction($slug, $linkid, Request $request)**

Processa o formulário de edição de um link.

* **Parâmetros**:
  + $slug: Slug da página.
  + $linkid: ID do link a ser editado.
  + $request: Contém os dados submetidos do formulário.
* **Validações**:
  + status: Booleano que indica se o link está ativo ou inativo.
  + title: String, título do link com no mínimo 2 caracteres.
  + href: URL válida para o link.
  + op\_bg\_color: Cor de fundo no formato hexadecimal (#RRGGBB ou #RGB).
  + op\_text\_color: Cor do texto no formato hexadecimal.
  + op\_border\_type: Tipo de borda, aceitando square (quadrada) ou rounded (arredondada).
  + label: Campo opcional para uma etiqueta de texto.
* **Lógica**:
  + Busca o link para edição e aplica os campos validados.
  + Salva as alterações no link.
  + Redireciona o usuário para a página de links.

### **8.4. Exemplos de Configuração de Cores**

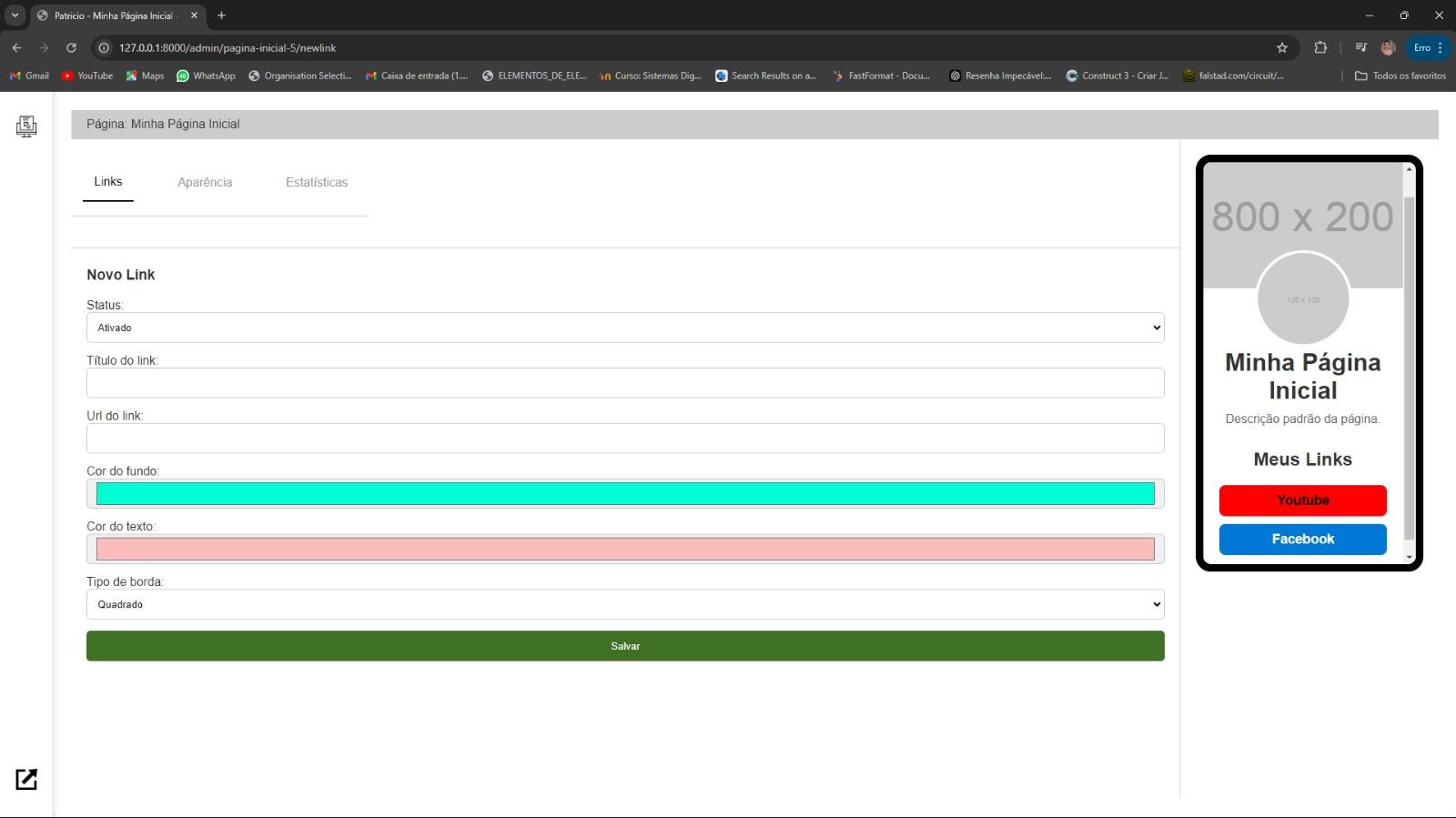
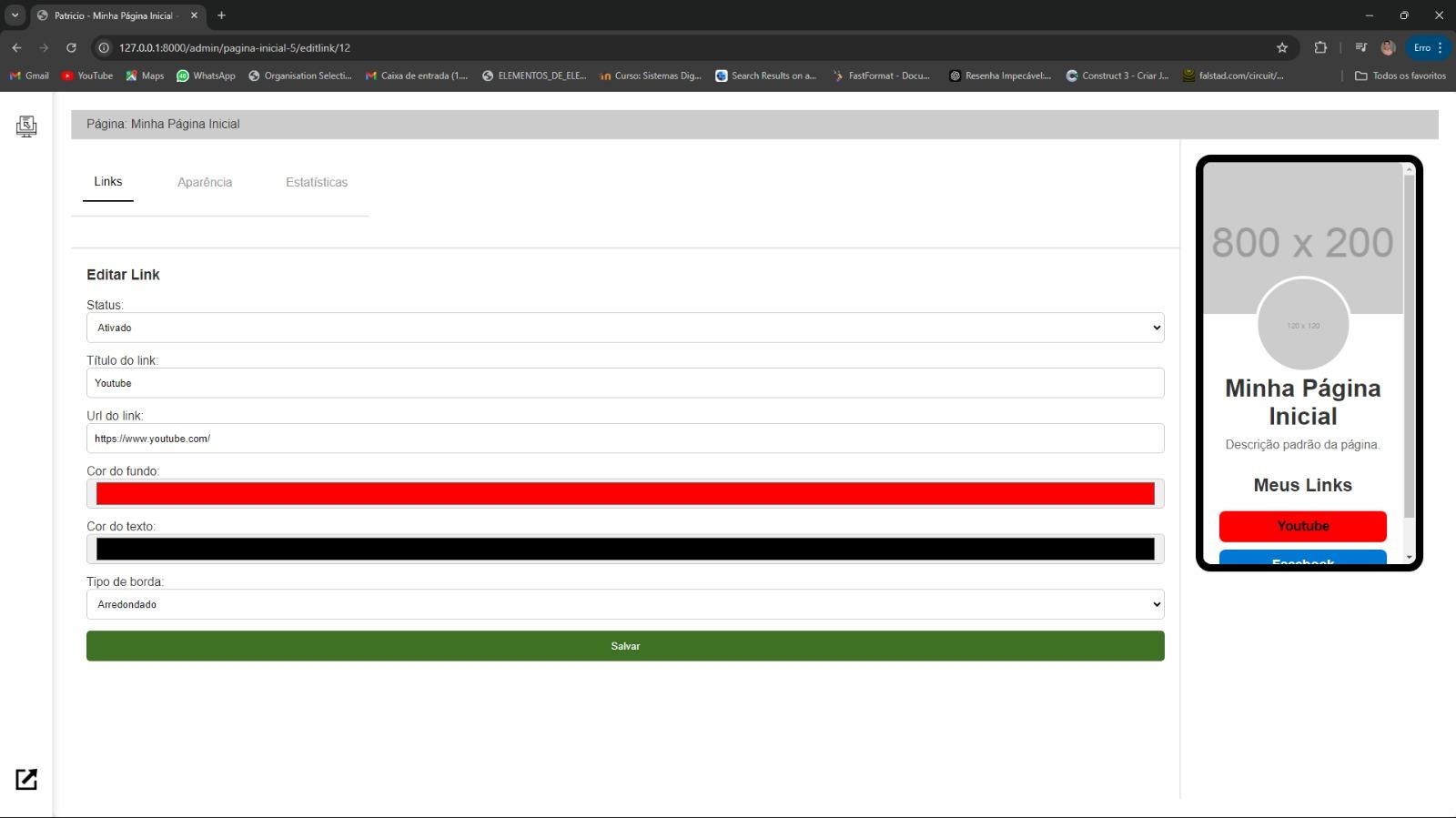
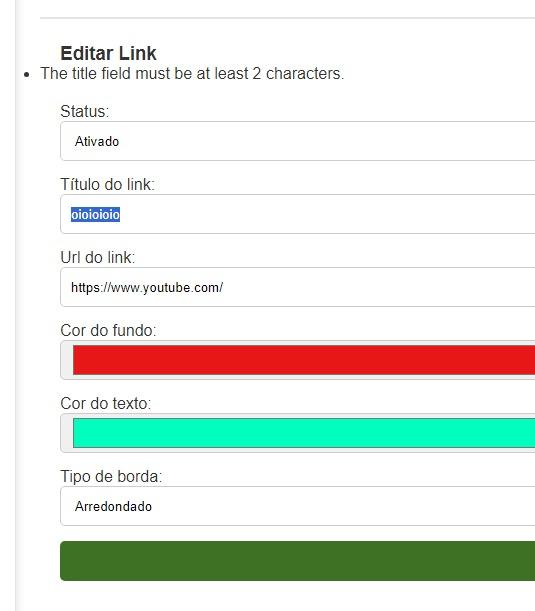
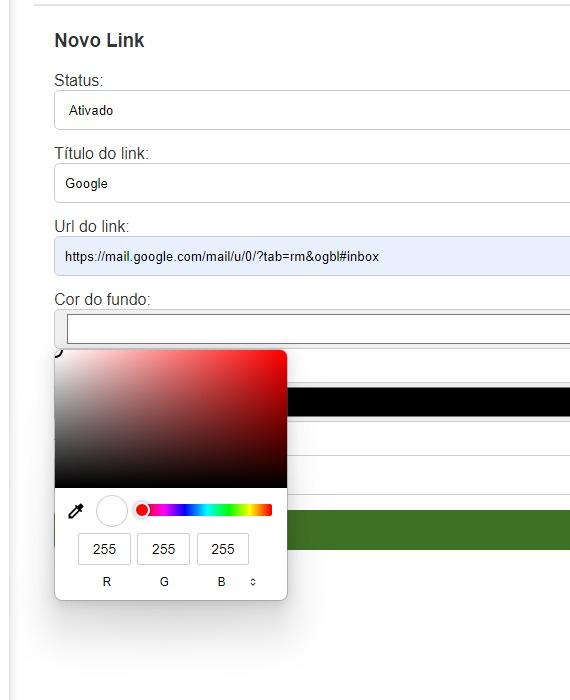
Para alterar a cor de fundo e a cor do texto de um link, utilize valores hexadecimais, como mostrado nos exemplos abaixo:

* **Cor de Fundo (op\_bg\_color)**: #FFFFFF (branco) ou #000000 (preto)
* **Cor do Texto (op\_text\_color)**: #FF5733 (vermelho alaranjado) ou #333333 (cinza escuro)

**Nota**: A validação hex é aplicada para assegurar que somente valores válidos de cor hexadecimal sejam aceitos.

### **8.5. Campos Disponíveis para Edição**

* **Título (title)**: Nome exibido para o link.
* **URL (href)**: Endereço para o qual o link redireciona.
* **Cor de Fundo (op\_bg\_color)**: Configuração visual do link.
* **Cor do Texto (op\_text\_color)**: Ajuste da cor da fonte do link.
* **Tipo de Borda (op\_border\_type)**: Formato da borda; aceita valores square ou rounded.

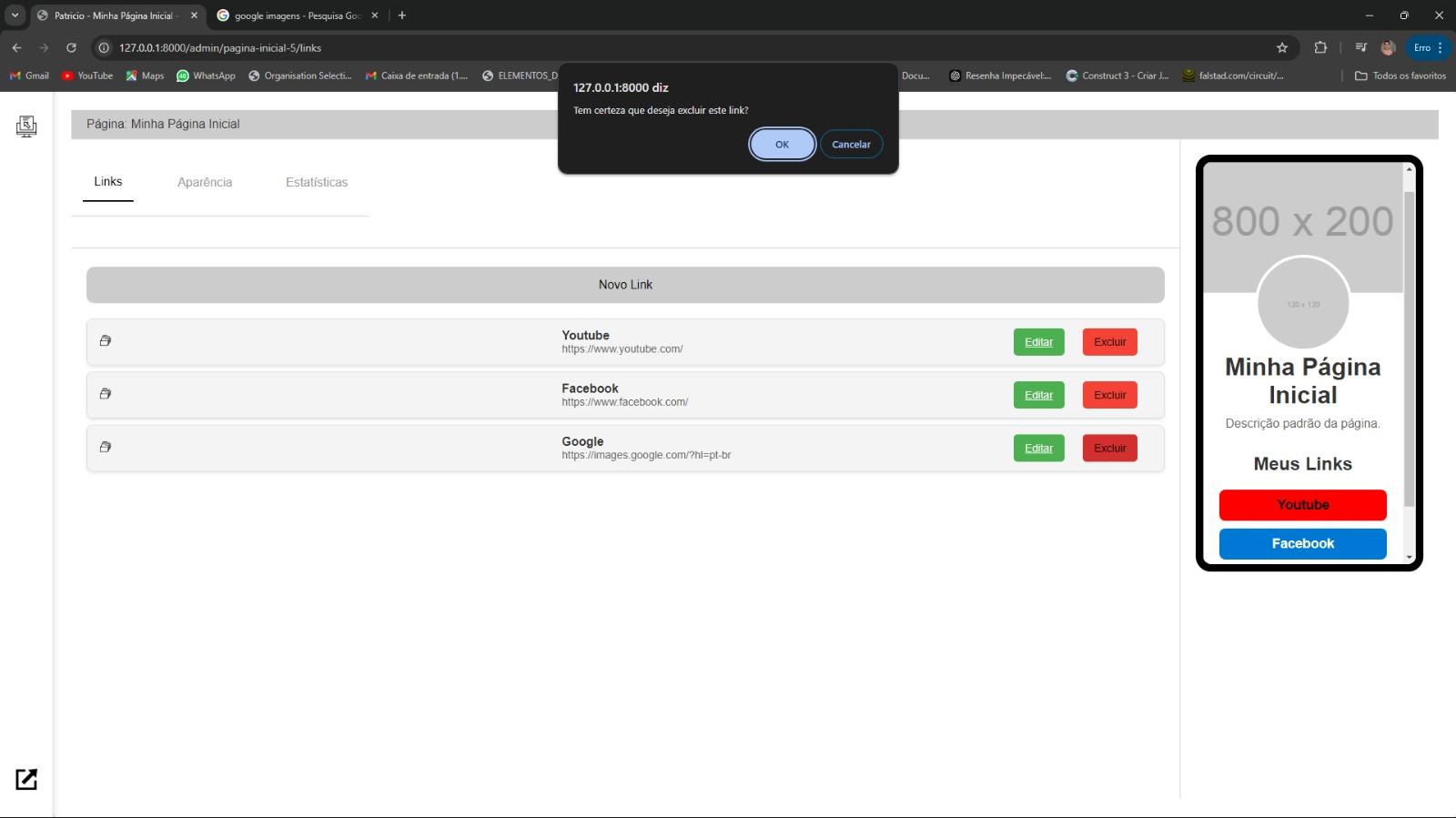


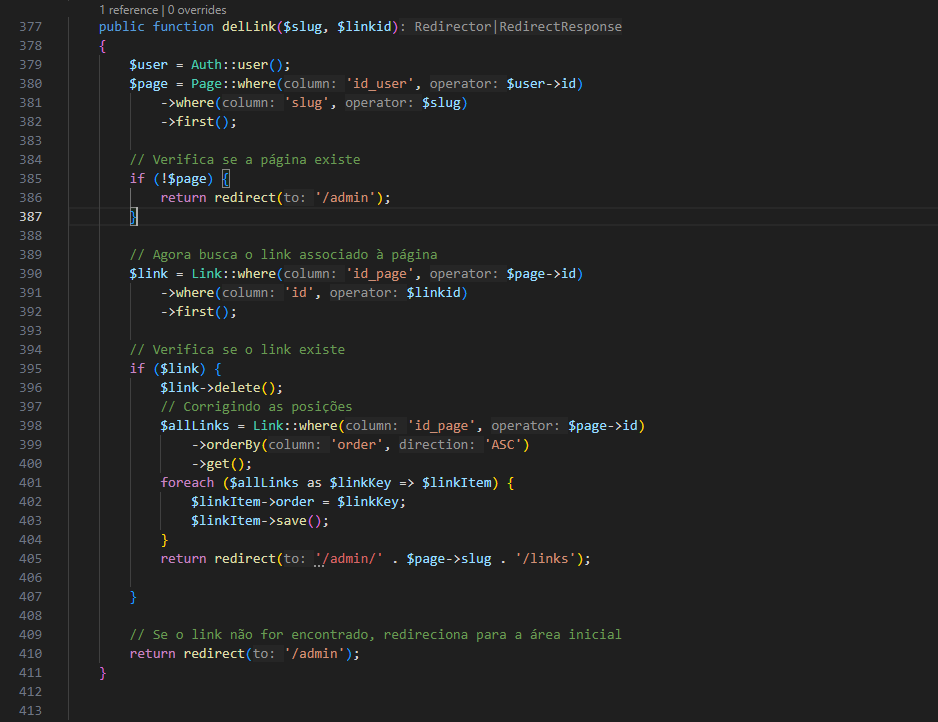
# 9. Método delLink - Funcionalidade Excluir Link

O método delLink do AdminController é responsável por excluir um link específico associado a uma página do usuário autenticado. Essa exclusão é feita diretamente no banco de dados, removendo permanentemente o link. Além de apagar o link, o método reordena os links restantes para manter uma sequência de exibição correta.

#### **Funcionamento**

* **Busca da Página**: Através do slug, o método localiza a página associada ao usuário autenticado.
* **Verificação do Link**: Verifica a existência do link específico (linkid) relacionado a essa página.
* **Exclusão do Link**: Se o link existir, ele é deletado do banco de dados.
* **Reordenação dos Links Restantes**: Após a exclusão, o método ajusta a ordem dos links restantes, garantindo que a sequência continue consistente. Ele reatribui a posição (order) de cada link na página para evitar buracos na sequência de exibição.





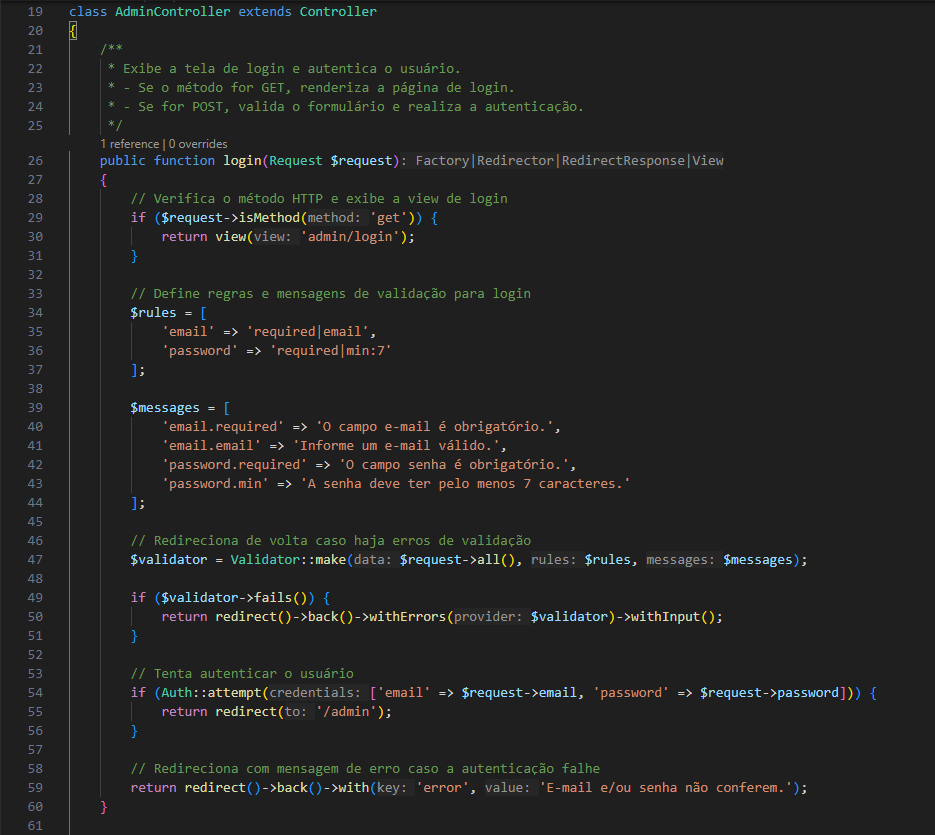
# 10. Controladores

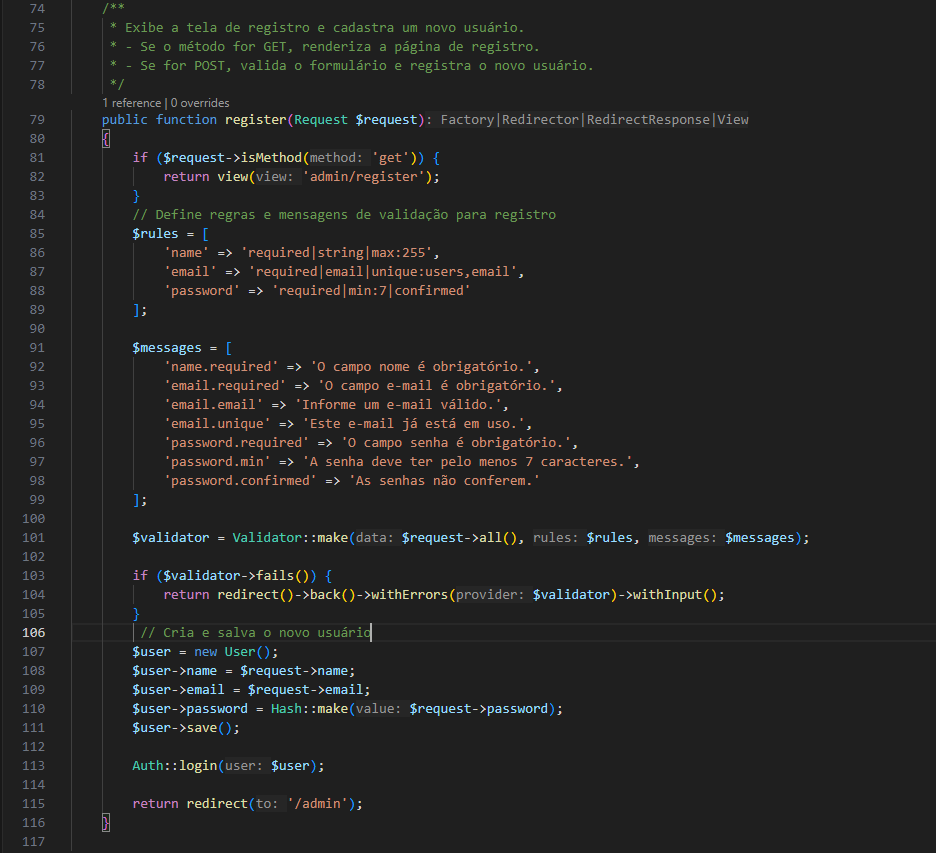
### **10.1 AdminController**

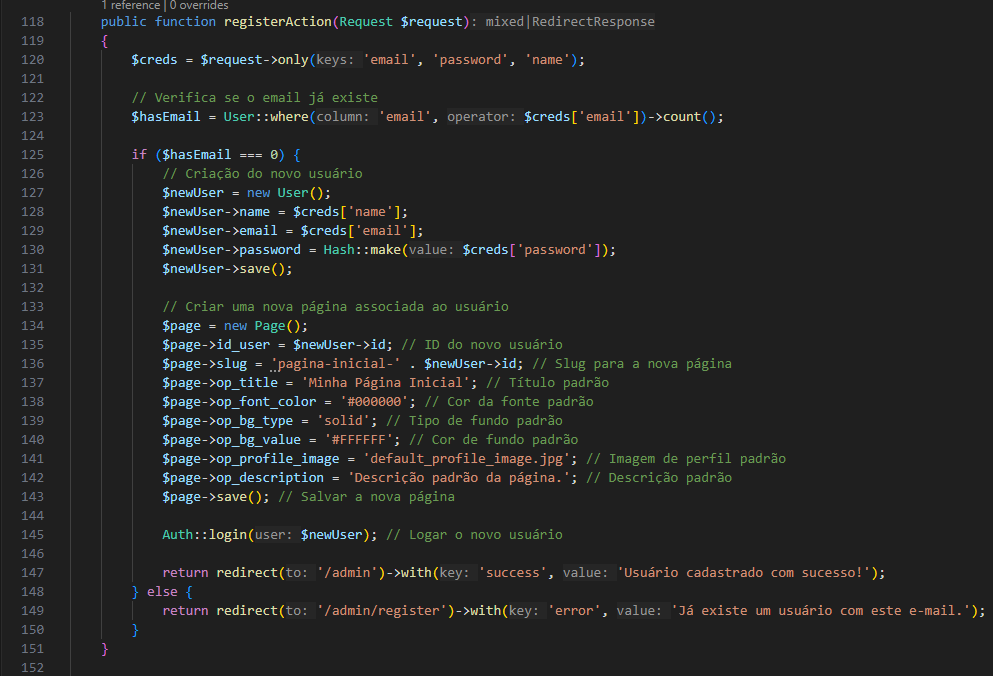
O AdminController é o controlador responsável por gerenciar as principais funcionalidades administrativas da aplicação. Ele engloba operações relacionadas à autenticação de usuários, manipulação de links personalizados e gestão de páginas associadas aos usuários registrados. Suas funções incluem:

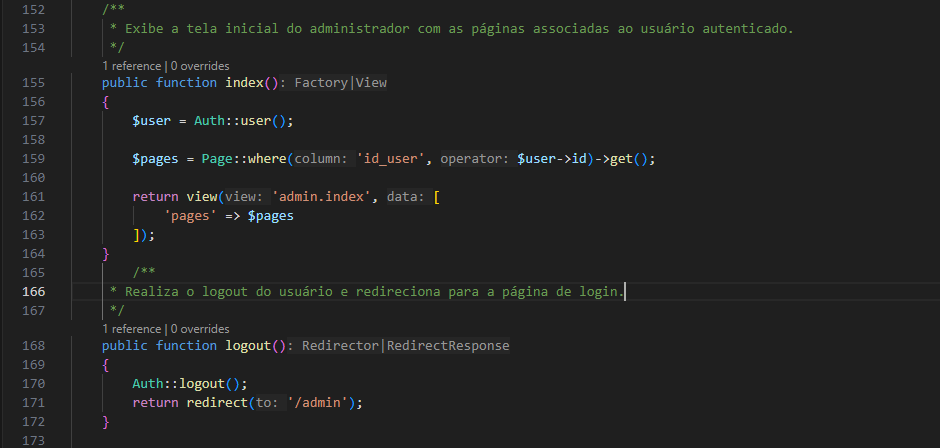
* **Autenticação de Usuários**: realiza o login e logout dos administradores, além de exibir a interface de login e registro.
* **Gerenciamento de Usuários**: valida e cadastra novos usuários, incluindo a criação de uma página inicial padrão para cada usuário recém-registrado.
* **Gestão de Páginas e Links**: exibe páginas e links associados aos usuários, permite adicionar, editar, reordenar e excluir links personalizados, utilizando uma biblioteca de drag-and-drop para reordenar visualmente e salvar no banco de dados.
* **Configurações de Design e Estatísticas**: inclui as interfaces para personalização de design da página e visualização de estatísticas (botões atualmente desativados e indicados como funcionalidades futuras).

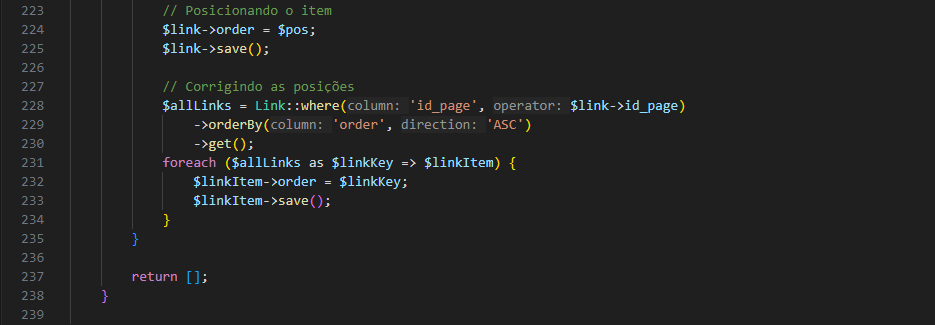
O AdminController assim organiza e facilita a administração de conteúdos e interações, oferecendo um ponto central de controle e navegação para usuários autenticados.

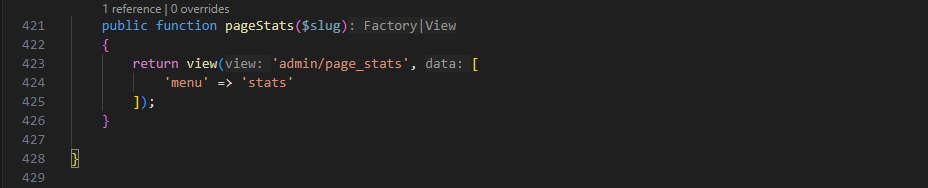
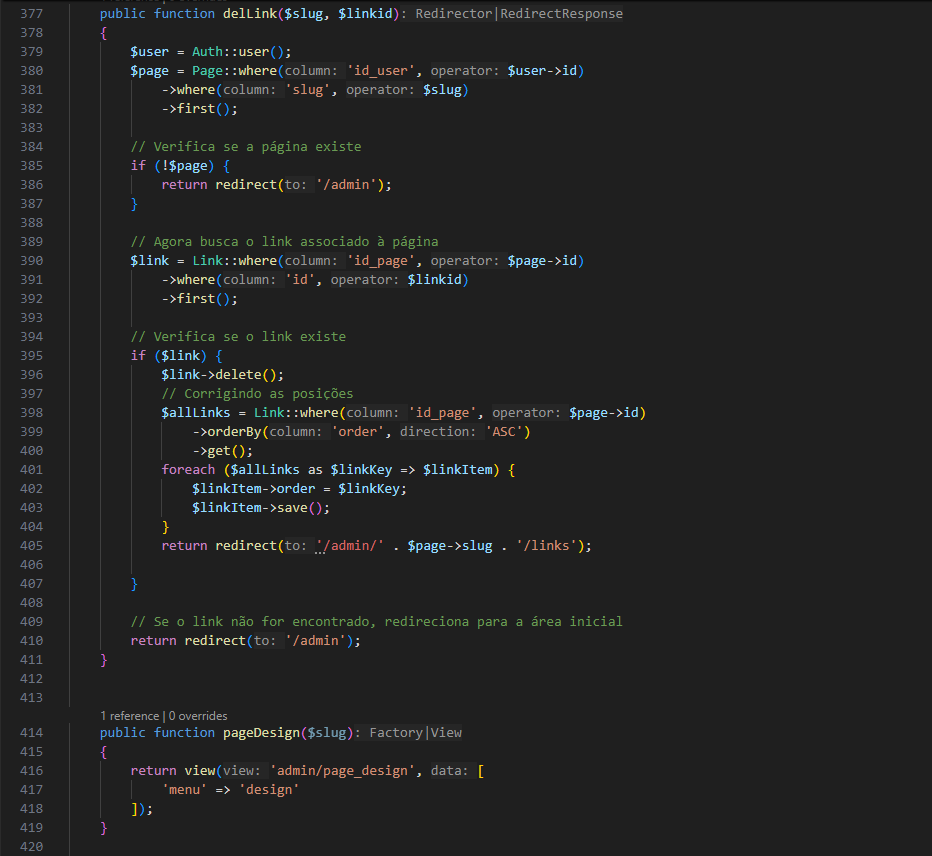
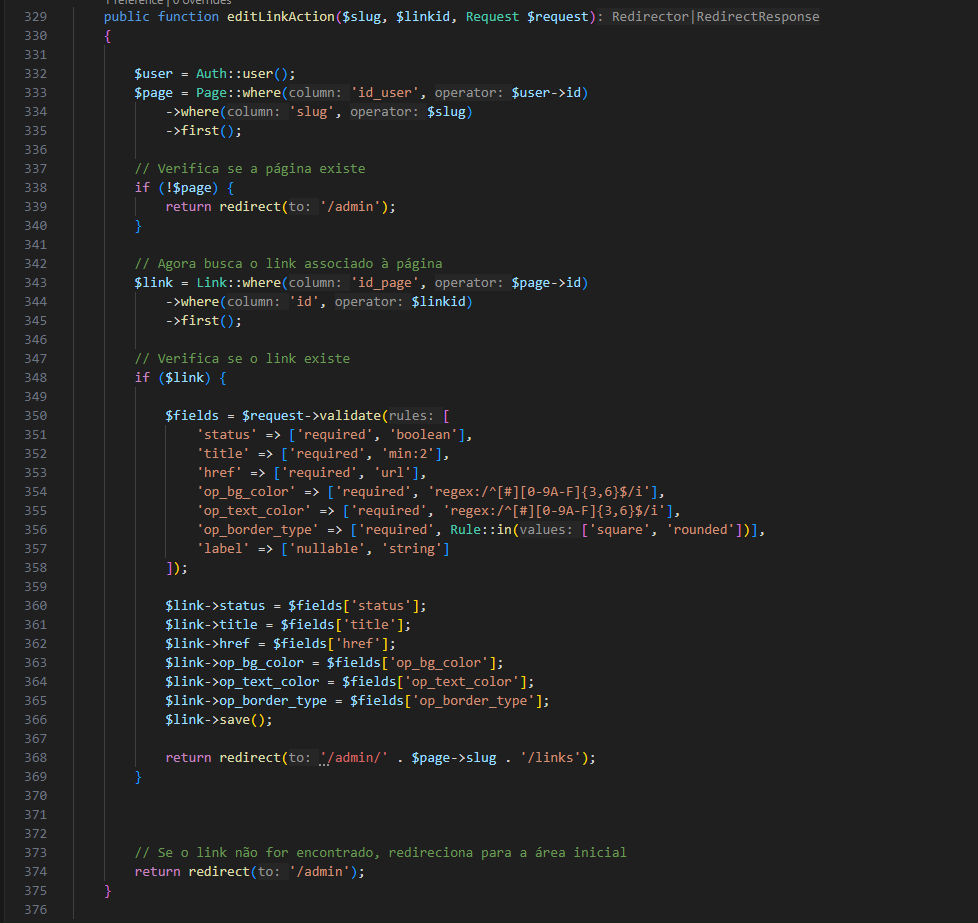
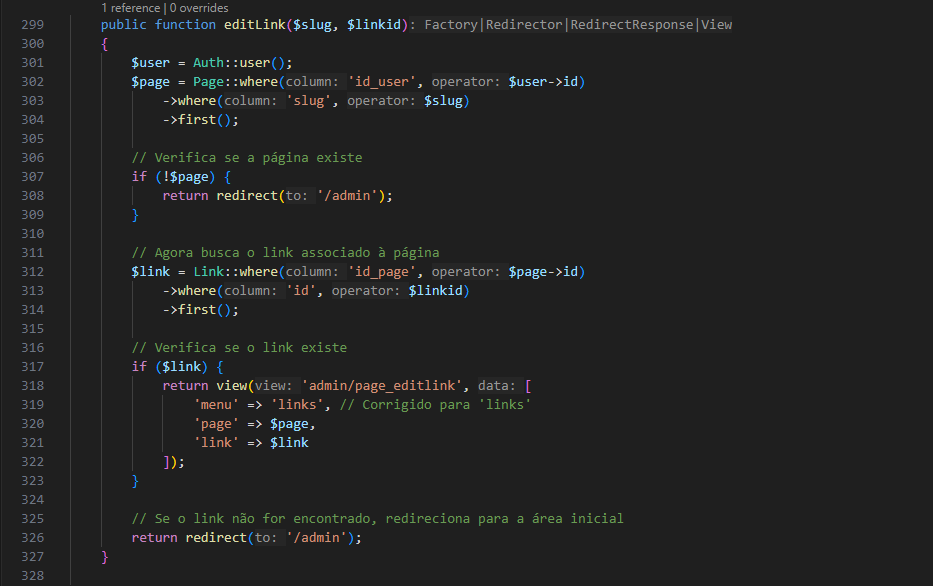
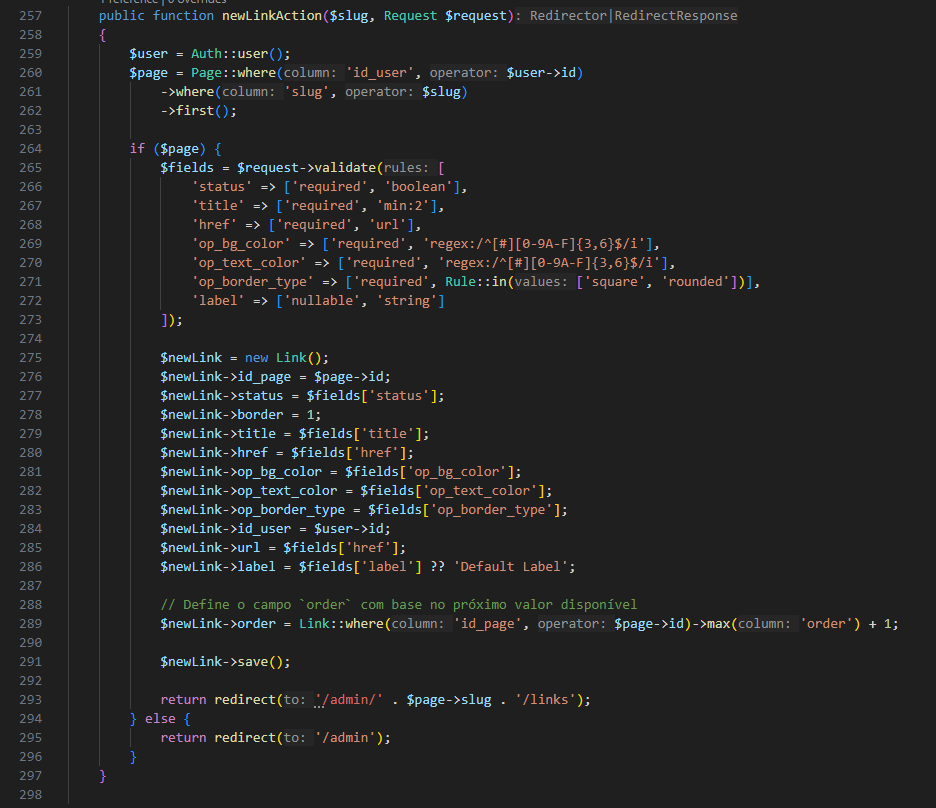












### **10.2 PageController**

O PageController é responsável por gerenciar a exibição e criação de páginas de perfil no sistema, permitindo que cada usuário visualize e configure suas páginas personalizadas. Abaixo estão os métodos e funcionalidades principais desse controlador.

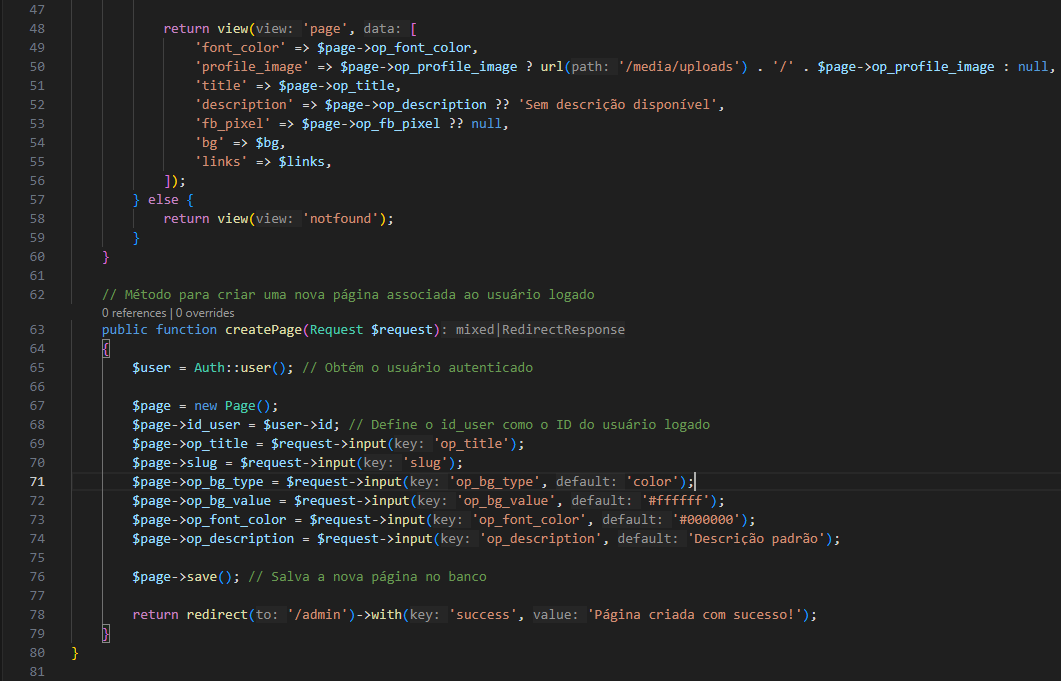
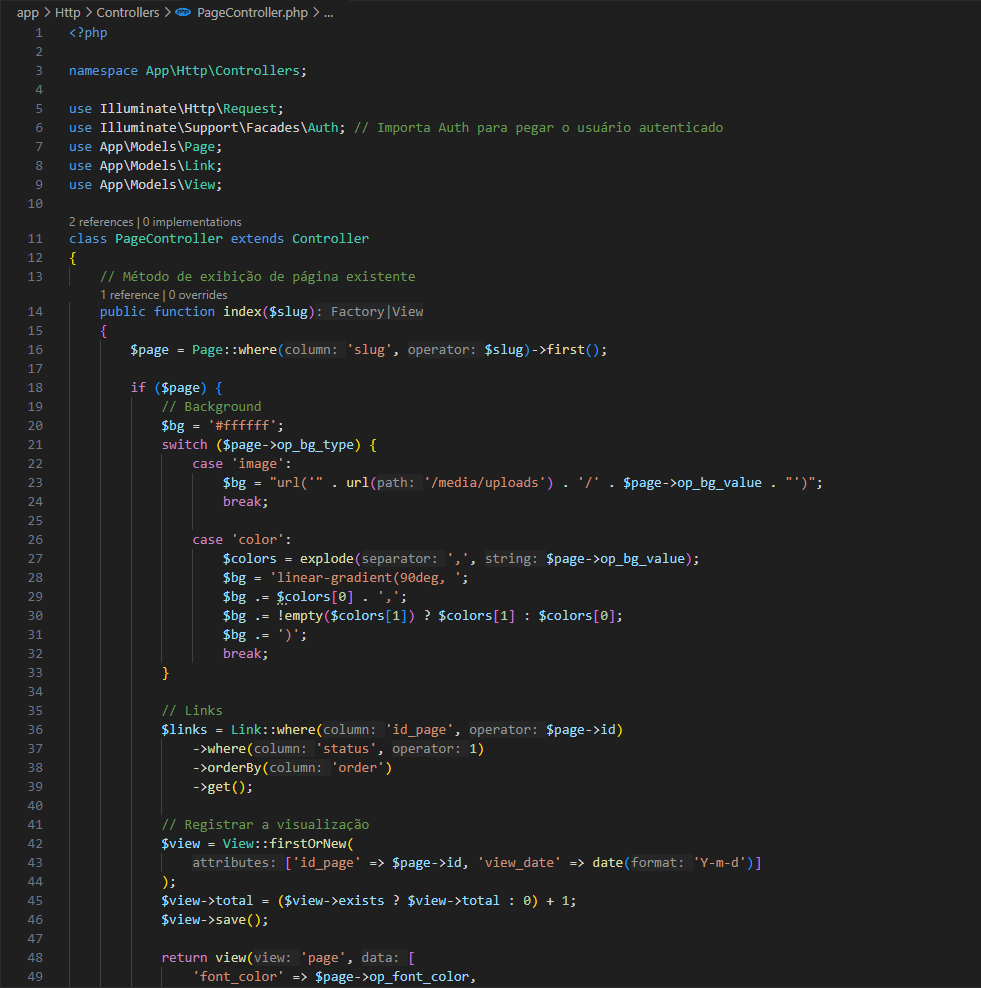
### **Método index**

* **Descrição**: Exibe uma página existente com base no slug fornecido na URL.
* **Funcionamento**:
  + Primeiro, busca uma página no banco de dados com o slug especificado.
  + Se a página for encontrada, define o **background** da página com base nas configurações do usuário:
    - Imagem: utiliza uma URL para exibir a imagem de fundo.
    - Cor: define uma cor ou gradiente linear.
  + Recupera e exibe os links ativos associados a essa página, ordenados pela posição configurada.
  + Registra uma visualização no banco de dados para a página exibida, armazenando a data e incrementando o contador.
* **Retorno**: Renderiza a página de visualização com as informações configuradas ou exibe uma página "não encontrada" se o slug não existir.
* **Variáveis Disponíveis na View**:
  + font\_color, profile\_image, title, description, fb\_pixel, bg, e links.

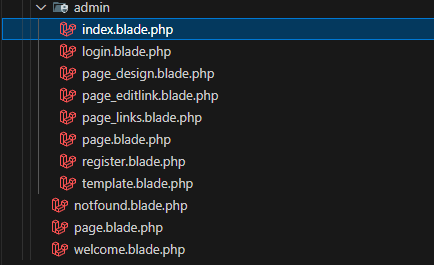
### **Método createPage**

* **Descrição**: Cria uma nova página personalizada associada ao usuário autenticado.
* **Funcionamento**:
  + Obtém o usuário atualmente autenticado no sistema.
  + Cria um novo registro na tabela Page e associa o id\_user ao ID do usuário logado.
  + Preenche os campos da página com valores enviados pelo usuário e aplica padrões onde necessário (como o tipo de fundo ou cor).
  + Salva a nova página no banco de dados.
* **Retorno**: Redireciona para a área administrativa com uma mensagem de sucesso.

Esses métodos juntos proporcionam ao usuário a funcionalidade de criar e visualizar páginas personalizadas, com controle de design e links personalizados para sua exibição.



# 11.Modelos Blade e Estrutura Visual



### **11.1Index.blade.php**

### **11.2 Visão Geral**

A página index.blade.php é uma página de listagem das páginas gerenciáveis pelo usuário no sistema de administração. Ela utiliza o template admin.template.

Dentro da seção content, a página apresenta um cabeçalho com o título "Suas páginas" e uma tabela listando as páginas do usuário. Para cada página, exibe o título e uma série de links para acessar e gerenciar a página.

* + **Cabeçalho da Tabela**
    - Coluna **Título**: Exibe o título da página.
    - Coluna **Ações**: Fornece uma largura fixa de 20px para exibir as ações disponíveis para cada página.
  + **Corpo da Tabela**
    - Cada linha da tabela representa uma página e contém:
      * **Título da Página**: Nome da página com o título $page->op\_title, seguido pelo slug.
      * **Ações**:
        + **Abrir**: Link para abrir a página em uma nova aba.
        + **Links**: Link para gerenciar os links associados à página.
        + **Aparência**: Link desativado (classe "disabled") para acessar o design da página.
        + **Estatística**: Link desativado para acessar as estatísticas da página.

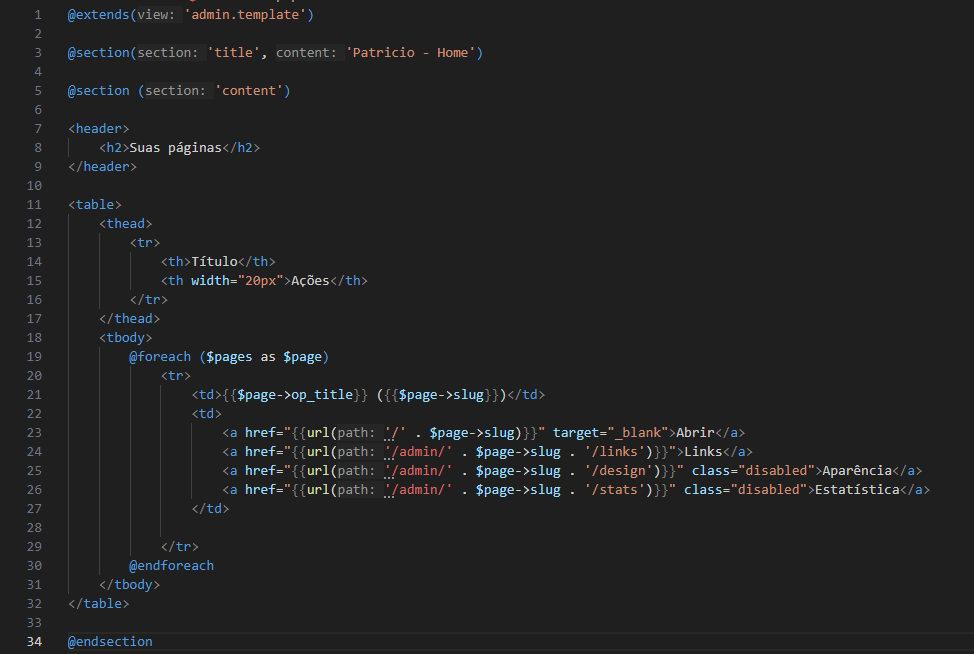
#### **Observações Importantes**

* **Links Desativados**: Os links "Aparência" e "Estatística" têm a classe disabled, indicando que essas funcionalidades podem estar indisponíveis.
* **Iteração de Dados**: A exibição das páginas é feita usando o loop @foreach, que percorre a coleção $pages.
* **URLs Dinâmicas**: Os URLs são construídos dinamicamente com o uso de {{ url() }}, com base no slug de cada página para permitir acesso e gerenciamento específicos.

#### **Dependências**

Para o correto funcionamento dessa página, é necessário:

* Um layout admin.template.
* A coleção $pages, que deve ser passada do controlador para a view, contendo o título (op\_title) e slug de cada página.



### **11.3 page\_editlink.blade.php**

### **page\_editlink.blade.php**

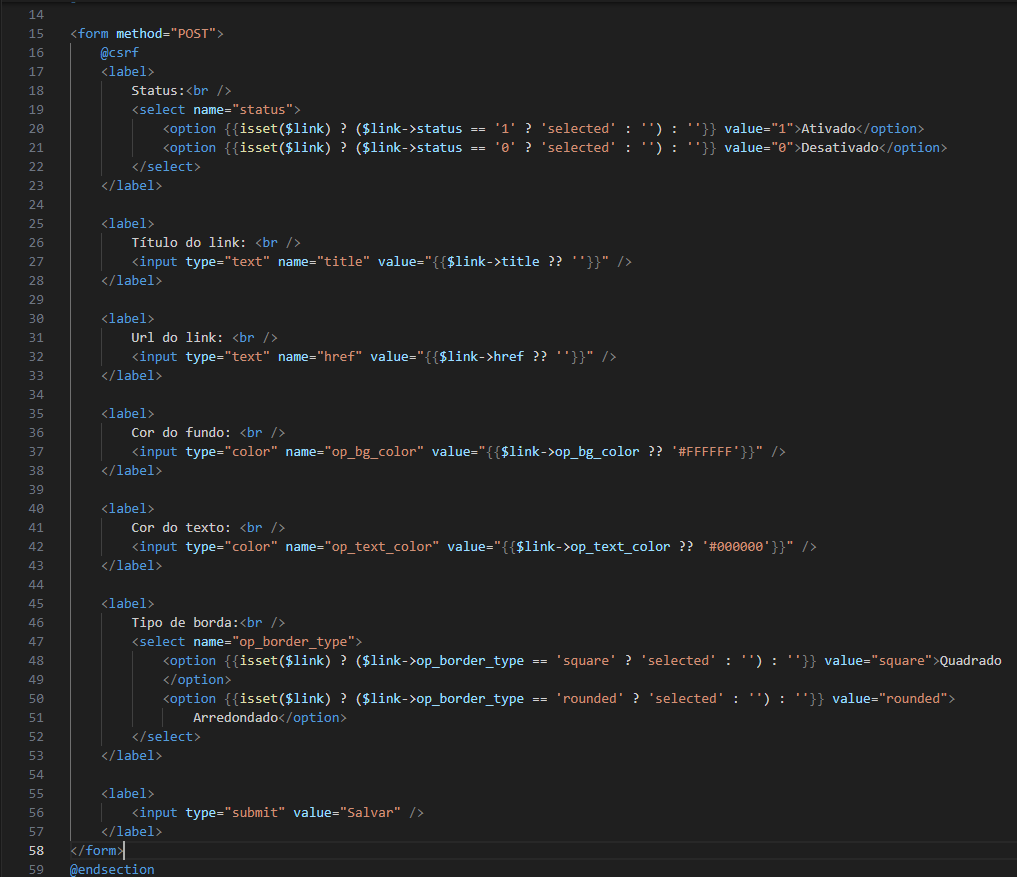
#### **Visão Geral**

Página para criação e edição de links no painel administrativo, utilizando o template admin.page. Exibe um formulário para configurar atributos do link, como status, título, URL, cores e tipo de borda.

* **Cabeçalho**: Mostra "Editar Link" se $link estiver definido; caso contrário, "Novo Link".
* **Exibição de Erros**: Exibe uma lista de erros de validação, se houver.
* **Formulário**:
  + **Status**: Define se o link está ativado ou desativado.
  + **Título**: Título do link.
  + **URL**: Endereço do link.
  + **Cor do Fundo/Text**: Escolha de cores.
  + **Tipo de Borda**: Opções de borda quadrada ou arredondada.

Formulário protegido por CSRF (@csrf) para segurança.

Esta página centraliza a criação e edição dos links com campos básicos de configuração.



### **11.4 page\_links.blade.php**

### **page\_links.blade.php**

#### **Visão Geral**

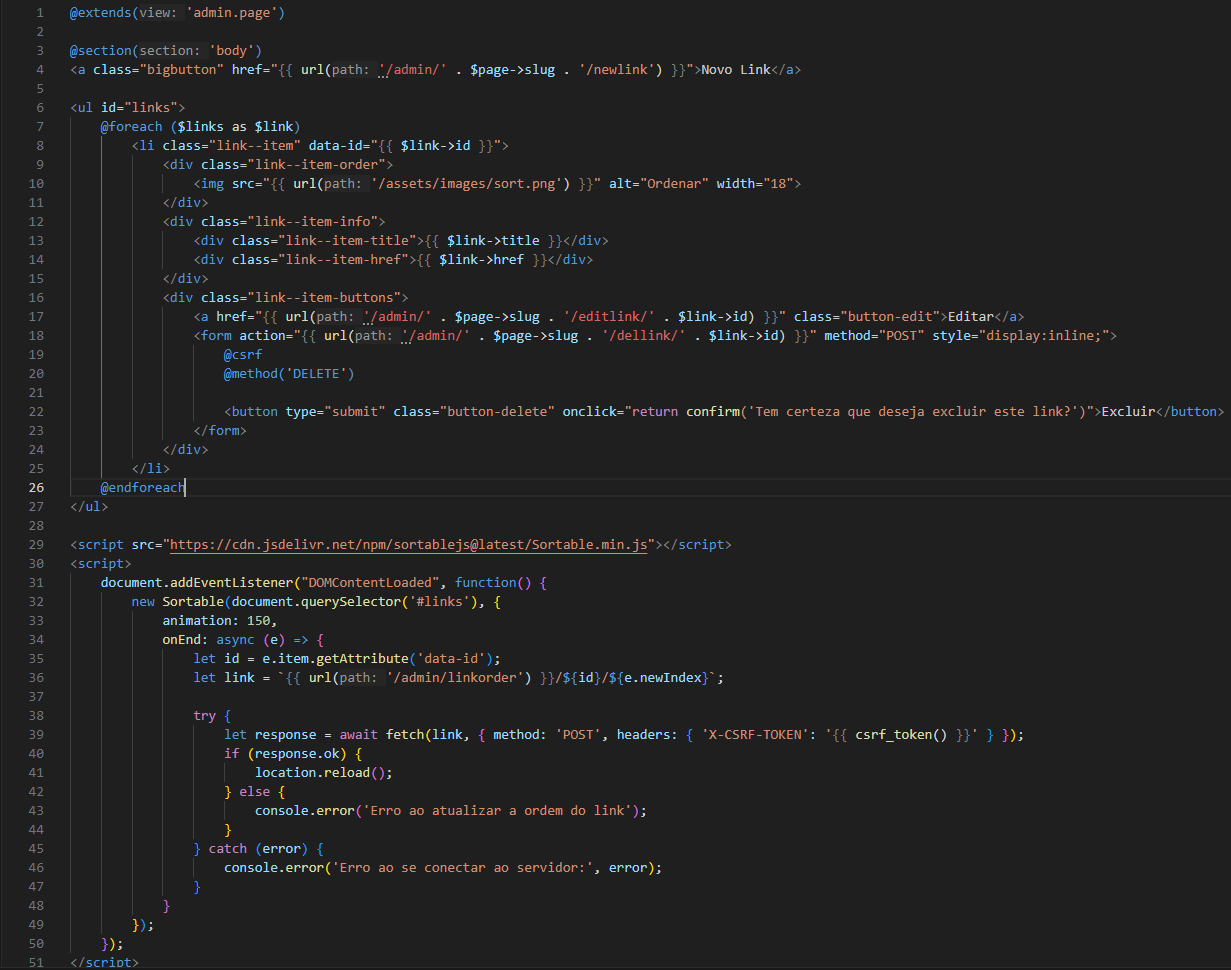
Página administrativa para listar, criar, editar, excluir e reordenar links associados a uma página específica.

#### **Estrutura**

**Template**php  
  
@extends('admin.page')

* Usa o layout admin.page para consistência visual no painel.
* **Botão "Novo Link"** Botão para adicionar um novo link, direcionando para a página de criação.
* **Listagem de Links** Cada link é listado em um item (li) com:
  + Ícone para arrastar e reordenar.
  + Informações do link (título e URL).
  + Botões de ação:
    - **Editar**: Abre a página de edição do link.
    - **Excluir**: Remove o link com confirmação.
* **Reordenação Dinâmica** Utiliza **Sortable.js** para permitir reordenação por "arrastar e soltar", enviando a nova ordem ao servidor via JavaScript e AJAX.

Essa página oferece uma interface completa para gerenciamento e ordenação de links.



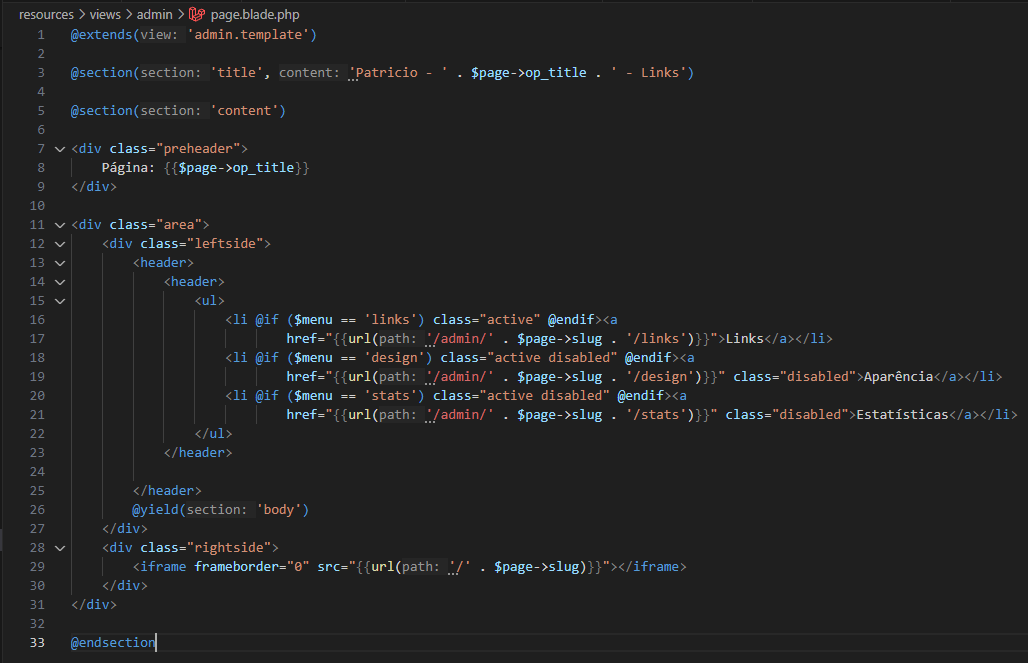
### **11.5 page.blade.php**

#### **Visão Geral**

O arquivo page\_links.blade.php é uma interface de administração no sistema, projetada para gerenciar links específicos de uma página. O layout da página oferece um painel com navegação lateral para acessar diferentes seções (Links, Aparência e Estatísticas) e uma área de visualização em tempo real da página selecionada.

### **11.6 Menu de Navegação Lateral**

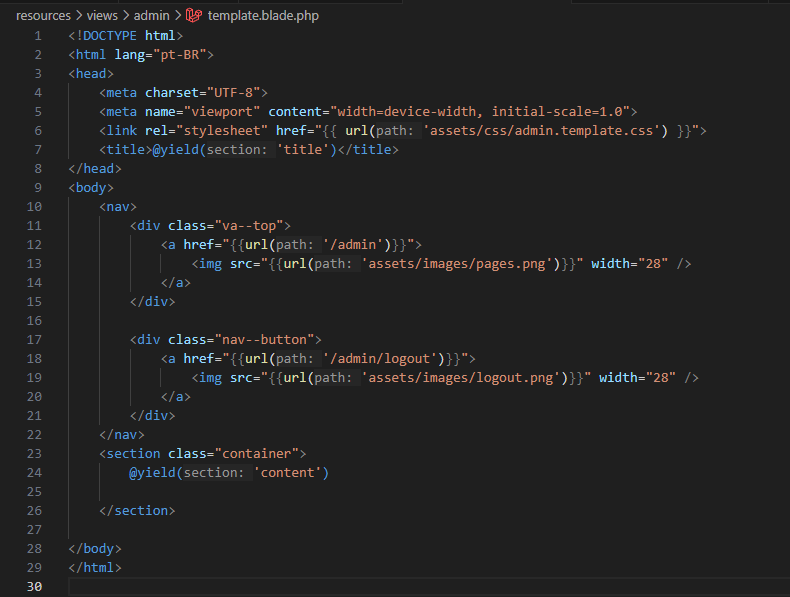
O menu de navegação lateral permite ao usuário alternar entre as seções de Links, Aparência e Estatísticas da página. Cada item de menu usa uma lógica de estado ativo (active) e desativado (disabled) para mostrar visualmente ao usuário a seção que está acessando e desabilitar as seções que não estão ativas.



### **11.7 template.blade.php**

#### **Visão Geral**

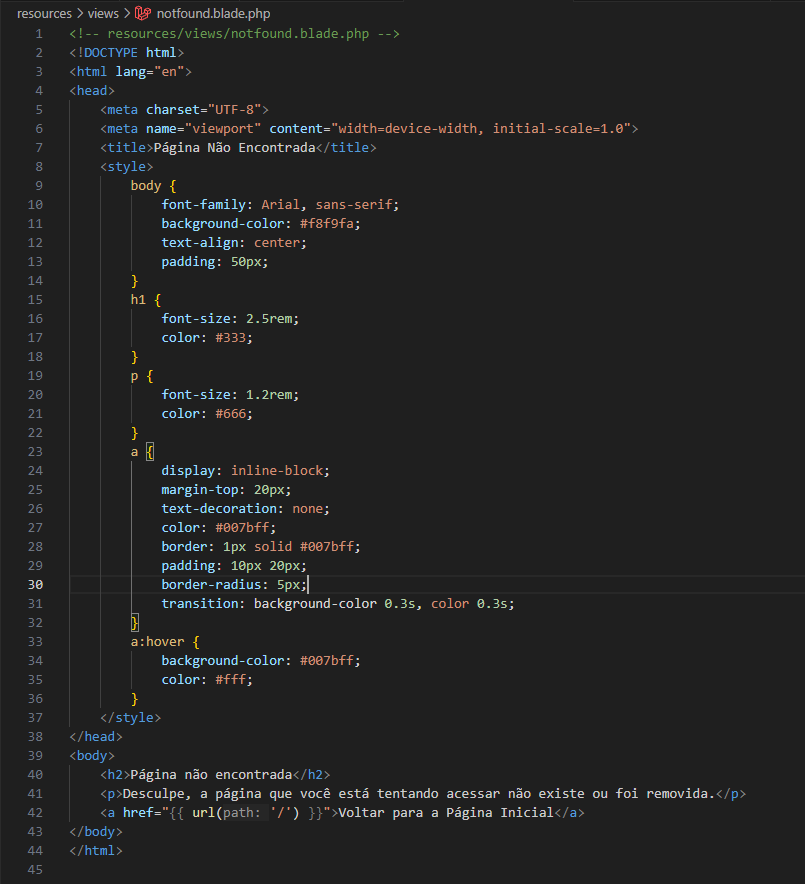
Este arquivo (template.blade.php) é o layout principal para as páginas do painel administrativo. Ele define a estrutura HTML básica e inclui uma navegação superior com links para a página inicial do painel e para logout. Além disso, permite a inclusão dinâmica de conteúdo e títulos de página, mantendo o layout consistente em todas as páginas do painel.



### **11.8 notfound.blade.php**

#### **Visão Geral**

O arquivo notfound.blade.php é uma página de erro personalizada que informa ao usuário que a página solicitada não foi encontrada (404). Esta página é projetada para ser simples e amigável, proporcionando ao usuário uma experiência clara e um link de retorno à página inicial.



# 12. Considerações Finais

Este projeto CRUD em Laravel foi desenvolvido com o intuito de oferecer uma plataforma simples e eficiente para o gerenciamento de links. A arquitetura em Laravel, juntamente com as boas práticas de segurança e a interface amigável, tornam esta aplicação robusta e acessível. O projeto também está preparado para futuras melhorias, como estatísticas e customizações de design.

# 13. Contato e Suporte

Para dúvidas, sugestões ou suporte técnico, entre em contato:  
**Matheus Patricio**Email: matheuspatricio.aspx@gmail.com  
LinkedIn: https://www.linkedin.com/in/matheus-patricio-ab77031b6/