# Referência da API do Phishing Manager

Este documento detalha as rotas da API RESTful do Phishing Manager, incluindo endpoints, métodos HTTP, parâmetros de requisição e exemplos de resposta. A API é a principal forma de comunicação entre o frontend e o backend, e também pode ser utilizada por outras aplicações ou scripts.

## 1. Convenções da API

- **Base URL**: Todas as rotas são prefixadas com `/api`.
- **Formato de Dados**: Todas as requisições e respostas utilizam o formato JSON.
- **Autenticação**: Rotas protegidas exigem um token de autenticação (JWT ou sessão) no cabeçalho `Authorization: Bearer <token>`.
- **Códigos de Status HTTP**: Os códigos de status HTTP são utilizados para indicar o resultado da operação (e.g., `200 OK`, `201 Created`, `400 Bad Request`, `401 Unauthorized`, `403 Forbidden`, `404 Not Found`, `500 Internal Server Error`).
- **Validação de Entrada**: Todos os dados de entrada são validados. Erros de validação resultam em respostas `400 Bad Request` com detalhes sobre os campos inválidos.

## 2. Rotas de Autenticação (`src/routes/auth.py`)

### 2.1. Registro de Usuário

`POST /api/register`

Registra um novo usuário no sistema.

### Parâmetros de Requisição (JSON Body)

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `username` | `string` | Sim | Nome de usuário único. | `novo_usuario` |
| `email` | `string` | Sim | Endereço de e-mail único. | `user@example.com` |
| `password` | `string` | Sim | Senha do usuário (deve ser forte). | `SenhaSegura123!` |

### Respostas

- **`201 Created`**: Usuário registrado com sucesso. `json { "message": "Usuário registrado com sucesso!" }`
- **`400 Bad Request`**: Erro de validação ou usuário/email já existe. `json { "error": "Nome de usuário ou e-mail já existe." }` ou `json { "errors": { "username": ["O nome de usuário deve ter entre 3 e 30 caracteres e conter apenas letras, números, hífens ou underscores."], "password": ["A senha deve ter pelo menos 8 caracteres, incluindo uma letra maiúscula, uma minúscula, um número e um caractere especial."] } }`

## 2.2. Login de Usuário

`POST /api/login`

Autentica um usuário e retorna um token de sessão.

### Parâmetros de Requisição (JSON Body)

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `username` | `string` | Sim | Nome de usuário. | `novo_usuario` |
| `password` | `string` | Sim | Senha do usuário. | `SenhaSegura123!` |

### Respostas

- **`200 OK`**: Login bem-sucedido. `json { "message": "Login bem-sucedido!", "user": { "id": 1, "username": "novo_usuario", "email":`

`"user@example.com", "is_admin": false, "credits": 100, "is_banned": false, "otp_enabled": false }, "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9..." }`

- **`401 Unauthorized`**: Credenciais inválidas. `json { "error": "Nome de usuário ou senha inválidos." }`

## 2.3. Logout de Usuário

`POST /api/logout`

Invalida a sessão do usuário atual. Requer autenticação.

### Parâmetros de Requisição

Nenhum.

### Respostas

- **`200 OK`**: Logout bem-sucedido. `json { "message": "Logout bem-sucedido!" }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

## 2.4. Verificação de Token

`GET /api/verify_token`

Verifica a validade do token de autenticação atual. Requer autenticação.

### Parâmetros de Requisição

Nenhum.

### Respostas

- **`200 OK`**: Token válido. `json { "message": "Token válido!", "user": { "id": 1, "username": "novo_usuario", "email": "user@example.com",`

`"is_admin": false, "credits": 100, "is_banned": false, "otp_enabled": false } }`

- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

# 3. Rotas de Usuário (`src/routes/user.py`)

## 3.1. Obter Perfil do Usuário

`GET /api/user/profile`

Obtém os detalhes do perfil do usuário autenticado. Requer autenticação.

### Parâmetros de Requisição

Nenhum.

### Respostas

- **`200 OK`**: Perfil do usuário retornado com sucesso. `json { "id": 1, "username": "novo_usuario", "email": "user@example.com", "is_admin": false, "credits": 100, "is_banned": false, "otp_enabled": false }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

## 3.2. Atualizar Perfil do Usuário

`PUT /api/user/profile`

Atualiza os detalhes do perfil do usuário autenticado. Requer autenticação.

#### Parâmetros de Requisição (JSON Body)

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `username` | `string` | Não | Novo nome de usuário. | `usuario_atualizado` |
| `email` | `string` | Não | Novo endereço de e-mail. | `updated@example.com` |

#### Respostas

- **`200 OK`**: Perfil atualizado com sucesso. `json { "message": "Perfil atualizado com sucesso!", "user": { "id": 1, "username": "usuario_atualizado", "email": "updated@example.com", "is_admin": false, "credits": 100, "is_banned": false, "otp_enabled": false } }`
- **`400 Bad Request`**: Erro de validação ou nome de usuário/email já existe. `json { "error": "Nome de usuário ou e-mail já existe." }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

### 3.3. Alterar Senha do Usuário

`PUT /api/user/change_password`

Altera a senha do usuário autenticado. Requer autenticação.

#### Parâmetros de Requisição (JSON Body)

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `current_password` | `string` | Sim | Senha atual do usuário. | `SenhaAtual123!` |
| `new_password` | `string` | Sim | Nova senha do usuário (deve ser forte). | `NovaSenha456!` |

#### Respostas

- **`200 OK`**: Senha alterada com sucesso. `json { "message": "Senha alterada com sucesso!" }`

- **`400 Bad Request`**: Senha atual incorreta ou nova senha fraca. `json { "error": "Senha atual incorreta." }` ou `json { "errors": { "new_password": ["A nova senha deve ser forte."] } }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

## 3.4. Ativar 2FA

`POST /api/user/2fa/enable`

Inicia o processo de ativação da autenticação de dois fatores (2FA) para o usuário autenticado. Retorna um QR code para configuração.

**Parâmetros de Requisição**

Nenhum.

**Respostas**

- **`200 OK`**: QR code e segredo retornados. `json { "message": "2FA ativado com sucesso! Escaneie o QR code para configurar seu aplicativo de autenticação.", "qr_code_svg": "<svg>...</svg>", "secret": "JBSWY3DPEHPK3PXP" }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

## 3.5. Verificar 2FA

`POST /api/user/2fa/verify`

Verifica o código TOTP fornecido para completar a ativação do 2FA ou para login com 2FA ativado.

#### Parâmetros de Requisição (JSON Body)

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `otp_code` | `string` | Sim | Código TOTP gerado pelo aplicativo de autenticação. | `123456` |

#### Respostas

- **`200 OK`**: Código TOTP válido. `json { "message": "Código OTP verificado com sucesso!" }`
- **`400 Bad Request`**: Código TOTP inválido. `json { "error": "Código OTP inválido." }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

### 3.6. Desativar 2FA

`POST /api/user/2fa/disable`

Desativa a autenticação de dois fatores (2FA) para o usuário autenticado.

#### Parâmetros de Requisição (JSON Body)

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `password` | `string` | Sim | Senha do usuário para confirmação. | `SenhaSegura123!` |

#### Respostas

- **`200 OK`**: 2FA desativado com sucesso. `json { "message": "2FA desativado com sucesso!" }`
- **`400 Bad Request`**: Senha incorreta. `json { "error": "Senha incorreta." }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

# 4. Rotas Administrativas (`src/routes/user.py` - Requer `is_admin: true`)

## 4.1. Listar Todos os Usuários

`GET /api/admin/users`

Obtém uma lista de todos os usuários registrados no sistema. Requer autenticação e privilégios de administrador.

### Parâmetros de Requisição

Nenhum.

### Respostas

- **`200 OK`**: Lista de usuários retornada com sucesso. `json { "users": [ { "id": 1, "username": "admin", "email": "admin@example.com", "is_admin": true, "credits": 9999, "is_banned": false, "otp_enabled": true }, { "id": 2, "username": "usuario_comum", "email": "user@example.com", "is_admin": false, "credits": 100, "is_banned": false, "otp_enabled": false } ] }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`
- **`403 Forbidden`**: Usuário não tem privilégios de administrador. `json { "error": "Acesso negado. Requer privilégios de administrador." }`

## 4.2. Obter Detalhes de um Usuário

`GET /api/admin/users/<int:user_id>`

Obtém os detalhes de um usuário específico pelo seu ID. Requer autenticação e privilégios de administrador.

**Parâmetros de Requisição (URL Path)**

| Parâmetro | Tipo | Descrição | Exemplo |
|---|---|---|---|
| `user_id` | `integer` | ID do usuário. | `1` |

**Respostas**

- **`200 OK`**: Detalhes do usuário retornados com sucesso. `json { "id": 1, "username": "admin", "email": "admin@example.com", "is_admin": true, "credits": 9999, "is_banned": false, "otp_enabled": true }`
- **`404 Not Found`**: Usuário não encontrado. `json { "error": "Usuário não encontrado." }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`
- **`403 Forbidden`**: Usuário não tem privilégios de administrador. `json { "error": "Acesso negado. Requer privilégios de administrador." }`

## 4.3. Criar Novo Usuário (Admin)

`POST /api/admin/users`

Cria um novo usuário no sistema com privilégios de administrador. Requer autenticação e privilégios de administrador.

**Parâmetros de Requisição (JSON Body)**

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `username` | `string` | Sim | Nome de usuário único. | `novo_admin` |
| `email` | `string` | Sim | Endereço de e-mail único. | `admin2@example.com` |
| `password` | `string` | Sim | Senha do usuário (deve ser forte). | `AdminPass123!` |
| `is_admin` | `boolean` | Não | Define se o usuário é administrador. Padrão: `false`. | `true` |

**Respostas**

- **`201 Created`**: Usuário criado com sucesso. `json { "message": "Usuário criado com sucesso!", "user": { "id": 3, "username": "novo_admin", "email": "admin2@example.com", "is_admin": true, "credits": 0, "is_banned": false, "otp_enabled": false } }`
- **`400 Bad Request`**: Erro de validação ou usuário/email já existe. `json { "error": "Nome de usuário ou e-mail já existe." }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`
- **`403 Forbidden`**: Usuário não tem privilégios de administrador. `json { "error": "Acesso negado. Requer privilégios de administrador." }`

## 4.4. Atualizar Usuário (Admin)

`PUT /api/admin/users/<int:user_id>`

Atualiza os detalhes de um usuário específico. Requer autenticação e privilégios de administrador.

### Parâmetros de Requisição (URL Path e JSON Body)

| Parâmetro | Tipo | Descrição | Exemplo |
|---|---|---|---|
| `user_id` | `integer` | ID do usuário. | `2` |

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `username` | `string` | Não | Novo nome de usuário. | `user_updated` |
| `email` | `string` | Não | Novo endereço de e-mail. | `user_updated@example.com` |
| `password` | `string` | Não | Nova senha do usuário (deve ser forte). | `NewPass456!` |
| `is_admin` | `boolean` | Não | Define se o usuário é administrador. | `true` |
| `credits` | `integer` | Não | Quantidade de créditos do usuário. | `200` |
| `is_banned` | `boolean` | Não | Define se o usuário está banido. | `true` |

### Respostas

- **`200 OK`**: Usuário atualizado com sucesso. `json { "message": "Usuário atualizado com sucesso!", "user": { "id": 2, "username": "user_updated", "email": "user_updated@example.com", "is_admin": false, "credits": 200, "is_banned": true, "otp_enabled": false } }`
- **`400 Bad Request`**: Erro de validação ou nome de usuário/email já existe. `json { "error": "Nome de usuário ou e-mail já existe." }`
- **`404 Not Found`**: Usuário não encontrado. `json { "error": "Usuário não encontrado." }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

- **`403 Forbidden`**: Usuário não tem privilégios de administrador. `json { "error": "Acesso negado. Requer privilégios de administrador." }`

## 4.5. Deletar Usuário (Admin)

`DELETE /api/admin/users/<int:user_id>`

Deleta um usuário específico pelo seu ID. Requer autenticação e privilégios de administrador.

### Parâmetros de Requisição (URL Path)

| Parâmetro | Tipo | Descrição | Exemplo |
|---|---|---|---|
| `user_id` | `integer` | ID do usuário. | `2` |

### Respostas

- **`200 OK`**: Usuário deletado com sucesso. `json { "message": "Usuário deletado com sucesso!" }`
- **`404 Not Found`**: Usuário não encontrado. `json { "error": "Usuário não encontrado." }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`
- **`403 Forbidden`**: Usuário não tem privilégios de administrador. `json { "error": "Acesso negado. Requer privilégios de administrador." }`

## 4.6. Gerenciar Créditos do Usuário (Admin)

`POST /api/admin/users/<int:user_id>/credits`

Adiciona ou remove créditos de um usuário específico. Requer autenticação e privilégios de administrador.

**Parâmetros de Requisição (URL Path e JSON Body)**

| Parâmetro | Tipo | Descrição | Exemplo |
|---|---|---|---|
| `user_id` | `integer` | ID do usuário. | `2` |

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `credits` | `integer` | Sim | Quantidade de créditos a adicionar/remover. Pode ser positivo ou negativo. | `50` |

**Respostas**

- **`200 OK`**: Créditos atualizados com sucesso. `json { "message": "Créditos atualizados com sucesso!", "user": { "id": 2, "username": "usuario_comum", "email": "user@example.com", "is_admin": false, "credits": 150, "is_banned": false, "otp_enabled": false } }`
- **`400 Bad Request`**: Quantidade de créditos inválida. `json { "error": "Quantidade de créditos inválida." }`
- **`404 Not Found`**: Usuário não encontrado. `json { "error": "Usuário não encontrado." }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`
- **`403 Forbidden`**: Usuário não tem privilégios de administrador. `json { "error": "Acesso negado. Requer privilégios de administrador." }`

## 4.7. Banir/Desbanir Usuário (Admin)

`POST /api/admin/users/<int:user_id>/ban`

Bane ou desbane um usuário específico. Requer autenticação e privilégios de administrador.

**Parâmetros de Requisição (URL Path e JSON Body)**

| Parâmetro | Tipo | Descrição | Exemplo |
|---|---|---|---|
| `user_id` | `integer` | ID do usuário. | `2` |

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `ban` | `boolean` | Sim | `true` para banir, `false` para desbanir. | `true` |

**Respostas**

- **`200 OK`**: Status de banimento atualizado com sucesso. `json { "message": "Usuário banido com sucesso!" }` ou `json { "message": "Usuário desbanido com sucesso!" }`
- **`404 Not Found`**: Usuário não encontrado. `json { "error": "Usuário não encontrado." }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`
- **`403 Forbidden`**: Usuário não tem privilégios de administrador. `json { "error": "Acesso negado. Requer privilégios de administrador." }`

## 4.8. Obter Logs do Sistema (Admin)

`GET /api/admin/logs`

Obtém uma lista de logs do sistema. Requer autenticação e privilégios de administrador.

### Parâmetros de Requisição (Query Parameters)

| Parâmetro | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `level` | `string` | Não | Filtra logs por nível (INFO, WARNING, ERROR, CRITICAL). | `ERROR` |
| `user_id` | `integer` | Não | Filtra logs por ID de usuário. | `1` |
| `limit` | `integer` | Não | Limite de resultados. Padrão: `100`. | `50` |
| `offset` | `integer` | Não | Offset para paginação. Padrão: `0`. | `100` |

### Respostas

- **`200 OK`**: Lista de logs retornada com sucesso. `json { "logs": [ { "id": 1, "timestamp": "2025-06-28T10:00:00.000Z", "level": "INFO", "message": "Usuário 'admin' logado com sucesso.", "user_id": 1, "ip_address": "192.168.1.1", "event_type": "auth", "extra_data": {"device": "desktop"} }, { "id": 2, "timestamp": "2025-06-28T10:05:00.000Z", "level": "ERROR", "message": "Tentativa de login falha para 'usuario_inexistente'.", "user_id": null, "ip_address": "192.168.1.2", "event_type": "auth_fail", "extra_data": {"username_attempt": "usuario_inexistente"} } ] }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`
- **`403 Forbidden`**: Usuário não tem privilégios de administrador. `json { "error": "Acesso negado. Requer privilégios de administrador." }`

## 4.9. Obter Estatísticas do Sistema (Admin)

`GET /api/admin/stats`

Obtém estatísticas gerais do sistema, como número de usuários, campanhas ativas, etc. Requer autenticação e privilégios de administrador.

**Parâmetros de Requisição**

Nenhum.

**Respostas**

- **`200 OK`**: Estatísticas retornadas com sucesso. `json { "total_users": 100, "active_campaigns": 5, "total_emails_sent": 10000, "total_clicks": 500, "total_credentials_captured": 50, "storage_used_gb": 0.5, "last_backup_date": "2025-06-27T23:00:00.000Z" }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`
- **`403 Forbidden`**: Usuário não tem privilégios de administrador. `json { "error": "Acesso negado. Requer privilégios de administrador." }`

# 5. Rotas de Campanhas (`src/routes/campaign.py` - Exemplo)

(Este é um exemplo. As rotas reais podem variar e serão documentadas conforme implementadas.)

## 5.1. Criar Nova Campanha

`POST /api/campaigns`

Cria uma nova campanha de phishing. Requer autenticação.

**Parâmetros de Requisição (JSON Body)**

| Campo | Tipo | Obrigatório | Descrição | Exemplo |
|---|---|---|---|---|
| `name` | `string` | Sim | Nome da campanha. | `Campanha de Teste` |
| `description` | `string` | Não | Descrição da campanha. | `Phishing para RH` |
| `template_id` | `integer` | Sim | ID do modelo de e-mail/página a ser usado. | `1` |
| `target_list` | `array` | Sim | Lista de e-mails dos alvos. | `["a@b.com", "c@d.com"]` |

**Respostas**

- **`201 Created`**: Campanha criada com sucesso. `json { "message": "Campanha criada com sucesso!", "campaign": { "id": 1, "name": "Campanha de Teste", "status": "created", "created_at": "2025-06-28T11:00:00.000Z" } }`
- **`400 Bad Request`**: Erro de validação. `json { "errors": { "name": ["O nome da campanha é obrigatório."] } }`
- **`401 Unauthorized`**: Token inválido ou ausente. `json { "error": "Token de autenticação inválido ou ausente." }`

# 6. Modelos de Banco de Dados

(Esta seção será detalhada em `DATABASE_SCHEMA.md`)

# 7. Referências

[1] Flask Documentation: https://flask.palletsprojects.com/en/latest/ [2] React Documentation: https://react.dev/ [3] SQLAlchemy Documentation: https://docs.sqlalchemy.org/en/latest/ [4] Marshmallow Documentation:

https://marshmallow.readthedocs.io/en/stable/ [5] OWASP Top 10: https://owasp.org/www-project-top-ten/

---

**Autor:** Manus AI **Data:** 28 de Junho de 2025