GUIA DO USUÁRIO - Sistema de Geração Automática de APIs

Introdução

Este sistema permite que você **crie APIs RESTful completas** editando apenas **2 arquivos de configuração**.

Você não precisa escrever código! Apenas configure suas tabelas e endpoints, e o sistema gera automaticamente: - Migrations (estrutura do banco de dados) - Models (representação das tabelas) - Controllers (lógica de negócio) - Rotas (endpoints da API) - Validações automáticas - Autenticação com tokens - Middlewares de segurança

---

Sumário

Os Dois Arquivos Que Você Vai Editar

Localização dos arquivos:

```
api/
  config/
     api_tables.php      ← ARQUIVO 1: Define suas tabelas
     api_endpoints.php   ← ARQUIVO 2: Define seus endpoints
```

Fluxo de trabalho:

```
1. Editar api_tables.php
   ↓
2. Editar api_endpoints.php
   ↓
3. Executar: php artisan api:generate
   ↓
4. Executar: php artisan migrate
   ↓
5. API pronta!
```

Arquivo 1: api_tables.php

Propósito: Define a estrutura do seu banco de dados (tabelas, campos, tipos, validações).

Estrutura Básica

```php
return [
    'nome_da_tabela' => [
        'fields' => [
            'nome_do_campo' => [
```

```
                'type' => 'tipo_do_campo',
                'length' => 255,
                'nullable' => false,
                'unique' => true,
                'validation' => 'regras_de_validacao',
            ],
        ],
        'timestamps' => true,
        'soft_deletes' => false,
    ],
];
```

Tipos de Campos Disponíveis

Tipo	Descrição	Exemplo de Uso
`string`	Texto curto (padrão 255 caracteres)	Nome, email, título
`text`	Texto longo	Descrição, conteúdo
`integer`	Número inteiro	Idade, quantidade
`bigInteger`	Número inteiro grande	IDs, contadores
`boolean`	Verdadeiro/Falso	Ativo, publicado
`date`	Data (YYYY-MM-DD)	Data de nascimento
`datetime`	Data e hora	Data de criação
`timestamp`	Timestamp Unix	Data de atualização
`decimal`	Número decimal	Preço, valor
`float`	Número decimal (menos preciso)	Média, percentual
`json`	Objeto JSON	Configurações, metadata

Propriedades de Campos

type (obrigatório) Define o tipo de dado do campo.

```
'idade' => [
    'type' => 'integer',
]
```

length (opcional) Define o tamanho máximo (usado com string/char).

```
'nome' => [
    'type' => 'string',
    'length' => 100,  // Máximo 100 caracteres
]
```

nullable (opcional, padrão: false) Define se o campo pode ser nulo.

```
'telefone' => [
    'type' => 'string',
    'nullable' => true,  // Campo opcional
]
```

unique (opcional, padrão: false) Define se o valor deve ser único no banco.

```
'email' => [
    'type' => 'string',
    'unique' => true,  // Não permite duplicatas
]
```

default (opcional) Define um valor padrão.

```
'status' => [
    'type' => 'string',
    'default' => 'ativo',  // Valor padrão
]
```

unsigned (opcional, padrão: false) Para números inteiros, define se aceita apenas valores positivos.

```
'quantidade' => [
    'type' => 'integer',
    'unsigned' => true,  // Apenas números positivos
]
```

validation (obrigatório para campos editáveis) Define as regras de validação do Laravel.

```
'email' => [
    'type' => 'string',
    'validation' => 'required|email|unique:users,email',
]
```

Regras de validação comuns: - `required` - Campo obrigatório - `nullable` - Campo opcional - `email` - Deve ser um email válido - `unique:tabela,campo` - Deve ser único - `min:5` - Mínimo de 5 caracteres - `max:100` - Máximo de 100 caracteres - `in:valor1,valor2` - Deve ser um dos valores - `integer` - Deve ser inteiro - `date` - Deve ser uma data válida - `exists:tabela,campo` - Deve existir em outra tabela

hidden (opcional, padrão: false) Define se o campo deve ser oculto nas respostas JSON (usado para senhas, tokens).

```
'password' => [
    'type' => 'string',
    'hidden' => true,  // Não aparece no JSON
]
```

encrypted (opcional, padrão: false) Define se o campo deve ser criptografado automaticamente (senhas).

```
'password' => [
    'type' => 'string',
    'encrypted' => true,  // Será criptografado
]
```

foreign_key (opcional) Define relacionamento com outra tabela.

```
'user_id' => [
    'type' => 'bigInteger',
    'unsigned' => true,
    'foreign_key' => 'users.id',  // Relaciona com users.id
]
```

Propriedades da Tabela

timestamps (opcional, padrão: false) Adiciona automaticamente os campos `created_at` e `updated_at`.

```
'timestamps' => true,
```

soft_deletes (opcional, padrão: false) Adiciona o campo `deleted_at` para exclusão lógica (não deleta fisicamente).

```
'soft_deletes' => true,
```

Exemplo Completo: Tabela de Produtos

```
'produtos' => [
    'fields' => [
        'nome' => [
            'type' => 'string',
            'length' => 255,
            'nullable' => false,
            'validation' => 'required|string|max:255',
        ],
        'descricao' => [
            'type' => 'text',
            'nullable' => true,
            'validation' => 'nullable|string',
        ],
        'preco' => [
            'type' => 'decimal',
            'nullable' => false,
            'validation' => 'required|numeric|min:0',
        ],
        'estoque' => [
            'type' => 'integer',
            'unsigned' => true,
            'default' => 0,
            'validation' => 'required|integer|min:0',
        ],
        'categoria_id' => [
            'type' => 'bigInteger',
            'unsigned' => true,
            'nullable' => false,
            'foreign_key' => 'categorias.id',
            'validation' => 'required|integer|exists:categorias,id',
        ],
        'ativo' => [
            'type' => 'boolean',
            'default' => true,
            'validation' => 'required|boolean',
        ],
    ],
    'timestamps' => true,
    'soft_deletes' => false,
],
```

Arquivo 2: api_endpoints.php

Propósito: Define os endpoints da sua API (rotas, métodos HTTP, permissões).

Estrutura Básica

```
return [
    'prefix' => 'api',
    'version' => null,

    'auth' => [
        'user_table' => 'users',
        'username_field' => 'username',
        'password_field' => 'password',
        'token_field' => 'token',
        'hash_algorithm' => 'bcrypt',
    ],

    'auth_endpoints' => [
        // Endpoints de autenticação
    ],

    'custom_endpoints' => [
        // Seus endpoints personalizados
    ],
];
```

Configuração de Autenticação

```
'auth' => [
    'user_table' => 'users',          // Tabela de usuários
    'username_field' => 'username',    // Campo para login
    'password_field' => 'password',    // Campo de senha
    'token_field' => 'token',          // Campo do token
    'token_expires' => false,          // Token expira? (false ou número de horas)
    'hash_algorithm' => 'bcrypt',      // bcrypt ou md5
],
```

Endpoints de Autenticação (Padrão)

Estes endpoints já vêm configurados:

```
'auth_endpoints' => [
    'signup' => [
        'enabled' => true,
        'path' => '/signup',
        'method' => 'POST',
        'fields' => ['name', 'email', 'username', 'password'],
        'responses' => [
            201 => ['message' => 'Cadastro efetuado com sucesso'],
            422 => ['message' => 'Dados inválidos'],
        ],
    ],
    'login' => [
        'enabled' => true,
        'path' => '/login',
        'method' => 'POST',
        'responses' => [
            200 => ['token' => '{token}'],
            401 => ['message' => 'Login inválido'],
```

```
        ],
    ],
    'logout' => [
        'enabled' => true,
        'path' => '/logout',
        'method' => 'GET',
        'middleware' => ['auth.api'],
        'responses' => [
            200 => ['message' => 'Logout efetuado com sucesso'],
        ],
    ],
],
```

Endpoints Customizados

Aqui você define os endpoints específicos da sua API.

Estrutura de um endpoint:

```
'nome_do_endpoint' => [
    'enabled' => true,                    // Ativar/desativar
    'path' => '/caminho',                 // URL do endpoint
    'method' => 'GET',                    // GET, POST, PUT, DELETE
    'table' => 'nome_tabela',             // Tabela relacionada
    'controller' => 'Controller@metodo',  // Controller e método
    'middleware' => ['auth.api'],         // Middlewares
    'permission' => 'authenticated',      // Permissão necessária
    'responses' => [                      // Respostas esperadas
        200 => ['message' => 'Sucesso'],
    ],
],
```

Propriedades de um endpoint: **enabled** (obrigatório)

```
'enabled' => true,  // true ou false
```

path (obrigatório)

```
'path' => '/produtos',           // Listar todos
'path' => '/produto/{id}',       // Com parâmetro
'path' => '/produtos/{categoria}',  // Com parâmetro nomeado
```

method (obrigatório)

```
'method' => 'GET',     // Buscar dados
'method' => 'POST',    // Criar
'method' => 'PUT',     // Atualizar
'method' => 'DELETE',  // Deletar
```

table (obrigatório)

```
'table' => 'produtos',  // Nome da tabela relacionada
```

controller (obrigatório)

```
// Formato: NomeController@metodo
'controller' => 'ProdutoController@index',    // Listar
'controller' => 'ProdutoController@show',     // Mostrar um
'controller' => 'ProdutoController@store',    // Criar
```

```
'controller' => 'ProdutoController@update',   // Atualizar
'controller' => 'ProdutoController@destroy',  // Deletar
```

middleware (opcional)

```
'middleware' => ['auth.api'],  // Requer autenticação
'middleware' => [],            // Público
```

permission (opcional)

```
'permission' => 'authenticated',      // Qualquer usuário logado
'permission' => ['role' => 'admin'],  // Apenas admin
```

responses (opcional)

```
'responses' => [
    200 => ['message' => 'Sucesso'],
    404 => ['message' => 'Não encontrado'],
    422 => ['message' => 'Dados inválidos'],
],
```

Exemplo Completo: CRUD de Produtos

```
'custom_endpoints' => [

    // Listar todos os produtos
    'lista_produtos' => [
        'enabled' => true,
        'path' => '/produtos',
        'method' => 'GET',
        'table' => 'produtos',
        'controller' => 'ProdutoController@index',
        'middleware' => ['auth.api'],
    ],

    // Buscar um produto específico
    'busca_produto' => [
        'enabled' => true,
        'path' => '/produto/{id}',
        'method' => 'GET',
        'table' => 'produtos',
        'controller' => 'ProdutoController@show',
        'middleware' => ['auth.api'],
    ],

    // Criar novo produto
    'adiciona_produto' => [
        'enabled' => true,
        'path' => '/produto',
        'method' => 'POST',
        'table' => 'produtos',
        'controller' => 'ProdutoController@store',
        'middleware' => ['auth.api'],
        'permission' => ['role' => 'admin'],
        'responses' => [
            201 => ['message' => 'Produto criado com sucesso'],
```

```
            422 => ['message' => 'Dados inválidos'],
        ],
    ],

    // Atualizar produto
    'atualiza_produto' => [
        'enabled' => true,
        'path' => '/produto/{id}',
        'method' => 'PUT',
        'table' => 'produtos',
        'controller' => 'ProdutoController@update',
        'middleware' => ['auth.api'],
        'permission' => ['role' => 'admin'],
        'responses' => [
            200 => ['message' => 'Produto atualizado'],
            404 => ['message' => 'Produto não encontrado'],
        ],
    ],

    // Deletar produto
    'deleta_produto' => [
        'enabled' => true,
        'path' => '/produto/{id}',
        'method' => 'DELETE',
        'table' => 'produtos',
        'controller' => 'ProdutoController@destroy',
        'middleware' => ['auth.api'],
        'permission' => ['role' => 'admin'],
        'responses' => [
            200 => ['message' => 'Produto deletado'],
            404 => ['message' => 'Produto não encontrado'],
        ],
    ],

    // Buscar produtos por categoria
    'produtos_categoria' => [
        'enabled' => true,
        'path' => '/produtos/categoria/{categoria_id}',
        'method' => 'GET',
        'table' => 'produtos',
        'controller' => 'ProdutoController@byCategoria',
        'middleware' => ['auth.api'],
        'filter_by' => 'categoria_id',
    ],

],
```

Exemplos Práticos

Exemplo 1: Blog Simples

api_tables.php

```
return [
    'users' => [
        'fields' => [
            'name' => ['type' => 'string', 'validation' => 'required|max:255'],
            'email' => ['type' => 'string', 'unique' => true, 'validation' => 'required|email|unique:us
            'username' => ['type' => 'string', 'unique' => true, 'validation' => 'required|unique:users
            'password' => ['type' => 'string', 'hidden' => true, 'encrypted' => true, 'validation' => '
            'token' => ['type' => 'string', 'length' => 500, 'nullable' => true, 'hidden' => true],
        ],
        'timestamps' => true,
    ],

    'posts' => [
        'fields' => [
            'titulo' => ['type' => 'string', 'validation' => 'required|max:255'],
            'conteudo' => ['type' => 'text', 'validation' => 'required'],
            'autor_id' => ['type' => 'bigInteger', 'unsigned' => true, 'foreign_key' => 'users.id', 'val
            'publicado' => ['type' => 'boolean', 'default' => false, 'validation' => 'boolean'],
        ],
        'timestamps' => true,
    ],

    'comentarios' => [
        'fields' => [
            'post_id' => ['type' => 'bigInteger', 'unsigned' => true, 'foreign_key' => 'posts.id', 'vali
            'autor_id' => ['type' => 'bigInteger', 'unsigned' => true, 'foreign_key' => 'users.id', 'val
            'conteudo' => ['type' => 'text', 'validation' => 'required'],
        ],
        'timestamps' => true,
    ],
];
```

api_endpoints.php

```
'custom_endpoints' => [
    'lista_posts' => [
        'enabled' => true,
        'path' => '/posts',
        'method' => 'GET',
        'table' => 'posts',
        'controller' => 'PostController@index',
        'middleware' => [],  // Público
    ],
    'cria_post' => [
        'enabled' => true,
        'path' => '/post',
        'method' => 'POST',
        'table' => 'posts',
        'controller' => 'PostController@store',
        'middleware' => ['auth.api'],
    ],
    'adiciona_comentario' => [
        'enabled' => true,
        'path' => '/post/{id}/comentario',
```

```
        'method' => 'POST',
        'table' => 'comentarios',
        'controller' => 'ComentarioController@store',
        'middleware' => ['auth.api'],
    ],
],
```

Exemplo 2: E-commerce Básico

api_tables.php

```
'categorias' => [
    'fields' => [
        'nome' => ['type' => 'string', 'validation' => 'required|max:100'],
        'descricao' => ['type' => 'text', 'nullable' => true, 'validation' => 'nullable'],
    ],
    'timestamps' => true,
],

'produtos' => [
    'fields' => [
        'nome' => ['type' => 'string', 'validation' => 'required|max:255'],
        'descricao' => ['type' => 'text', 'validation' => 'required'],
        'preco' => ['type' => 'decimal', 'validation' => 'required|numeric|min:0'],
        'estoque' => ['type' => 'integer', 'unsigned' => true, 'default' => 0, 'validation' => 'require
        'categoria_id' => ['type' => 'bigInteger', 'unsigned' => true, 'foreign_key' => 'categorias.id'
        'imagem' => ['type' => 'string', 'nullable' => true, 'validation' => 'nullable|url'],
    ],
    'timestamps' => true,
],

'pedidos' => [
    'fields' => [
        'user_id' => ['type' => 'bigInteger', 'unsigned' => true, 'foreign_key' => 'users.id', 'validat
        'total' => ['type' => 'decimal', 'validation' => 'required|numeric|min:0'],
        'status' => ['type' => 'string', 'default' => 'pendente', 'validation' => 'required|in:pendente
    ],
    'timestamps' => true,
],
```

Comandos Para Gerar a API

Passo a passo completo:

```
# 1. Edite os arquivos de configuração
vim config/api_tables.php
vim config/api_endpoints.php

# 2. Gere toda a estrutura da API
php artisan api:generate

# 3. Execute as migrations (cria as tabelas no banco)
```

```
php artisan migrate

# 4. Inicie o servidor de desenvolvimento
php artisan serve
```

Comandos úteis:

```
# Regenerar tudo do zero (cuidado: apaga o banco!)
php artisan api:generate --fresh
php artisan migrate:fresh

# Ver todas as rotas criadas
php artisan route:list

# Ver logs de erro
tail -f storage/logs/laravel.log

# Limpar cache
php artisan cache:clear
php artisan config:clear
php artisan route:clear
```

Testando Sua API

Usando cURL:

Signup (Registro)

```
curl -X POST http://localhost:8000/api/signup \
  -H "Content-Type: application/json" \
  -d '{
    "name": "João Silva",
    "email": "joao@email.com",
    "username": "joao",
    "password": "senha123"
  }'
```

Login

```
curl -X POST http://localhost:8000/api/login \
  -H "Content-Type: application/json" \
  -d '{
    "username": "joao",
    "password": "senha123"
  }'
```

Usar endpoint protegido (com token)

```
curl -X GET http://localhost:8000/api/produtos \
  -H "Authorization: Bearer SEU_TOKEN_AQUI"
```

Criar produto

```
curl -X POST http://localhost:8000/api/produto \
  -H "Authorization: Bearer SEU_TOKEN_AQUI" \
  -H "Content-Type: application/json" \
  -d '{
```

```
    "nome": "Notebook",
    "preco": 2500.00,
    "estoque": 10,
    "categoria_id": 1
  }'
```

Dicas Avançadas

1. Relacionamentos entre tabelas

Use `foreign_key` para criar relacionamentos:

```
'post_id' => [
    'type' => 'bigInteger',
    'unsigned' => true,
    'foreign_key' => 'posts.id',  // Formato: tabela.coluna
    'validation' => 'required|exists:posts,id',
],
```

2. Campos com valores fixos (enum)

Use validação `in:` para limitar valores:

```
'status' => [
    'type' => 'string',
    'validation' => 'required|in:ativo,inativo,bloqueado',
],
```

3. Campos opcionais

```
'telefone' => [
    'type' => 'string',
    'nullable' => true,
    'validation' => 'nullable|string',
],
```

4. Senhas seguras

Sempre use `hidden` e `encrypted`:

```
'password' => [
    'type' => 'string',
    'hidden' => true,       // Não aparece em JSON
    'encrypted' => true,    // Criptografado automaticamente
    'validation' => 'required|min:8',
],
```

Perguntas Frequentes

P: Posso criar quantas tabelas quiser? R: Sim! Não há limite.

P: Posso ter relacionamentos complexos? R: Sim! Use `foreign_key` para relacionar tabelas.

P: Posso desabilitar um endpoint temporariamente? R: Sim! Mude `'enabled' => false`.

P: Como adiciono novos campos em uma tabela existente? R: Edite o arquivo `api_tables.php`, adicione o campo e rode `php artisan api:generate --fresh` e `php artisan migrate:fresh`.

P: Posso usar MySQL ao invés de SQLite? R: Sim! Configure o `.env` com suas credenciais MySQL.

Conclusão

Com este sistema, você pode criar APIs complexas editando apenas 2 arquivos!

Resumo do fluxo: 1. Editar `api_tables.php` (estrutura do banco) 2. Editar `api_endpoints.php` (rotas da API) 3. Executar `php artisan api:generate` 4. Executar `php artisan migrate` 5. API pronta!

Documentação criada por: Sistema de Geração Automática de APIs
Versão: 1.0
Data: 2025-01-13