Atividade 01 - Migrations no Laravel

Objetivo

O objetivo desta atividade é familiarizar-se com o uso das migrations no Laravel para criar, modificar e gerenciar a estrutura de tabelas no banco de dados. A atividade envolve a criação de tabelas, aplicação de constraints e definição de relacionamentos entre elas.

Tarefa: Gerenciamento de um Sistema de Biblioteca

Você irá criar um sistema básico de gerenciamento de biblioteca que inclui tabelas para `books`, `authors, publishers` e `categories`. A atividade envolve a criação dessas tabelas, aplicação de constraints e definição de relacionamentos entre elas.

INSTRUÇÕES

PASSO 1: Configuração Inicial (caso não tenham um projeto criado)

1.1 Inicie um novo projeto Laravel (se ainda não o fez) executando o comando para criar um novo projeto Laravel e navegue até a pasta do projeto.

```
laravel new atividade_01
```

ou

```
composer create-project --prefer-dist laravel/laravel atividade_01
```

Which starter kit would you like to install? none

Which database will your application use? Mysql

Default database updated. Would you like to run the default database migrati.. ? No

Would you like to run npm install and npm run… Yes

1.2 Configure as credenciais do seu banco de dados no arquivo `.env` para garantir que o Laravel possa se conectar ao seu banco de dados.

PASSO 2: Criar Tabela de Authors (Autores)

2.1 Na linha de comando, execute o comando para criar uma migration para a tabela `authors`.

```
php artisan make:migration create_authors_table
```

2.2 Abra o arquivo gerado na pasta `database/migrations` que corresponde à migration de `authors`.

2.3 Edite o arquivo de migration para que a tabela `authors` tenha os seguintes campos:

- `id`: chave primária (use o comando adequado para primary key).
- `name`: tipo string.
- `email`: tipo string - unique.
- `birth_date`: tipo date, pode ser atribuído nulo.
- `created_at` e `updated_at`: timestamps automáticos.

2.4 Na linha de comando, execute o comando para rodar a migration e criar a tabela `authors` no banco de dados.

```
php artisan migrate
```

PASSO 3: Criar Tabela de Categorias

3.1 Na linha de comando, execute o comando para criar uma migration para a tabela `categories`.

```
php artisan make:migration create_categories_table
```

3.2 Abra o arquivo gerado na pasta `database/migrations` que corresponde à migration de `categories`.

3.3 Edite o arquivo de migration para que a tabela `categories` tenha os seguintes campos:

- `id`: chave primária, tipo inteiro, auto-incremento (use o comando adequado para primary key).
- `name`: tipo string, deve ser único.
- `created_at` e `updated_at`: timestamps automáticos.

3.4 Na linha de comando, execute o comando para rodar a migration e criar a tabela `categories` no banco de dados.

```
php artisan migrate
```

PASSO 4: Criar Tabela de Editora

3.1 Na linha de comando, execute o comando para criar uma migration para a tabela `publishers`.

```
php artisan make:migration create_publishers_table
```

3.2 Abra o arquivo gerado na pasta `database/migrations` que corresponde à migration de `publishers`.

3.3 Edite o arquivo de migration para que a tabela `publishers` tenha os seguintes campos:

- `id`: chave primária, tipo inteiro, auto-incremento (use o comando adequado para primary key).
- `name`: tipo string, deve ser único.
- `address`: tipo string, e pode ser preenchido com nulo.
- `created_at` e `updated_at`: timestamps automáticos.

3.4 Na linha de comando, execute o comando para rodar a migration e criar a tabela `publishers` no banco de dados.

```
php artisan migrate
```

PASSO 5: Criar Tabela de Livros

5.1 Na linha de comando, execute o comando para criar uma migration para a tabela `books`.

```
php artisan make:migration create_books_table --create=books
```

5.2 Abra o arquivo gerado na pasta `database/migrations` que corresponde à migration de `books`.

5.3 Edite o arquivo de migration para que a tabela `books` tenha os seguintes campos:

- `id`: chave primária, tipo inteiro, auto-incremento.
- `title`: tipo string.
- `pages`: tipo inteiro.
- `author_id`: Chave estrangeira para a tabela authors.
- `category_id`: Chave estrangeira para a tabela categories.
- `publisher_id`: Chave estrangeira para a tabela publishers.
- `created_at` e `updated_at`: timestamps automáticos.

5.5 Na linha de comando, execute o comando para rodar a migration e criar a tabela `books` no banco de dados.

```
php artisan migrate
```

PASSO 6: Testando Rollbacks

6.1 Na linha de comando, execute o comando para reverter a última migration (que criou a tabela `books`).

```
php artisan migrate:rollback --step=1
```

6.2 Verifique que a tabela `books` foi removida do banco de dados.

6.3 Na linha de comando, execute o comando para refazer todas as migrations, garantindo que todas as tabelas sejam recriadas.

```
php artisan migrate:refresh
```

PASSO 7: Adicionar Nova Coluna na Tabela de Livros

7.1 Na linha de comando, execute o comando para criar uma migration para adicionar a coluna `published_year` à tabela `books`.

```
php artisan make:migration add_published_year_to_books_table --table=books
```

7.2 Abra o arquivo gerado na pasta `database/migrations` que corresponde à migration para adicionar a nova coluna.

7.3 Edite o arquivo de migration para adicionar a coluna `published_year` do tipo integer, que pode ser nulo, à tabela `books`.

7.4 Na linha de comando, execute o comando para rodar a migration e adicionar a coluna `published_year` à tabela `books`.

```
php artisan migrate
```

PASSO 8: Revertendo e Refazendo Migrations

8.1 Na linha de comando, execute o comando para reverter todas as migrations, removendo todas as tabelas do banco de dados.

```
php artisan migrate:reset
```

8.2 Na linha de comando, execute o comando para refazer todas as migrations, recriando todas as tabelas definidas anteriormente.

```
php artisan migrate
```