Atividade Prática 5.2: Construção do Controller, Rotas e Views para a Entidade `Category`

Objetivo

Nesta atividade, você irá criar o controller para a entidade `Category`, implementando as funcionalidades de CRUD (Create, Read, Update, Delete). Essa etapa é essencial para gerenciar as categorias do sistema e fornecer uma interface para as operações relacionadas.

Passo 1: Criar o Controller para a Entidade `Category`

1.1 **Comando para Gerar o Controller**

- Execute o seguinte comando no terminal para criar o controller com métodos de recurso (`resource`):

```
php artisan make:controller CategoryController --resource
```

Este comando criará um arquivo chamado `CategoryController.php` em `app/Http/Controllers`, já contendo os métodos principais para o CRUD.

1.2 **Estrutura Inicial do Controller**

- Após a criação, abra o arquivo gerado (`app/Http/Controllers/CategoryController.php`) e ajuste os métodos gerados para incluir o comportamento necessário:

```php
namespace App\Http\Controllers;

use App\Models\Category;
use Illuminate\Http\Request;

class CategoryController extends Controller
{
    // Exibe uma lista de categorias
    public function index()
    {
        $categories = Category::all();
        return view('categories.index', compact('categories'));
    }
```

```
    // Mostra o formulário para criar uma nova categoria
    public function create()
    {
        return view('categories.create');
    }

    // Armazena uma nova categoria no banco de dados
    public function store(Request $request)
    {
        $request->validate([
            'name' => 'required|string|unique:categories|max:255',
        ]);

        Category::create($request->all());

        return redirect()->route('categories.index')->with('success', 'Categoria
criada com sucesso.');
    }

    // Exibe uma categoria específica
    public function show(Category $category)
    {
        return view('categories.show', compact('category'));
    }

    // Mostra o formulário para editar uma categoria existente
    public function edit(Category $category)
    {
        return view('categories.edit', compact('category'));
    }

    // Atualiza uma categoria no banco de dados
    public function update(Request $request, Category $category)
    {
        $request->validate([
            'name' => 'required|string|unique:categories,name,' . $category->id .
'|max:255',
        ]);

        $category->update($request->all());

        return redirect()->route('categories.index')->with('success', 'Categoria
atualizada com sucesso.');
    }

    // Remove uma categoria do banco de dados
    public function destroy(Category $category)
    {
        $category->delete();
```

```
        return redirect()->route('categories.index')->with('success', 'Categoria
excluída com sucesso.');
    }
}
```

1.3 **Configuração Adicional**

- **Adicionar Rotas**: Certifique-se de registrar as rotas no arquivo `routes/web.php`:

```
use App\Http\Controllers\CategoryController;

Route::resource('categories', CategoryController::class);
```

- **Criar as Views**: Em breve, você desenvolverá as views necessárias para as operações do CRUD (`index`, `create`, `edit`, `show`).

Você pode listar todas as rotas registradas no Laravel executando:

```
php artisan route:list
```

Descrição das Rotas Geradas

O comando `Route::resource` cria automaticamente as seguintes rotas RESTful associadas ao `CategoryController`:

Método HTTP	Caminho	Nome da Rota	Ação no Controller	Descrição
GET	`/categories`	`categories.index`	`index()`	Lista todas as categorias
GET	`/categories/create`	`categories.create`	`create()`	Formulário para criar
POST	`/categories`	`categories.store`	`store()`	Salva uma nova categoria
GET	`/categories/{id}`	`categories.show`	`show()`	Exibe uma categoria
GET	`/categories/{id}/edit`	`categories.edit`	`edit()`	Formulário para editar
PUT/PATCH	`/categories/{id}`	`categories.update`	`update()`	Atualiza uma categoria
DELETE	`/categories/{id}`	`categories.destroy`	`destroy()`	Exclui uma categoria

Passo 2: Criar as Views para `Category`

2.1 Estrutura de Diretórios Crie uma pasta chamada `categories` dentro do diretório `resources/views` para organizar as views relacionadas.

```
mkdir resources/views/categories
```

2.2 View: `index.blade.php` (Lista de Categorias)

```
@extends('layouts.app')

@section('content')
<div class="container">
    <h1 class="my-4">Lista de Categorias</h1>

    <a href="{{ route('categories.create') }}" class="btn btn-success mb-3">
        <i class="bi bi-plus"></i> Adicionar Categoria
    </a>

    @if(session('success'))
        <div class="alert alert-success">
            {{ session('success') }}
        </div>
    @endif

    <table class="table table-striped">
        <thead>
            <tr>
                <th>#</th>
                <th>Nome</th>
                <th>Ações</th>
            </tr>
        </thead>
        <tbody>
            @forelse($categories as $category)
                <tr>
                    <td>{{ $loop->iteration }}</td>
                    <td>{{ $category->name }}</td>
                    <td>
                        <!-- Botão de Visualizar -->
                        <a href="{{ route('categories.show', $category) }}"
class="btn btn-info btn-sm">
                            <i class="bi bi-eye"></i> Visualizar
                        </a>

                        <!-- Botão de Editar -->
                        <a href="{{ route('categories.edit', $category) }}"
class="btn btn-primary btn-sm">
```

```
                            <i class="bi bi-pencil"></i> Editar
                        </a>

                        <!-- Botão de Excluir -->
                        <form action="{{ route('categories.destroy', $category)
}}" method="POST" style="display: inline;">
                            @csrf
                            @method('DELETE')
                            <button class="btn btn-danger btn-sm" onclick="return
confirm('Deseja excluir esta categoria?')">
                                <i class="bi bi-trash"></i> Excluir
                            </button>
                        </form>
                    </td>
                </tr>
            @empty
                <tr>
                    <td colspan="3">Nenhuma categoria encontrada.</td>
                </tr>
            @endforelse
        </tbody>
    </table>
</div>
@endsection
```

2.3 View: `create.blade.php` (Criar Categoria)

```
@extends('layouts.app')

@section('content')
<div class="container">
    <h1 class="my-4">Adicionar Categoria</h1>

    <form action="{{ route('categories.store') }}" method="POST">
        @csrf
        <div class="mb-3">
            <label for="name" class="form-label">Nome</label>
            <input type="text" class="form-control @error('name') is-invalid
@enderror" id="name" name="name" value="{{ old('name') }}" required>
            @error('name')
                <div class="invalid-feedback">
                    {{ $message }}
                </div>
            @enderror
        </div>

        <button type="submit" class="btn btn-success">
            <i class="bi bi-save"></i> Salvar
```

```
        </button>
        <a href="{{ route('categories.index') }}" class="btn btn-secondary">
            <i class="bi bi-arrow-left"></i> Voltar
        </a>
    </form>
</div>
@endsection
```

2.4 View: `edit.blade.php` (Editar Categoria)

```
@extends('layouts.app')

@section('content')
<div class="container">
    <h1 class="my-4">Editar Categoria</h1>

    <form action="{{ route('categories.update', $category) }}" method="POST">
        @csrf
        @method('PUT')
        <div class="mb-3">
            <label for="name" class="form-label">Nome</label>
            <input type="text" class="form-control @error('name') is-invalid
@enderror" id="name" name="name" value="{{ old('name', $category->name) }}"
required>
            @error('name')
                <div class="invalid-feedback">
                    {{ $message }}
                </div>
            @enderror
        </div>

        <button type="submit" class="btn btn-success">
            <i class="bi bi-save"></i> Atualizar
        </button>
        <a href="{{ route('categories.index') }}" class="btn btn-secondary">
            <i class="bi bi-arrow-left"></i> Voltar
        </a>
    </form>
</div>
@endsection
```

2.5 View: `show.blade.php` (Exibir Categoria)

```
@extends('layouts.app')

@section('content')
<div class="container">
```

```
    <h1 class="my-4">Detalhes da Categoria</h1>

    <div class="card">
        <div class="card-header">
            Categoria: {{ $category->name }}
        </div>
        <div class="card-body">
            <p><strong>ID:</strong> {{ $category->id }}</p>
            <p><strong>Nome:</strong> {{ $category->name }}</p>
        </div>
    </div>

    <a href="{{ route('categories.index') }}" class="btn btn-secondary mt-3">
        <i class="bi bi-arrow-left"></i> Voltar
    </a>
</div>
@endsection
```

2.6 Inclusão de Ícones (Bootstrap) Adicione a biblioteca de ícones do bootstrap no arquivo `resources/views/layouts/app.blade.php` para usar os ícones nos botões:

```
<head>
    <link
href="https://cdn.jsdelivr.net/npm/bootstrap-icons/font/bootstrap-icons.css"
rel="stylesheet">
</head>
```

Passo 3: Testar as Funcionalidades

Rodar em um terminal o comando (deixa rodando em segundo plano)

```
npm install && npm run dev
```

Em um segundo terminal:

```
composer run dev
```

Teste as funcionalidades implementadas para garantir que tudo funcione corretamente. Utilize as rotas abaixo para realizar as operações de CRUD:

Caminho	Descrição
`/categories`	Lista todas as categorias
`/categories/create`	Adiciona uma nova categoria
`/categories/{id}`	Exibe detalhes de uma categoria
`/categories/{id}/edit`	Editar uma categoria existente
`/categories/{id}`	Exclui uma categoria

O que Fazer:

1. **Cadastrar**: Adicione novas categorias através do formulário de criação.
2. **Listar**: Verifique se as categorias aparecem na listagem.
3. **Editar**: Altere os dados de uma categoria existente.
4. **Visualizar**: Consulte os detalhes de qualquer categoria.
5. **Excluir**: Remova categorias e confirme a exclusão.

Passo 4: Replicar para Author e Publisher

Após terminar essa atividade vocês vão criar, seguindo o mesmo padrão, o Controller, Rotas e Views para as entidades **Author** e **Publisher**.