



Laravel 12: Criando um CRUD para Alunos

Rotas, Controllers e Templates

Uma abordagem prática para implementar operações CRUD (Create, Read, Update, Delete) em uma aplicação de gerenciamento de alunos utilizando Laravel 12.

Rotas RESTful

```
Route::resource('aluno', App\Http\Controllers\AlunoController::class);
```

O que é Route::resource?

O método **resource()** do Laravel cria automaticamente 7 rotas RESTful para o recurso especificado, mapeando-as para os métodos correspondentes no controller.

Método HTTP	URI	Nome da Rota	Método do Controller
GET	/aluno	aluno.index	index
GET	/aluno/create	aluno.create	create
POST	/aluno	aluno.store	store
GET	/aluno/{aluno}	aluno.show	show
GET	/aluno/{aluno}/edit	aluno.edit	edit
PUT/PATCH	/aluno/{aluno}	aluno.update	update

Controller: Método Index

```
/**
 * Lista todos os alunos.
 */
public function index()
{
    $alunos = Aluno::all();
    return view('alunos.index',
compact('alunos'));
}
```

O que faz o método index?

O método **index()** é responsável por listar todos os registros de alunos cadastrados no sistema.

- 1 Recupera todos os registros da tabela **alunos** usando `Aluno::all()`
- 2 Armazena os resultados na variável **\$alunos**
- 3 Retorna a view **alunos.index**
- 4 Passa a coleção de alunos para a view usando `compact('alunos')`

Controller: Método Create

```
/**
 * Mostra o formulário de criação.
 */
public function create()
{
    return view("alunos.create");
}
```

📌 Este método é chamado quando o usuário acessa a rota **GET /aluno/create**

O que faz o método create?

O método **create()** é responsável por exibir o formulário para criação de um novo aluno.

- 1 Usuário clica no botão "Adicionar Novo Aluno"
- 2 O navegador acessa a rota **GET /aluno/create**
- 3 O Laravel direciona a requisição para o método **create()**
- 4 O método retorna a view **alunos.create** com o formulário

Controller: Método Store

```
/**
 * Salva um novo aluno no banco.
 */
public function store(Request $request)
{
    $validated = $request->validate([
        'nome' => 'required|string|max:255',
        'email' =>
'required|email|unique:alunos,email',
        'idade' => 'nullable|integer|min:0',
    ]);

    Aluno::create($validated);

    return redirect()->route('alunos.index')
        ->with('success', 'Aluno
cadastrado com sucesso!');
}
```

O que faz o método store?

O método **store()** é responsável por validar os dados do formulário e salvar um novo aluno no banco de dados.

Regras de Validação:

- ✓ **nome:** obrigatório, texto, máximo 255 caracteres
- ✓ **email:** obrigatório, formato de email válido, único na tabela alunos
- ✓ **idade:** opcional, número inteiro, valor mínimo 0

Após validação bem-sucedida, o método:

- Cria um novo registro usando `Aluno::create($validated)`
- Redireciona para a rota `alunos.index`
- Adiciona mensagem de sucesso à sessão

Controller: Método Show





```
/**
 * Mostra os dados de um aluno específico.
 */
public function show(Aluno $aluno)
{
    return view("alunos.show",
compact("aluno"));
}
```

📌 Este método é chamado quando o usuário acessa a rota **GET /aluno/{aluno}**

O que faz o método show?

O método **show()** é responsável por exibir os detalhes de um aluno específico.

Características importantes:

-  **Route Model Binding:** O Laravel automaticamente busca o aluno pelo ID na rota
-  **Injeção de Dependência:** O parâmetro `$aluno` já contém o modelo carregado
-  **Visualização:** Retorna a view `alunos.show` com os dados do aluno
-  **Tratamento de Erro:** Se o aluno não existir, o Laravel retorna automaticamente um erro 404

Controller: Método Edit

```
/**
 * Mostra formulário de edição.
 */
public function edit(Aluno $aluno)
{
    return view("alunos.edit",
compact("aluno"));
}
```

📌 Este método é chamado quando o usuário acessa a rota **GET /aluno/{aluno}/edit**

O que faz o método edit?

O método **edit()** é responsável por exibir o formulário para edição de um aluno existente.

- 1 Recebe uma instância do modelo **Aluno** via injeção de dependência
- 2 O Laravel automaticamente busca o aluno pelo ID na rota
- 3 Retorna a view **alunos.edit**
- 4 Passa o objeto **\$aluno** para a view usando `compact("aluno")`
- 5 A view exibe um formulário pré-preenchido com os dados do aluno

Controller: Método Update

```
/**
 * Atualiza um aluno existente.
 */
public function update(Request $request, Aluno
$aluno)
{
    $validated = $request->validate([
        'nome' => 'required|string|max:255',
        'email' =>
'required|email|unique:alunos,email,' . $aluno->id,
        'idade' => 'nullable|integer|min:0',
    ]);

    $aluno->update($validated);

    return redirect()->route('alunos.index')
        ->with('success', 'Aluno
atualizado com sucesso!');
}
```

O que faz o método update?

O método **update()** é responsável por atualizar os dados de um aluno existente no banco de dados.

- 1 Recebe os dados do formulário (`$request`) e o modelo do aluno (`$aluno`)
- 2 Valida os dados recebidos conforme as regras definidas
- 3 Note a regra `unique:alunos,email,' . $aluno->id` que ignora o próprio email do aluno
- 4 Atualiza o registro do aluno com `$aluno->update($validated)`
- 5 Redireciona para a lista de alunos com mensagem de sucesso

Controller: Método Destroy

```
/**
 * Remove um aluno do banco.
 */
public function destroy(Aluno $aluno)
{
    $aluno->delete();

    return redirect()->route("alunos.index")
        ->with("success", "Aluno
excluído com sucesso!");
}
```

O que faz o método destroy?

O método **destroy()** é responsável por remover um aluno do banco de dados.

- 1 Recebe o modelo do aluno (`$aluno`) via injeção de dependência
- 2 Chama o método `delete()` para remover o registro do banco
- 3 Redireciona para a lista de alunos (`alunos.index`)
- 4 Adiciona mensagem de sucesso à sessão

No template, este método é chamado através de um formulário com método **DELETE** :

```
<form action="{{ route('aluno.destroy', $aluno->id) }}"
method="POST">
    @csrf
```

Template: Lista de Alunos

```
<!-- resources/views/alunos/index.blade.php -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Lista de Alunos</title>
  <style>
    /* Estilos removidos para código mais compacto */
  </style>
</head>
<body>
  <div class="container">
    <h1>Alunos Cadastrados</h1>

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

    <a href="{{ route('aluno.create') }}" class="btn btn-
primary">Adicionar Novo Aluno</a>
```

```
<table>
  <thead>
```

Template de Listagem

Este template exibe todos os alunos cadastrados em uma tabela e oferece opções para gerenciá-los.

- ✓ Exibe mensagens de sucesso após operações
- ✓ Botão para adicionar novo aluno
- ✓ Tabela com todos os alunos cadastrados
- ✓ Botões de ação para cada aluno: Ver, Editar e Excluir
- ✓ Confirmação antes de excluir um aluno
- ✓ Classes CSS para estilização (estilos removidos para código mais compacto)

Template: Formulário de Criação

```
<!-- resources/views/alunos/create.blade.php -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Criar Novo Aluno</title>
    <style>
        /* Estilos removidos para código mais compacto */
    </style>
</head>
<body>
    <div class="container">
        <h1>Criar Novo Aluno</h1>

        @if ($errors->any())
            <div class="alert-danger">
                <ul>
                    @foreach ($errors->all() as $error)
                        <li>{{ $error }}</li>
                    @endforeach
                </ul>
            </div>
        @endif

        <form action="{{ route('aluno.store') }}" method="POST">
```

Formulário de Criação

Este template exibe um formulário para criar um novo aluno, com campos para nome, email e idade.

- ✓ Exibe mensagens de erro de validação quando ocorrem
- ✓ Usa `@csrf` para proteção contra ataques CSRF
- ✓ Mantém valores antigos com `{{ old("campo") }}` após falha de validação
- ✓ Envia dados para a rota `aluno.store` via método POST
- ✓ Inclui botão para cancelar e voltar à listagem
- ✓ Classes CSS para estilização (estilos removidos para código mais compacto)

Templates: Visualização e Edição

Template de Visualização (show.blade.php)

```
<!-- resources/views/alunos/show.blade.php -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Detalhes do Aluno</title>
    <style>
        /* Estilos removidos para código mais compacto */
    </style>
</head>
<body>
    <div class="container">
        <h1>Detalhes do Aluno</h1>

        <div class="detail-item">
            <label>ID:</label>
            <p>{{ $aluno->id }}</p>
        </div>
        <div class="detail-item">
            <label>Nome:</label>
            <p>{{ $aluno->nome }}</p>
        </div>
        <div class="detail-item">
            <label>Email:</label>
            <p>{{ $aluno->email }}</p>
        </div>
    </div>
</body>
</html>
```

Template de Edição (edit.blade.php)

```
<!-- resources/views/alunos/edit.blade.php -->
<!DOCTYPE html>
<html lang="pt-br">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Editar Aluno</title>
    <style>
        /* Estilos removidos para código mais compacto */
    </style>
</head>
<body>
    <div class="container">
        <h1>Editar Aluno</h1>

        @if ($errors->any())
            <div class="alert-danger">
                <ul>
                    @foreach ($errors->all() as $error)
                        <li>{{ $error }}</li>
                    @endforeach
                </ul>
            </div>
        @endif

        <form action="{{ route('aluno.update', $aluno->id) }}"
method="POST">
```