

**07**

# **DESENVOLVIMENTO DE APLICAÇÃO**

## **PARTE 1**



# Criando um CRUD em Laravel

03

## Criando um CRUD em Laravel

Depois de criar a pasta ProjLaravel e instalar o Laravel. Digite no CMD:

- **cd ProjLaravel**

- **code .**

Para abrir o projeto no VSCode.

INFO Running migrations.

```
0001_01_01_000000_create_users_table  
0001_01_01_000001_create_cache_table  
0001_01_01_000002_create_jobs_table .
```

```
C:\>cd ProjLaravel
```

```
C:\ProjLaravel>code .
```



# Criando um CRUD em Laravel

03

## Criando o banco de dados e tabelas



> vendor

⚙ .editorconfig

⚙ .env

📁 resources

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
DB_DATABASE=crud_laravel
DB_USERNAME=root
DB_PASSWORD=

SESSION_DRIVER=file
```

1 - Localize o arquivo .env

2 - Abra o Arquivo .env e edite:

3 - **DB\_CONNECTION=mysql**

4 - **DB\_DATABASE =bd\_laravel**

5 - **SESSION\_DRIVER=file**

6 - Depois salve as alterações.



# Criando um CRUD em Laravel

03

## Criando o banco de dados e tabelas



E sem seguida, em seu navegador de preferência, acesse localhost/phpmyadmin, clique em Novo e digite o nome do seu novo banco de dados. Ou você pode criar pelo próprio Laravel.

**Atenção:** Por padrão o Laravel utiliza o Agrupamento `utf8_unicode_ci`.



# Criando um CRUD em Laravel

03

## Criando uma migration



Com o banco de dados devidamente configurado chegou a hora de criarmos as primeiras tabelas. O Laravel já possui algumas migrations (arquivos de instrução para criação de tabelas) nativas, para acessá-las basta encontrar, a partir da raiz do projeto, o diretório **database/migrations**.

```
0001_01_01_000000_create_users_table  
0001_01_01_000001_create_cache_table  
0001_01_01_000002_create_jobs_table .
```

Por enquanto, vamos apagar todas e criar a nossa própria migration.



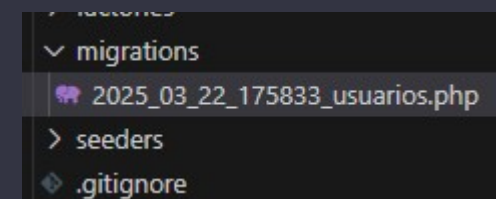
# Criando um CRUD em Laravel

03

## Criando uma migration

Você pode usar o `make:migration` comando Artisan para gerar uma migração de banco de dados. A nova migração será colocada no seu **database/migrations** diretório. Cada nome de arquivo de migração contém um timestamp que permite que o Laravel determine a ordem das migrações.

```
php artisan make:migration usuarios
```





# Criando um CRUD em Laravel

03

## Criando uma migration



O Laravel usará o nome da migração para tentar adivinhar o nome da tabela e se a migração criará ou não uma nova tabela. Se o Laravel for capaz de determinar o nome da tabela a partir do nome da migração, o Laravel preencherá previamente o arquivo de migração gerado com a tabela especificada. Caso contrário, você pode simplesmente especificar a tabela no arquivo de migração manualmente.



# Criando um CRUD em Laravel

03

## Criando uma migration

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    protected $table = "usuarios";

    public function up(): void
    {
        Schema::create('usuarios', function (Blueprint $table) {
            $table->id();
            $table->string('nome');
            $table->string('email');
            $table->string('senha');
            $table->timestamps();
        });
    }

    public function down()
    {
        Schema::table('usuarios', function (Blueprint $table) {
            $table->dropTimestamps();
        });
    }
};
```

**1** - Na migration **usuarios**, após criada, vamos editar com os campos id, nome, email e senha.

**2** - No seu terminal, na raiz do projeto, digite o seguinte comando:

**php artisan migrate**





# Criando um CRUD em Laravel

03

## Criando uma migration

No Laravel, a propriedade **protected \$table** em um modelo Eloquent é usada para especificar o nome da tabela do banco de dados associada a esse modelo.

```
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    protected $table = "usuarios";

    public function up(): void
    {
        Schema::create('usuarios', function (Blueprint $table) {
            $table->id();
            $table->string('nome');
            $table->string('email');
            $table->string('senha');
            $table->timestamps();
        });
    }

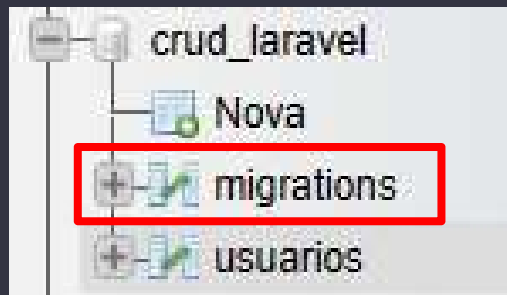
    public function down()
    {
        Schema::table('usuarios', function (Blueprint $table) {
            $table->dropTimestamps();
        });
    }
};
```



# Criando um CRUD em Laravel

03

## Criando uma migration



#	Nome	Tipo
<input type="checkbox"/> 1	id 🔑	bigint(20)
<input type="checkbox"/> 2	nome	varchar(255)
<input type="checkbox"/> 3	email 📧	varchar(255)
<input type="checkbox"/> 4	senha	varchar(255)

Perceba que uma tabela migrations foi criada. É um recurso nativo do Laravel que serve para registrar quais **migrations** já foram executadas para prevenir a tentativa de criação de uma tabela já existente, afinal de contas no dia a dia nós criaremos várias migrações que não serão necessariamente executadas ao mesmo tempo.



# Criando um CRUD em Laravel

03

## Criando uma migration

- **php artisan migrate:status** - ver quais migrações foram executadas até agora.
- php artisan migrate:rollback** - Para reverter a última operação de migração.
- php artisan migrate:reset** - Comando reverterá todas as migrações do seu aplicativo.
- php artisan migrate:refresh** - Este comando efetivamente recria todo o seu banco de dados.
- php artisan migrate:fresh** - comando apenas descarta tabelas da conexão de banco de dados padrão. No entanto, você pode usar a `-database` opção para especificar a conexão de banco de dados que deve ser migrada.



# Criando um CRUD em Laravel

03

## Criando uma model

- O Laravel inclui o Eloquent, um mapeador objeto-relacional (ORM) que torna agradável interagir com seu banco de dados. Ao usar o Eloquent, cada tabela do banco de dados tem um "Modelo" correspondente que é usado para interagir com essa tabela. Além de recuperar registros da tabela do banco de dados, os modelos do Eloquent permitem que você insira, atualize e exclua registros da tabela também. Você pode usar o **make:model** comando **Artisan** para gerar um novo modelo:

**php artisan make:model Usuarios**



# Criando um CRUD em Laravel

03

## Criando uma model



Se você quiser gerar uma migração de banco de dados ao mesmo tempo gerar o modelo, você pode usar a opção:

**--migration** ou **:-m**

```
php artisan make:model nomemodel --migration
```

```
php artisan make:model nomemodel -m
```



# Criando um CRUD em Laravel

03

## Criando uma model



- Em PHP, assim como em muitas outras linguagens orientadas a objetos, as classes são geralmente nomeadas com letras maiúsculas no início. Isso ajuda a distinguir classes de variáveis e funções, que geralmente começam com letras minúsculas.
- O Laravel segue essas convenções para manter a consistência e a legibilidade do código.



# Criando um CRUD em Laravel

03

## Criando uma model



```
app > Models > Usuarios.php > ...
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Model;
6
7  class Usuarios extends Model
8  {
9      protected $fillable = ['nome', 'email', 'senha'];
10 }
11 |
```

Em Models > **Usuarios.php**, edite os códigos como a imagem ao lado.

**protected \$fillable** no Laravel é uma medida de segurança que define quais colunas de um banco de dados podem ser preenchidas ao usar métodos como `create()` ou `update()`. Isso protege seus modelos contra a atribuição de atributos não intencionais ou maliciosos.



# Criando um CRUD em Laravel

03

## Criando um blade



Blade é o mecanismo de template simples, mas poderoso, que está incluso no Laravel. Ao contrário de alguns mecanismos de template PHP, o Blade não o restringe de usar código PHP simples em seus templates. Na verdade, todos os templates Blade são compilados em código PHP simples e armazenados em cache até serem modificados, o que significa que o Blade adiciona essencialmente zero overhead ao seu aplicativo. Os arquivos de template **Blade** usam a `.blade.php` extensão de arquivo e são normalmente armazenados no diretório (**resources/views**).





# Criando um CRUD em Laravel

03

1

## Criando um blade

views \ Usuarios  
create.blade.php

2

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Cadastro de Usuarios!</title>
</head>
<body>
  <h1>Cadastrar um novo usuário</h1>
  <br>
  <label for="">Nome</label>
  <input type="text" name="nome" required><br><br>
  <label for="">E-mail</label>
  <input type="email" name="email" required><br><br>
  <label for="">Senha</label>
  <input type="password" name="senha" required><br><br>
  <button>Cadastrar</button>
</form>
</div>
</div>
</body>
</html>
```

1 – Em **resources/views**, crie uma pasta chamada: **Usuarios** e crie um arquivo chamado **create.blade.php**.

2 – Edite o arquivo crie o formulário para receber os dados de entrada dos usuários.



# Criando um CRUD em Laravel

03

## Criando um controller



Em vez de definir toda a sua lógica de tratamento de requisições como fechamentos em seus arquivos de rota, você pode desejar organizar esse comportamento usando classes "**controller**". Os controladores podem agrupar lógicas de tratamento de requisições relacionadas em uma única classe. Por exemplo, uma UserController classe pode manipular todas as requisições recebidas relacionadas a usuários, incluindo mostrar, criar, atualizar e excluir usuários. Por padrão, os controladores são armazenados no diretório (**app/Http/Controllers**).



# Criando um CRUD em Laravel

03

## Criando um controller



Para gerar rapidamente um novo controlador, você pode executar o **make:controller** comando **Artisan**. Por padrão, todos os controladores para sua aplicação são armazenados no diretório (app/Http/Controllers). Vamos criar um **UsuariosController** para podemos acessar o formulário já criado.

**php artisan make:controller UsuariosController**

Os Controllers recebem as requisições HTTP (GET, POST, PUT, DELETE, etc.) e processam os dados enviados pelo usuário.



# Criando um CRUD em Laravel

03

## Criando um controller

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;

class ExemploController extends Controller
{
    //
}
```

No Laravel, os Controllers desempenham um papel fundamental na arquitetura MVC (Model-View-Controller) da aplicação. Eles atuam como intermediários entre as rotas (URLs) e a lógica de negócios da sua aplicação, facilitando a organização e a manutenção do código.



# Criando um CRUD em Laravel

03

## Criando um controller

```
<?php

namespace App\Http\Controllers;

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

class UsuariosController extends Controller
{
    public function create()
    {
        return view('Usuarios.create');
    }

    public function store(Request $request)
    {
        $usuario = new Usuarios();
        $usuario->nome = $request->input('nome');
        $usuario->email = $request->input('email');
        $usuario->senha = bcrypt($request->input('senha')); // Criptografando a senha
        $usuario->save();

        return redirect()->route('usuarios.read')->with('success',
            'Usuário registrado com sucesso!');
    }
}
```

**create():** Mostra o formulário para adicionar um usuário.

**store():** Salva os dados do formulário no banco de dados, criptografa a senha e informa o sucesso.

Modelo "**Usuarios**": Facilita a interação com o banco de dados.

**bcrypt():** Protege as senhas armazenadas.



# Criando um CRUD em Laravel

03

## Criando uma Routing

- As rotas mais básicas do Laravel aceitam um URI e um fechamento, fornecendo um método muito simples e expressivo de definir rotas e comportamento sem arquivos de configuração de roteamento complicados.

Todas as rotas do Laravel são definidas nos seus arquivos de rota, que estão localizados no diretório (**routes**). Esses arquivos são carregados automaticamente pelo Laravel usando a configuração especificada no arquivo (**bootstrap/app.php**) do seu aplicativo. O arquivo (**routes/web.php**) define rotas que são para sua interface web. Essas rotas são atribuídas ao web grupo middleware , que fornece recursos como estado de sessão e proteção **CSRF**.



# Criando um CRUD em Laravel

03

## Criando uma Routing

```
<?php

use App\Http\Controllers\UsuariosController;
use App\Models\Usuarios;
use Illuminate\Support\Facades\Route;

Route::get('/', function () {
    return view('/Usuarios/create');
});

Route::get('/Usuarios/create', [UsuariosController::class, 'create']);
Route::post('/Usuarios/create', [UsuariosController::class, 'store'])->name('registrar_usuarios');
```

- **GET /Usuarios/create:** Mostra o formulário de novo usuário.
- **POST /Usuarios/create:** Salva o novo usuário (dados do formulário).
- **->name('registrar\_usuarios'):** Dá um nome à rota POST para facilitar o uso.



# Criando um CRUD em Laravel

03

## Criando uma Routing

```
<form action="{{route('registrar_usuarios')}}" method="POST">
  @csrf
  <label for="">Nome</label>
  <input type="text" name="nome" required><br><br>
  <label for="">E-mail</label>
  <input type="email" name="email" required><br><br>
  <label for="">Senha</label>
  <input type="password" name="senha" required><br><br>
  <button>Cadastrar</button>
</form>
```

Volte para o arquivo `create.blade.php` e add na form `{{ route('registrar_usuarios') }}` e o `method="POST"`.

Adicione também `@csrf`.

Este formulário envia dados via método POST para a rota nomeada "**registrar\_usuarios**", que está configurada para processar os dados do formulário e salvar um novo usuário. O uso de `@csrf` protege o formulário contra ataques CSRF.





# Criando um CRUD em Laravel

03

## Criando uma Routing

```
<form action="{{route('registrar_usuarios')}}" method="POST">
  @csrf
  <label for="">Nome</label>
  <input type="text" name="nome" required><br><br>
  <label for="">E-mail</label>
  <input type="email" name="email" required><br><br>
  <label for="">Senha</label>
  <input type="password" name="senha" required><br><br>
  <button>Cadastrar</button>
</form>
```

Volte para o arquivo `create.blade.php` e add na form `{{ route('registrar_usuarios') }}` e o `method="POST"`.

Adicione também `@csrf`.

Este formulário envia dados via método POST para a rota nomeada "**registrar\_usuarios**", que está configurada para processar os dados do formulário e salvar um novo usuário. O uso de `@csrf` protege o formulário contra ataques CSRF.



# Criando um CRUD em Laravel

03

## Create (Criação de registro)



← → ↻ ⓘ 127.0.0.1:8000

Tipos de Jogos Digi... 1 Contra 100 Prova Online

### Cadastrar um novo usuário

Nome

E-mail

Senha

Chegou a hora de testar, vamos ligar o servidor do Laravel e acessar pelo navegador para ver nosso formulário carregando. Para isso vá no seu terminal, na pasta raiz do projeto, e digite o seguinte comando:

**php artisan serve**



# Criando um CRUD em Laravel

03

## Create (Criação de registro)

Vamos verificar lá no banco de dados se o produto foi realmente criado! Acesse localhost/phpmyadmin, entre no banco que você criou e veja e na tabela produtos está presente o registro que acabamos de criar.

id	nome	email	senha	created_at	updated_at
2	Felipe Pimentel	felipe@gmail.com	\$2y\$12\$nCZufRYdoEZFegDRla7cheRRUw517nNCNaPTJyYlyUc...	2025-03-22 19:27:04	2025-03-22 19:27:04
3	Marcia Silva	marciaprof2023@gmail.com	\$2y\$12\$bNRNugyc236rV4IHXY0n.OyG.ddvcknEZ9WYwSbZS59...	2025-03-22 19:30:13	2025-03-22 19:30:13
4	Laura Silva	laura@gmail.com	\$2y\$12\$GAUIGx8uYrzD/teNdKEwI.bbK6oA8tXhyj0bNrYRr2a...	2025-03-22 19:30:28	2025-03-22 19:30:28

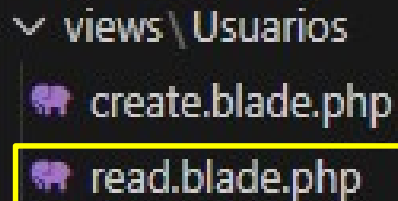


# Criando um CRUD em Laravel

03

## Read ( blade )

Vá em views\Usuarios e crie um arquivo chamado **read.blade.php**.



```
views \ Usuarios
├── create.blade.php
└── read.blade.php
```

read.blade.php é uma view Blade usada para exibir dados dinâmicos para o usuário. Ele pode conter HTML, diretivas Blade e lógica PHP para formatar e apresentar os dados de forma clara e organizada.



# Criando um CRUD em Laravel

03

## Read ( Controller )



Dentro deste arquivo UsuariosController.php, você adicionará a definição da função **read()**. Este método **read()** busca todos os registros de usuários do banco de dados e os passa para a view **read.blade.php** para serem exibidos.

```
public function read()
{
    $usuarios = Usuarios::all(); // Busca todos os usuários
    return view('Usuarios.read', compact('usuarios'));
}
```



# Criando um CRUD em Laravel

03

## Read ( web.php )

```
Route::get('/Usuarios/create', [UsuariosController::class, 'create']);  
Route::post('/Usuarios/create', [UsuariosController::class, 'store'])->name('registrar_usuarios');  
  
// Definindo a rota para a listagem de usuários  
Route::get('/Usuarios/read', [UsuariosController::class, 'read'])->name('usuarios.read');
```

Dentro do arquivo **web.php**, você adicionará uma rota.

Essa rota define que quando um usuário acessar a url **/Usuarios/read** via get, o metodo read do UsuariosController será executado, e que essa rota tem o nome de **usuarios.read**.



# Criando um CRUD em Laravel

03

## Read ( read.blade.php )

```
<body>
<div class="container">
  <h2 class="mt-5">Lista de Usuários</h2>

  @if(session('success'))
    <div class="alert alert-success alert-dismissible fade show" role="alert">
      <strong>Sucesso!</strong> {{ session('success') }}
      <button type="button" class="close" data-dismiss="alert" aria-label="Close">
        <span aria-hidden="true">&times;</span>
      </button>
    </div>
  @endif

  <table class="table table-bordered mt-4">
    <thead>
      <tr>
        <th>ID</th>
        <th>Nome</th>
        <th>Email</th>
        <th>Ações</th>
      </tr>
    </thead>
```



# Criando um CRUD em Laravel

03

## Read ( read.blade.php )

```
<tbody>
  @foreach($usuarios as $usuario)
    <tr>
      <td>{{ $usuario->id }}</td>
      <td>{{ $usuario->nome }}</td>
      <td>{{ $usuario->email }}</td>
      <td>
        <!-- Ações: editar e excluir -->
        <a href="#" class="btn btn-info btn-sm">Editar</a>
        <a href="#" class="btn btn-danger btn-sm">Excluir</a>
      </td>
    </tr>
  @endforeach
</tbody>
</table>
</div>
```





# Criando um CRUD em Laravel

03

## Read ( Resultado )

### Lista de Usuários

Sucesso! Usuário registrado com sucesso!

ID	Nome	Email	Ações
2	Felipe Pimentel	felipe@gmail.com	<a href="#">Editar</a> <a href="#">Excluir</a>
3	Marcia Silva	marciaprof2023@gmail.com	<a href="#">Editar</a> <a href="#">Excluir</a>
4	Laura Silva	laura@gmail.com	<a href="#">Editar</a> <a href="#">Excluir</a>
5	Dioneide Sales	dioneide@gmail.com	<a href="#">Editar</a> <a href="#">Excluir</a>
6	Grazi	grazi@gmail.com	<a href="#">Editar</a> <a href="#">Excluir</a>

**Até a  
Próxima  
Aula!**

