

Referência Rápida de Laravel

Artisan, roteamento, Eloquent, Blade, middleware, auth

Artisan

Comandos Comuns

<code>php artisan serve</code>	Iniciar servidor de desenvolvimento
<code>php artisan make:model Name -m</code>	Criar model com migration
<code>php artisan make:controller NameController</code>	Criar classe controller
<code>php artisan make:middleware Name</code>	Criar classe middleware
<code>php artisan migrate</code>	Executar migrations pendentes
<code>php artisan migrate:rollback</code>	Reverter último lote de migration
<code>php artisan db:seed</code>	Executar seeders do banco de dados
<code>php artisan tinker</code>	REPL interativo para a aplicação
<code>php artisan route:list</code>	Listar todas as rotas registradas
<code>php artisan cache:clear</code>	Limpar o cache da aplicação
<code>php artisan config:clear</code>	Limpar config em cache
<code>php artisan queue:work</code>	Iniciar processamento de jobs na fila

Roteamento

Rotas Básicas

```
Route::get('/users', [UserController::class, 'index']);
Route::post('/users', [UserController::class, 'store']);
Route::put('/users/{id}', [UserController::class, 'update']);
Route::delete('/users/{id}', [UserController::class, 'destroy']);
```

Parâmetros e Grupos de Rota

```
Route::get('/user/{id}', function (int $id) {
    return User::findOrFail($id);
});

Route::prefix('api')->middleware('auth')->group(function () {
    Route::get('/profile', [ProfileController::class, 'show']);
});
```

Recursos de Rota

<code>->name('route.name')</code>	Rota nomeada para geração de URL
<code>->where('id', '[0-9]+')</code>	Restrição regex no parâmetro
<code>Route::resource()</code>	Rotas de recurso RESTful (7 rotas)
<code>Route::apiResource()</code>	Recurso de API (sem views create/edit)
<code>Route::fallback()</code>	Captura rotas sem correspondência

Controllers

Resource Controller

```
class PostController extends Controller {
    public function index() {
        return view('posts.index', ['posts' => Post::all()]);
    }

    public function store(Request $request) {
        $validated = $request->validate(['title' => 'required|
max:255']);
        Post::create($validated);
        return redirect()->route('posts.index');
    }
}
```

Métodos de Resource

<code>index()</code>	GET /resource -- listar todos
<code>create()</code>	GET /resource/create -- exibir formulário
<code>store()</code>	POST /resource -- salvar novo
<code>show(\$id)</code>	GET /resource/{id} -- exibir um
<code>edit(\$id)</code>	GET /resource/{id}/edit -- formulário de edição
<code>update(\$id)</code>	PUT /resource/{id} -- atualizar
<code>destroy(\$id)</code>	DELETE /resource/{id} -- remover

Templates Blade

Layout e Seções

```
{{!-- layouts/app.blade.php --}}
<html><body>
    @yield('content')
</body></html>

{{!-- pages/home.blade.php --}}
@extends('layouts.app')
@section('content')
    <h1>Home</h1>
@endsection
```

Diretivas

<code>{{ \$var }}</code>	Echo com escape HTML
<code>{!! \$html !!}</code>	Echo bruto (sem escape)
<code>@if / @elseif / @else</code>	Blocos condicionais
<code>@foreach (\$items as \$item)</code>	Iterar sobre coleção
<code>@forelse / @empty</code>	Loop com fallback para vazio
<code>@include('partial')</code>	Incluir outra view Blade
<code>@component / @slot</code>	Componentes Blade reutilizáveis
<code>@csrf</code>	Campo hidden com token CSRF
<code>@auth / @guest</code>	Verificar status de autenticação
<code>@error('field')</code>	Exibir erro de validação

Eloquent ORM

Básico de Model

```
class Post extends Model {
    protected $fillable = ['title', 'body', 'user_id'];

    public function user() {
        return $this->belongsTo(User::class);
    }
}
```

Consultando

```
Post::all(); // all records
Post::find(1); // by primary key
Post::where('status', 'published')->get();
Post::where('views', '>', 100)->orderBy('created_at', 'desc')->first();
```

Operações CRUD

```
$post = Post::create(['title' => 'New', 'body' => '...']);
$post->update(['title' => 'Updated']);
$post->delete();
Post::destroy([1, 2, 3]); // delete by IDs
```

Relacionamentos

hasOne	Um-para-um (User -> Phone)
hasMany	Um-para-muitos (Post -> Comments)
belongsToMany	Inverso de hasOne/hasMany
belongsToMany	Muitos-para-muitos com tabela pivot
hasManyThrough	Has-many via model intermediário

Migrations

Criando Tabelas

```
Schema::create('posts', function (Blueprint $table) {
    $table->id();
    $table->foreignId('user_id')->constrained()->cascadeOnDelete();
    $table->string('title');
    $table->text('body')->nullable();
    $table->timestamps();
});
```

Tipos de Coluna

<code>\$table->id()</code>	Chave primária BIGINT auto-incremento
<code>\$table->string('col', 100)</code>	VARCHAR com comprimento opcional
<code>\$table->text('col')</code>	Coluna TEXT
<code>\$table->integer('col')</code>	Coluna INTEGER
<code>\$table->boolean('col')</code>	Coluna BOOLEAN
<code>\$table->json('col')</code>	Coluna JSON
<code>\$table->timestamp('col')</code>	Coluna TIMESTAMP
<code>\$table->timestamps()</code>	created_at e updated_at
<code>\$table->softDeletes()</code>	deleted_at para exclusão suave

Middleware

Middleware Personalizado

```
class EnsureAdmin {
    public function handle(Request $request, Closure $next) {
        if (!$request->user()->is_admin) {
            abort(403);
        }
        return $next($request);
    }
}
```

Registrando e Usando

```
// bootstrap/app.php
->withMiddleware(function (Middleware $middleware) {
    $middleware->alias(['admin' => EnsureAdmin::class]);
});

// In routes
Route::get('/admin', fn() => '...')->middleware('admin');
```

Middleware Integrado

auth	Exige autenticação
guest	Redireciona se autenticado
throttle:60,1	Limite de taxa (60 req/min)
verified	Exige verificação de e-mail
signed	Valida URL assinada

Referência Rápida de Laravel

Autenticação

Helpers de Auth

```
Auth::check(); // is user logged in?
Auth::user(); // current User model
Auth::id(); // current user ID
Auth::attempt(['email' => $e, 'password' => $p]);
Auth::logout();
```

Starter Kits

Laravel Breeze	Scaffolding mínimo de auth (Blade ou Inertia)
Laravel Jetstream	Completo (times, 2FA, tokens de API)
Sanctum	Autenticação por token para SPA / mobile
Passport	Implementação completa de servidor OAuth2

Protegendo Rotas

```
Route::middleware('auth')->group(function () {
    Route::get('/dashboard', [DashController::class, 'index']);
});
```

Validação

Validação no Controller

```
$validated = $request->validate([
    'title' => 'required|string|max:255',
    'email' => 'required|email|unique:users',
    'age' => 'nullable|integer|min:0',
]);
```

Form Request

```
class StorePostRequest extends FormRequest {
    public function rules(): array {
        return [
            'title' => 'required|max:255',
            'body' => 'required|min:10',
        ];
    }
}
```

Regras Comuns

required	Campo obrigatório e não vazio
string integer boolean	Validação de tipo
min:N max:N	Comprimento ou valor mínimo/máximo
email	Formato de e-mail válido
unique:table,column	Deve ser único na tabela do banco
exists:table,column	Deve existir na tabela do banco
in:a,b,c	Deve ser um dos valores listados
confirmed	Requer campo _confirmation correspondente
date after:date	Validação de data

Padrões Comuns

Resposta de API

```
return response()->json(['data' => $users], 200);
return response()->json(['error' => 'Not found'], 404);
```

Ambiente e Configuração

```
env('APP_KEY'); // read .env value
config('app.name'); // read config value
config(['app.debug' => true]); // set at runtime
```

Helpers Úteis

route('name', \$params)	Gerar URL para rota nomeada
redirect()->route('name')	Redirecionar para rota nomeada
back()->withErrors()	Redirecionar de volta com erros de validação
abort(404)	Lançar exceção HTTP
collect(\$array)	Criar uma Collection a partir de array
now()	Datetime Carbon atual
cache()->remember()	Armazenar valor em cache com TTL