REFERÊNCIA RÁPIDA DE LARAVEL

Artisan, roteamento, Eloquent, Blade, middleware, auth

Artisan

Comandos Comuns

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

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

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

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

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

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