

# Referencia Rápida de Laravel

Artisan, rutas, Eloquent, Blade, middleware, autenticación

## Artisan

### Comandos comunes

<code>php artisan serve</code>	Iniciar servidor de desarrollo
<code>php artisan make:model Name -m</code>	Crear modelo con migración
<code>php artisan make:controller NameController</code>	Crear clase de controlador
<code>php artisan make:middleware Name</code>	Crear clase de middleware
<code>php artisan migrate</code>	Ejecutar migraciones pendientes
<code>php artisan migrate:rollback</code>	Revertir el último lote de migraciones
<code>php artisan db:seed</code>	Ejecutar seeders de base de datos
<code>php artisan tinker</code>	REPL interactivo para la aplicación
<code>php artisan route:list</code>	Listar todas las rutas registradas
<code>php artisan cache:clear</code>	Limpiar la caché de la aplicación
<code>php artisan config:clear</code>	Limpiar la configuración en caché
<code>php artisan queue:work</code>	Iniciar el procesamiento de jobs en cola

## Rutas

### Rutas 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']);
```

### Parámetros y grupos de rutas

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

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

### Características de rutas

<code>-&gt;name('route.name')</code>	Ruta nombrada para generación de URL
<code>-&gt;where('id', '[0-9]+')</code>	Restricción de regex en el parámetro
<code>Route::resource()</code>	Rutas de recurso RESTful (7 rutas)
<code>Route::apiResource()</code>	Recurso API (sin vistas create/edit)
<code>Route::fallback()</code>	Captura todas las rutas no coincidentes

## Controladores

### Controlador de recurso

```
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 recurso

<code>index()</code>	GET /resource -- listar todos
<code>create()</code>	GET /resource/create -- mostrar formulario
<code>store()</code>	POST /resource -- guardar nuevo
<code>show(\$id)</code>	GET /resource/{id} -- mostrar uno
<code>edit(\$id)</code>	GET /resource/{id}/edit -- formulario de edición
<code>update(\$id)</code>	PUT /resource/{id} -- actualizar
<code>destroy(\$id)</code>	DELETE /resource/{id} -- eliminar

## Plantillas Blade

### Layout y secciones

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

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

### Directivas

<code>{{ \$var }}</code>	Imprimir con escape HTML
<code>{!! \$html !!}</code>	Imprimir sin escape (raw)
<code>@if / @elseif / @else</code>	Bloques condicionales
<code>@foreach (\$items as \$item)</code>	Iterar sobre colección
<code>@forelse / @empty</code>	Bucle con alternativa vacía
<code>@include('partial')</code>	Incluir otra vista Blade
<code>@component / @slot</code>	Componentes Blade reutilizables
<code>@csrf</code>	Campo oculto con token CSRF
<code>@auth / @guest</code>	Verificar estado de autenticación
<code>@error('field')</code>	Mostrar error de validación

## Eloquent ORM

### Fundamentos del modelo

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

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

### Consultas

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

## Operaciones CRUD

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

## Relaciones

<code>hasOne</code>	Uno a uno (Usuario -> Teléfono)
<code>hasMany</code>	Uno a muchos (Post -> Comentarios)
<code>belongsTo</code>	Inverso de hasOne/hasMany
<code>belongsToMany</code>	Muchos a muchos con tabla pivot
<code>hasManyThrough</code>	Has-many a través de modelo intermedio

## Migraciones

### Crear tablas

```
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 columna

<code>\$table-&gt;id()</code>	Clave primaria BIGINT autoincremental
<code>\$table-&gt;string('col', 100)</code>	VARCHAR con longitud opcional
<code>\$table-&gt;text('col')</code>	Columna TEXT
<code>\$table-&gt;integer('col')</code>	Columna INTEGER
<code>\$table-&gt;boolean('col')</code>	Columna BOOLEAN
<code>\$table-&gt;json('col')</code>	Columna JSON
<code>\$table-&gt;timestamp('col')</code>	Columna TIMESTAMP
<code>\$table-&gt;timestamps()</code>	created_at y updated_at
<code>\$table-&gt;softDeletes()</code>	deleted_at para eliminación suave

## Middleware

### Middleware personalizado

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

### Registrar y usar

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

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

### Middleware integrado

<code>auth</code>	Requerir autenticación
<code>guest</code>	Redirigir si está autenticado
<code>throttle:60,1</code>	Límite de velocidad (60 req/min)
<code>verified</code>	Requerir verificación de email
<code>signed</code>	Validar URL firmada

# Referencia Rápida de Laravel

## Autenticación

### 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();
```

### Kits de inicio

<b>Laravel Breeze</b>	Autenticación mínima (Blade o Inertia)
<b>Laravel Jetstream</b>	Completo (equipos, 2FA, tokens API)
<b>Sanctum</b>	Autenticación por token para SPA / móvil
<b>Passport</b>	Implementación completa de servidor OAuth2

### Proteger rutas

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

## Validación

### Validación en el controlador

```
$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',
        ];
    }
}
```

### Reglas comunes

<b>required</b>	El campo debe estar presente y no vacío
<b>string   integer   boolean</b>	Validación de tipo
<b>min:N   max:N</b>	Longitud o valor mínimo/máximo
<b>email</b>	Formato de email válido
<b>unique:table,column</b>	Debe ser único en la tabla de BD
<b>exists:table,column</b>	Debe existir en la tabla de BD
<b>in:a,b,c</b>	Debe ser uno de los valores listados
<b>confirmed</b>	Requiere campo <code>_confirmation</code> coincidente
<b>date   after:date</b>	Validación de fecha

### Patrones comunes

#### Respuesta de API

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

### Entorno y configuración

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

### Helpers útiles

<b>route('name', \$params)</b>	Generar URL para ruta nombrada
<b>redirect()-&gt;route('name')</b>	Redirigir a ruta nombrada
<b>back()-&gt;withErrors()</b>	Redirigir atrás con errores de validación
<b>abort(404)</b>	Lanzar excepción HTTP
<b>collect(\$array)</b>	Crear una Collection desde un array
<b>now()</b>	Fecha y hora actual de Carbon
<b>cache()-&gt;remember()</b>	Cachear un valor con TTL