REFERENCIA RÁPIDA DE LARAVEL

Artisan, rutas, Eloquent, Blade, middleware, autenticación

Artisan

Comandos comunes

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

`->name('route.name')`	Ruta nombrada para generación de URL
`->where('id', '[0-9]+')`	Restricción de regex en el parámetro
`Route::resource()`	Rutas de recurso RESTful (7 rutas)
`Route::apiResource()`	Recurso API (sin vistas create/edit)
`Route::fallback()`	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

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

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

`hasOne`	Uno a uno (Usuario -> Teléfono)
`hasMany`	Uno a muchos (Post -> Comentarios)
`belongsTo`	Inverso de hasOne/hasMany
`belongsToMany`	Muchos a muchos con tabla pivot
`hasManyThrough`	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

`$table->id()`	Clave primaria BIGINT autoincremental
`$table->string('col', 100)`	VARCHAR con longitud opcional
`$table->text('col')`	Columna TEXT
`$table->integer('col')`	Columna INTEGER
`$table->boolean('col')`	Columna BOOLEAN
`$table->json('col')`	Columna JSON
`$table->timestamp('col')`	Columna TIMESTAMP
`$table->timestamps()`	created_at y updated_at
`$table->softDeletes()`	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

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

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

`Laravel Breeze`	Autenticación mínima (Blade o Inertia)
`Laravel Jetstream`	Completo (equipos, 2FA, tokens API)
`Sanctum`	Autenticación por token para SPA / móvil
`Passport`	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

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

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