🚀 Minimal APIs + Laravel — محاضرة عملية

> **بتشرح ببساطة.. من الأول للآخر. مستعد؟**

📝 جدول المحتويات

🎯 مقدمة خفيفة

API يعني إيه؟ بسرعة

فكّر بنفسك في المطعم:

- أنت (العميل) = Client
- **الويتر** (واسطة التواصل) = API
- **الشيف/المطبخ** = الـ Server

```
أنت (طالب الأكل)
    ↓ (طلب)
الويتر (API)
    ↓ (يروح المطبخ)
الشيف (Server)
    ↓ (يرجع الأكل)
الويتر (API)
    ↓ (يجيب الأكل)
أنت (تاكل) ✅
```

API: هي الطريقة اللي بتتواصل بيها الـ frontend مع الـ backend **بسرعة**.

ليه "Minimal" أصلاً؟

كنت بتعمل Laravel API في الأول، لما تبني:

```
// ❌ كويسة بس معقدة شوية
Route::apiResource('posts', PostController::class);
// محتاج controller، models، resources، requests... كل ده في ملفات منفصلة
```

بسيط جداً، ليه كل هالتعقيد؟ endpoint صغير أو الـ project لكن لما يكون الـ 🤔

فاخترعوا Minimal APIs:

```
// ✅ أبسط و أسرع
Route::get('/posts', fn() => Post::all());
Route::post('/posts', fn(Request $r) => Post::create($r->validated()));
```

الفكرة: كتابة direct routes مع Closures (أو Invokable Controllers) بدون كل هالحاجات.

الأساسيات اللي لازم تعرفها 🤓

1) Endpoint؟ يعني إيه

Endpoint = عنوان محدد في الـ API بتطلب منه حاجة معينة.

```
https://myapp.com/api/posts       ← Endpoint للـ Posts
https://myapp.com/api/users       ← Endpoint للـ Users
https://myapp.com/api/posts/1     ← Endpoint لـ Post معين
```

2) HTTP Methods (الطرق الأساسية)

Method	المعنى	مثال	الاستخدام
GET	اجلب بيانات	`GET /posts`	أنا بدي أقرأ
POST	انشئ بيانات جديدة	`POST /posts`	أنا بدي أضيف حاجة
PUT	استبدل كل حاجة	`PUT /posts/1`	بدي أغيّر كل الـ record
PATCH	عدّل جزء بس	`PATCH /posts/1`	بدي أغيّر field واحد
DELETE	احذف	`DELETE /posts/1`	بدي احذف الحاجة

مثال بسيط جداً:

```
// في routes/api.php

// GET: اجلب كل الـ posts
Route::get('/posts', fn() => Post::all());

// POST: انشئ post جديد
Route::post('/posts', fn(Request $r) => Post::create($r->validated()));

// PATCH: عدّل post معين
Route::patch('/posts/{id}', fn($id, Request $r) =>
    Post::find($id)->update($r->validated())
);
```

```php
// DELETE: احذف post
Route::delete('/posts/{id}', fn($id) => Post::find($id)->delete());
```

3) Status Codes الأساسية

الكود	المعنى	متى تستخدمه
200	✅ نجح + فيه بيانات	لما تجلب بيانات
201	✅ تم الإنشاء	لما تنشئ resource جديد
400	❌ طلبك غلط	validation failed
401	❌ (غير مصرح (لازم تسجل	غلط token/password ما فيش
403	❌ (ممنوع (حتى لو مسجل	مش بتاعك post
404	❌ ما حصلت الحاجة	ما موجود post
422	❌ بيانات غير صحيحة	validation errors
500	❌ الخادم كسر	في الكود bug

4) Request و Response (شكلهم عامل إزاي)

Request (لما تبعت **مثال** POST لإنشاء post):

```http
{
  "title": "Laravel Basics",
  "content": "يلا نتعلم Laravel",
  "author_id": 1
}
```

Response (الشيء تم إنشاؤه **بتاع 201**):

```http
{
  "id": 42,
  "title": "Laravel Basics",
  "content": "يلا نتعلم Laravel",
  "author_id": 1,
  "created_at": "2026-01-21T10:30:00Z"
}
```

Response (خطأ في البيانات **بتاع 400**):

```http
{
  "message": "The given data was invalid.",
  "errors": {
```

```
    "title": ["The title field is required"],
    "content": ["The content must be at least 10 characters"]
  }
}
```

5) Naming Conventions (تسميات صح)

```
// ✅ صحيح
GET    /api/posts           // كل الـ posts
POST   /api/posts           // انشئ post
GET    /api/posts/{id}      // post معين
PATCH  /api/posts/{id}      // عدّل post
DELETE /api/posts/{id}      // احذف post

// ❌ غلط
GET    /api/GetAllPosts       // في الاسم بالفعل GET.. معناش حاجة
POST   /api/CreatePost        // CreatePost في الكود بتاع HTTP
GET    /api/post/details/1    // تعقيد بلا داعي
```

💻 Minimal APIs في Laravel

مثال عملي: Todo API

نبني API صغير للـ Todos. بسيط وكويس:

الخطوة 1: Model + Migration

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

```
// database/migrations/xxxx_create_todos_table.php
Schema::create('todos', function (Blueprint $table) {
    $table->id();
    $table->string('title');
    $table->text('description')->nullable();
    $table->boolean('completed')->default(false);
    $table->timestamps();
});
```

الخطوة 2: Routes في `routes/api.php`

```
<?php

use App\Models\Todo;
```

```php
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

// GET: اجلب كل الـ todos
Route::get('/todos', function () {
    return response()->json([
        'success' => true,
        'data' => Todo::latest()->get(),
    ]);
});

// POST: انشئ todo جديد
Route::post('/todos', function (Request $request) {
    $validated = $request->validate([
        'title' => 'required|string|max:255',
        'description' => 'nullable|string',
    ]);

    $todo = Todo::create($validated);

    return response()->json([
        'success' => true,
        'message' => 'Todo created successfully',
        'data' => $todo,
    ], 201);
});

// GET: اجلب todo محددة
Route::get('/todos/{id}', function ($id) {
    $todo = Todo::find($id);

    if (!$todo) {
        return response()->json([
            'success' => false,
            'message' => 'Todo not found',
        ], 404);
    }

    return response()->json([
        'success' => true,
        'data' => $todo,
    ]);
});

// PATCH: عدّل todo
Route::patch('/todos/{id}', function ($id, Request $request) {
    $todo = Todo::find($id);

    if (!$todo) {
        return response()->json([
            'success' => false,
            'message' => 'Todo not found',
        ], 404);
    }
```

```
    $validated = $request->validate([
        'title' => 'sometimes|string|max:255',
        'description' => 'nullable|string',
        'completed' => 'sometimes|boolean',
    ]);

    $todo->update($validated);

    return response()->json([
        'success' => true,
        'message' => 'Todo updated successfully',
        'data' => $todo,
    ]);
});

// DELETE: احذف todo
Route::delete('/todos/{id}', function ($id) {
    $todo = Todo::find($id);

    if (!$todo) {
        return response()->json([
            'success' => false,
            'message' => 'Todo not found',
        ], 404);
    }

    $todo->delete();

    return response()->json([
        'success' => true,
        'message' => 'Todo deleted successfully',
    ]);
});
```

الخطوة 3: أمثلة CURL (للـ Testing)

```
# ✅ GET كل الـ todos
curl http://localhost:8000/api/todos

# ✅ POST: انشئ todo جديد
curl -X POST http://localhost:8000/api/todos \
  -H "Content-Type: application/json" \
  -d '{
    "title": "Learn Laravel",
    "description": "Master Minimal APIs"
  }'

# Response:
# {
#   "success": true,
```

```
#   "message": "Todo created successfully",
#   "data": {
#     "id": 1,
#     "title": "Learn Laravel",
#     "description": "Master Minimal APIs",
#     "completed": false,
#     "created_at": "2026-01-21T10:30:00Z",
#     "updated_at": "2026-01-21T10:30:00Z"
#   }
# }

# ✅ GET: اجلب todo رقم 1
curl http://localhost:8000/api/todos/1

# ✅ PATCH: عدّل todo
curl -X PATCH http://localhost:8000/api/todos/1 \
  -H "Content-Type: application/json" \
  -d '{
    "completed": true
  }'

# ✅ DELETE: احذف todo
curl -X DELETE http://localhost:8000/api/todos/1
```

⚠ الأخطاء الشائعة

1) ترجع Errors بشكل عشوائي

```
// ❌ غلط: مافيش standard
Route::post('/todos', fn(Request $r) =>
    Todo::create($r->validated()) // إذا فشل، تصير exception!
);

// ✅ صح
Route::post('/todos', function (Request $r) {
    try {
        $todo = Todo::create($r->validated());
        return response()->json(['data' => $todo], 201);
    } catch (\Exception $e) {
        return response()->json([
            'success' => false,
            'message' => 'Something went wrong'
        ], 500);
    }
});
```

2) نسيان Validation

```
// ❌ خطيرة: أي حد يبعت أي حاجة
Route::post('/todos', fn(Request $r) => Todo::create($r->all()));

// ✅ آمنة
Route::post('/todos', fn(Request $r) =>
    Todo::create($r->validate([
        'title' => 'required|string|max:255',
        'description' => 'nullable|string',
    ]))
);
```

3) Logic كتير في الـ Route

```
// ❌ معقد جداً
Route::get('/todos/search', function (Request $r) {
    $query = Todo::query();

    if ($r->has('title')) $query->where('title', 'like', '%' . $r->title . '%');
    if ($r->has('completed')) $query->where('completed', $r->completed);
    if ($r->has('sort')) $query->orderBy('created_at', $r->sort);

    return $query->paginate(15);
});

// ✅ استخدم Service أو Controller، أوضح:
```

4) نسيان CORS + Authentication

```
// في routes/api.php، لازم تحط:
Route::middleware('auth:sanctum')->group(function () {
    Route::post('/todos', fn(Request $r) => Todo::create($r->validated()));
    Route::delete('/todos/{id}', fn($id) => Todo::find($id)->delete());
});
```

🎩 Senior Corner

1) متى Minimal ومتى Controller؟

```
📊 Decision Tree:

طلبك بسيط (1-2 أسطر)؟
  ✅ استخدم Minimal (Route + Closure)

Reusable؟ معقدة أو logic فيه
  ✅ استخدم Controller
```

```
فيه authorization معقدة؟
  ✅ استخدم Controller + Middleware + Policies

بتاع حاجات كتير؟ Endpoint الـ
  ✅ استخدم Service Layer
```

2) Form Request + Resources (بسرعة)

لما تكبر المشروع، استخدم الأدوات الصح:

```php
// php artisan make:request StoreTodoRequest
// app/Http/Requests/StoreTodoRequest.php

namespace App\Http\Requests;

class StoreTodoRequest extends FormRequest {
    public function authorize() {
        return true;
    }

    public function rules() {
        return [
            'title' => 'required|string|max:255',
            'description' => 'nullable|string',
        ];
    }
}

// في الـ route:
Route::post('/todos', function (StoreTodoRequest $request) {
    return response()->json([
        'data' => Todo::create($request->validated())
    ], 201);
});

// استخدام Resource للـ Response
// php artisan make:resource TodoResource

class TodoResource extends JsonResource {
    public function toArray($request) {
        return [
            'id' => $this->id,
            'title' => $this->title,
            'description' => $this->description,
            'is_completed' => (bool) $this->completed,
            'created_at' => $this->created_at->toIso8601String(),
        ];
    }
}
```

```
// في الـ route:
Route::get('/todos', fn() => TodoResource::collection(Todo::all()));
```

3) Authentication بـ Sanctum (Token-Based)

```
// Setup: php artisan install:api

// في routes/api.php
Route::middleware('auth:sanctum')->group(function () {
    // بس الـ authenticated users يقدروا يدخلوا هنا

    Route::post('/todos', fn(Request $r) =>
        Todo::create([...$r->validated(), 'user_id' => auth()->id()])
    );

    Route::delete('/todos/{id}', fn($id) =>
        Todo::where('id', $id)->where('user_id', auth()->id())->delete()
    );
});

// لما تسجل دخول:
POST /api/login
{
  "email": "ahmed@example.com",
  "password": "password123"
}

// ترجع token:
{
  "token": "abc123xyz...",
  "user": {...}
}

// بعدين استخدم الـ token في كل الـ requests:
curl -H "Authorization: Bearer abc123xyz..." http://localhost:8000/api/todos
```

4) Authorization (Policy بسيط)

```
// php artisan make:policy TodoPolicy

namespace App\Policies;

class TodoPolicy {
    public function update(User $user, Todo $todo) {
        return $user->id === $todo->user_id; // بتاعك بتقدر تعدل Todo مالك الـ
    }

    public function delete(User $user, Todo $todo) {
        return $user->id === $todo->user_id;
```

```
    }
}

// في الـ route
Route::patch('/todos/{id}', function ($id, Request $r) {
    $todo = Todo::find($id);

    // تفحص الـ policy
    $this->authorize('update', $todo);

    $todo->update($r->validated());
    return $todo;
});
```

5) Rate Limiting (تحديد الطلبات)

```
// في routes/api.php
Route::middleware('throttle:60,1')->group(function () {
    // 60 طلب في الدقيقة
    Route::get('/todos', fn() => Todo::all());
});

// للـ sensitive endpoints:
Route::middleware('throttle:5,1')->group(function () {
    // 5 طلبات فقط في الدقيقة
    Route::post('/todos', fn(Request $r) => Todo::create($r->validated()));
});
```

6) Pagination + Filtering

```
// GET /api/todos?page=1&limit=10&completed=true&sort=-created_at

Route::get('/todos', function (Request $request) {
    $query = Todo::query();

    // Filter
    if ($request->has('completed')) {
        $query->where('completed', $request->boolean('completed'));
    }

    if ($request->has('search')) {
        $query->where('title', 'like', '%' . $request->search . '%');
    }

    // Sort
    $sortField = $request->input('sort', '-created_at');
    $direction = str_starts_with($sortField, '-') ? 'desc' : 'asc';
    $field = ltrim($sortField, '-');
```

```
    $query->orderBy($field, $direction);

    // Paginate
    $limit = min($request->input('limit', 15), 100); // Max 100

    return response()->json([
        'success' => true,
        'data' => $query->paginate($limit),
    ]);
});
```

7) Error Handling (شكل Standard)

```
// app/Exceptions/Handler.php
public function render($request, Exception $exception) {
    if ($request->expectsJson()) {
        return response()->json([
            'success' => false,
            'message' => $exception->getMessage(),
            'errors' => method_exists($exception, 'errors') ? $exception->errors()
: [],
        ], $this->getStatusCode($exception));
    }

    return parent::render($request, $exception);
}

// Result: بتاعتك بنفس الشكل errors كل الـ
{
  "success": false,
  "message": "Validation failed",
  "errors": {"title": ["Title is required"]}
}
```

🎓 خاتمة + تحدي

تحديك ليك:

اعمل Endpoint جديد للـ Search مع Pagination:

```
GET /api/todos/search?q=laravel&page=1&limit=10
```

لازم تشتغل عليه:

- ✅ Search في الـ title و description
- ✅ Pagination (page + limit)
- ✅ Validation للـ query params

- ✅ Standard error response

الحل:

```
Route::get('/todos/search', function (Request $request) {
    $validated = $request->validate([
        'q' => 'required|string|min:1|max:100',
        'page' => 'integer|min:1',
        'limit' => 'integer|min:1|max:100',
    ]);

    $query = Todo::where('title', 'like', '%' . $validated['q'] . '%')
                 ->orWhere('description', 'like', '%' . $validated['q'] . '%');

    return response()->json([
        'success' => true,
        'data' => $query->paginate($validated['limit'] ?? 15),
    ]);
});
```

Checklist ليك:

- ✅ فهمت الفرق بين GET/POST/PATCH/DELETE
- ✅ عملت Minimal API بسيط
- ✅ عرفت ليه Status Codes مهمة
- ✅ عملت Validation صح
- ✅ فهمت Authentication + Authorization
- ✅ عرفت متى تستخدم Controller vs Minimal

📋 Cheat Sheet

Routes الأساسية

```
Route::get('/posts', [PostController::class, 'index']);
Route::post('/posts', [PostController::class, 'store']);
Route::get('/posts/{id}', [PostController::class, 'show']);
Route::patch('/posts/{id}', [PostController::class, 'update']);
Route::delete('/posts/{id}', [PostController::class, 'destroy']);

// أو بسيط (Minimal):
Route::get('/posts', fn() => Post::all());
Route::post('/posts', fn(Request $r) => Post::create($r->validated()));
```

Status Codes الأساسية

```
200 ✅ OK (GET/PATCH عملت)
201 ✅ Created (POST عملت)
204 ✅ No Content (DELETE بدون data)

400 ❌ Bad Request (غلط في الـ data)
401 ❌ Unauthorized (لازم تسجل دخول)
403 ❌ Forbidden (ممنوع عليك)
404 ❌ Not Found (الحاجة ما موجودة)
422 ❌ Unprocessable (Validation failed)

500 ❌ Server Error (bug في الكود)
```

Validation Rules الشهيرة

```php
'email' => 'required|email|unique:users',
'password' => 'required|min:8|confirmed',
'title' => 'required|string|max:255',
'age' => 'integer|min:18|max:65',
'status' => 'in:active,inactive,pending',
```

Best Practices

الممارسة	ليه؟
استخدم JSON بدل XML	أسرع + أسهل
ترجع Standard Response Shape	سهل على الـ frontend
استخدم Status Codes الصحيحة	واضح إيه اللي حصل
Validate كل الـ inputs	أمان
استخدم HTTPS في Production	أمان + تشفير
Pagination للـ Lists الكبيرة	Performance
استخدم Resources للـ Transform	Consistent output

النهاية 🚀

الفكرة الأساسية:

> بسيط، واضح، آمن، وسهل للـ maintain.

في السينيور عندك:

- معرفة متى تستخدم الـ tools الصح
- فهم عميق للـ HTTP و REST
- معمارة فكيك + قابل للـ scale

يلا.. ابدأ تكود ! 💪

□□ ♡ □□ □□□□□□ *Backend Instructor*