

How to make rest API in Laravel (API CRUD from Scratch)

1. Project Setup (if you haven't already)

```
composer create-project --prefer-dist laravel/laravel your-project-name  
cd your-project-name
```

2. Database Configuration

- Open `.env` and configure your database credentials (DB_HOST, DB_DATABASE, DB_USERNAME, DB_PASSWORD).

3. Model Creation

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

This command creates the **Student** model and a migration file.

4. Migration (Database Schema)

Open the generated migration file (e.g., `database/migrations/YYYY_MM_DD_HHMMSS_create_students_table.php`) and define your table schema. For example:

```
public function up(): void  
{  
    Schema::create('students', function (Blueprint $table) {  
        $table->id();  
        $table->string('student_name');  
        $table->string('student_gender');  
        $table->date('date_of_birth');  
        $table->string('student_phone')->nullable();  
        $table->text('student_address');  
        $table->timestamps();  
    });  
}
```

Run the migration:

```
php artisan migrate
```

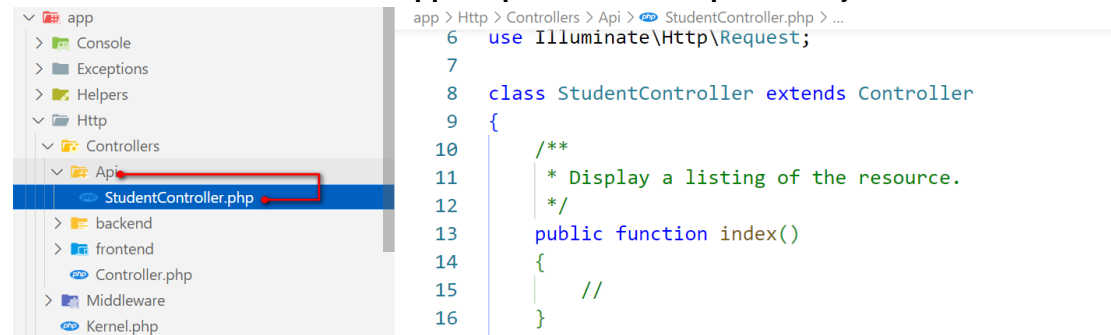
5. Model Definition (app/Models/Student.php)

```
class Student extends Model  
{  
    use HasFactory;  
    protected $table = 'students';  
    protected $primaryKey = 'id';  
    protected $fillable = [  
        'id',  
        'student_name',  
        'date_of_birth',  
        'student_phone',  
        'student_address',  
        'gender_id'  
    ];  
}
```

6. Controller Creation

`php artisan make:controller Api/StudentController --api`

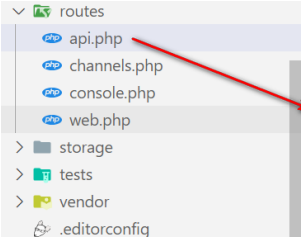
This creates a controller in the **app/Http/Controllers/Api** directory.



7. Controller Implementation (app/Http/Controllers/Api/StudentController.php)

```
php
class StudentController extends Controller
{
    protected $student;
    public function __construct(){
        $this->student = new Student();
    }
    public function index()
    {
        return $this->student->all();
    }
    public function store(Request $request)
    {
        return $this->student->create($request->all());
    }
    public function show(string $id)
    {
        $student = $this->student->find($id);
    }
    public function update(Request $request, string $id)
    {
        $student = $this->student->find($id);
        $student->update($request->all());
        return $student;
    }
    public function destroy(string $id)
    {
        $student = $this->student->find($id);
        return $student->delete();
    }
}
```

8. API Routes (routes/api.php)



```
21
22 Route::apiResource('students', StudentController::class); // Simplifies routing
23 // OR, if you prefer explicit routes:
24 // Route::get('/students', [StudentController::class, 'index']);
25 // Route::post('/students', [StudentController::class, 'store']);
26 // Route::get('/students/{student}', [StudentController::class, 'show']);
27 // Route::put('/students/{student}', [StudentController::class, 'update']);
28 // Route::delete('/students/{student}', [StudentController::class, 'destroy']);
29
```

9. API Test URL

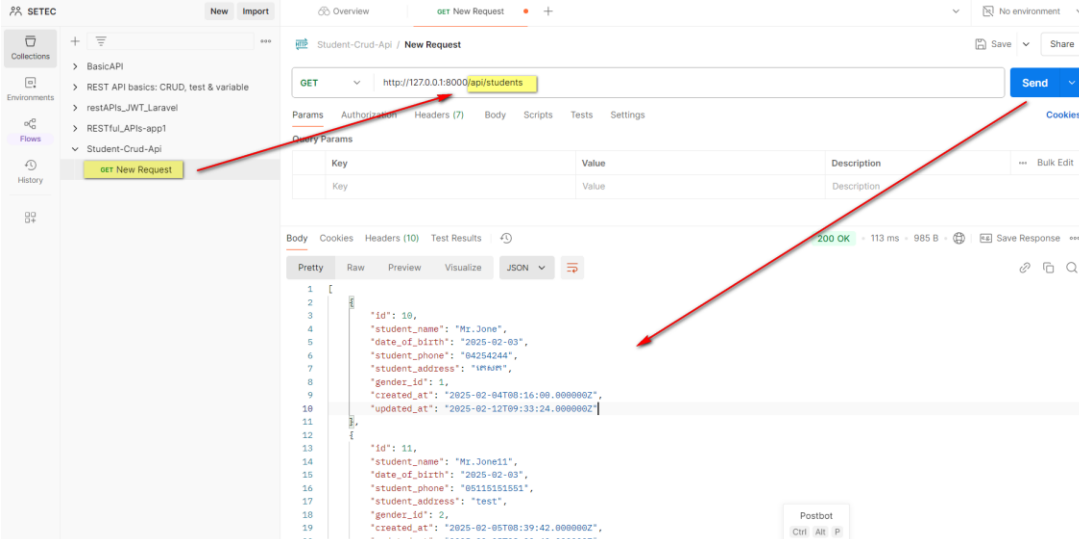
```
$ php artisan route:list --name='students'
```

Method	Route	Action	Controller
GET HEAD	api/students	students.index	Api\StudentController@index
POST	api/students	students.store	Api\StudentController@store
GET HEAD	api/students/{student}	students.show	Api\StudentController@show
PUT PATCH	api/students/{student}	students.update	Api\StudentController@update
DELETE	api/students/{student}	students.destroy	Api\StudentController@destroy

Showing [5] routes

10. Open Postman

✓ Get-All-Student-Lists

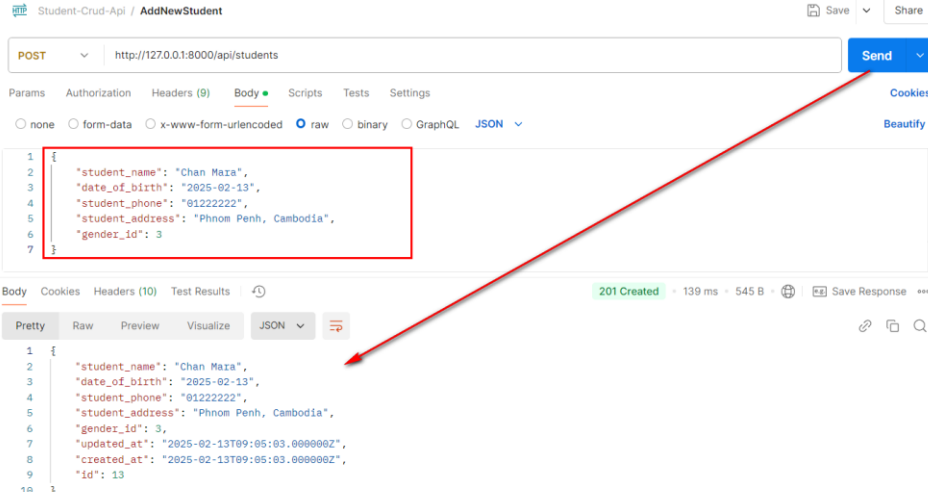


GET http://127.0.0.1:8000/api/students

200 OK · 113 ms · 995 B

```
{
  "id": 10,
  "student_name": "Mr. Jone",
  "date_of_birth": "2025-02-03",
  "student_phone": "84254244",
  "student_address": "Nsum",
  "gender_id": 1,
  "created_at": "2025-02-04T08:16:00.000000Z",
  "updated_at": "2025-02-12T09:33:24.000000Z"
},
{
  "id": 11,
  "student_name": "Mr. Jone11",
  "date_of_birth": "2025-02-03",
  "student_phone": "09115151551",
  "student_address": "Test",
  "gender_id": 2,
  "created_at": "2025-02-05T08:39:42.000000Z",
  "updated_at": "2025-02-12T09:33:24.000000Z"
}
```

✓ Add-New-Student



POST http://127.0.0.1:8000/api/students

201 Created · 139 ms · 545 B

```
{
  "student_name": "Chan Maza",
  "date_of_birth": "2025-02-13",
  "student_phone": "01222222",
  "student_address": "Phnom Penh, Cambodia",
  "gender_id": 3
}
```

```
{
  "student_name": "Chan Maza",
  "date_of_birth": "2025-02-13",
  "student_phone": "01222222",
  "student_address": "Phnom Penh, Cambodia",
  "gender_id": 3,
  "updated_at": "2025-02-13T09:05:03.000000Z",
  "created_at": "2025-02-13T09:05:03.000000Z",
  "id": 13
}
```

✓ Get-Student-ById

Student-Crud-API / GetStudentById

Save Share

GET http://127.0.0.1:8000/api/students/13 Send

Params Authorization Headers (7) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL

This request does not have a body

200 OK 58 ms 540 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 13,
3   "student_name": "Chan Maia",
4   "date_of_birth": "2025-02-13",
5   "student_phone": "01222222",
6   "student_address": "Phnom Penh, Cambodia",
7   "gender_id": 3,
8   "created_at": "2025-02-13T09:05:03.000000Z",
9   "updated_at": "2025-02-13T09:05:03.000000Z"
10 }
```

✓ Update-Student-ById

Student-Crud-API / EditStudentById

Save Share

PUT http://127.0.0.1:8000/api/students/13 Send

Params Authorization Headers (9) Body Scripts Tests Settings

none form-data x-www-form-urlencoded raw binary GraphQL JSON

```
1 {
2   "student_name": "Chan1 Maia1",
3   "date_of_birth": "2025-02-13",
4   "student_phone": "01222222",
5   "student_address": "Phnom Penh, Cambodia",
6   "gender_id": 3
7 }
```

Body Cookies Headers (10) Test Results

200 OK 91 ms 542 B Save Response

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 13,
3   "student_name": "Chan1 Maia1",
4   "date_of_birth": "2025-02-13",
5   "student_phone": "01222222",
6   "student_address": "Phnom Penh, Cambodia",
7   "gender_id": 3,
8   "created_at": "2025-02-13T09:05:03.000000Z",
9   "updated_at": "2025-02-13T09:16:50.000000Z"
10 }
```

✓ Delete-Student-ById

Student-Crud-API / DeleteStudentById

Save Share

DELETE http://127.0.0.1:8000/api/students/13 Send

Params Authorization Headers (7) Body Scripts Tests Settings

Query Params

Key	Value	Description	Bulk Edit
Key	Value	Description	

Body Cookies Headers (10) Test Results

200 OK 172 ms 316 B Save Response

Pretty Raw Preview Visualize HTML

```
1 1
```

How To Create Custom Standardized HTTP Responses In Laravel

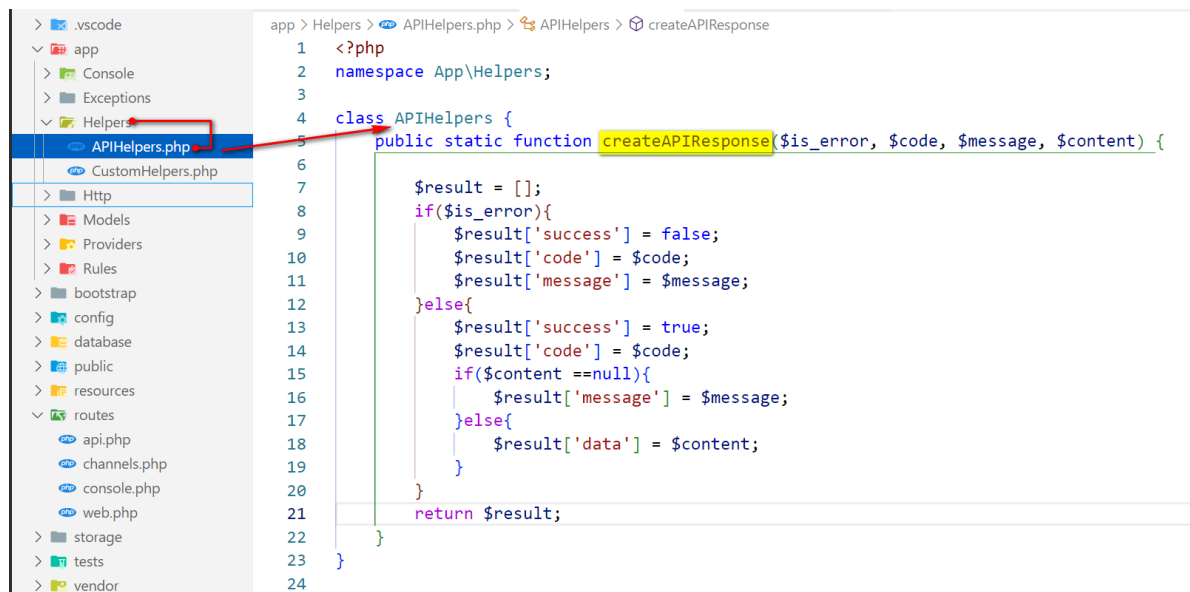
1. Create the **APIHelpers** Class

```
<?php

namespace App\Helpers;

class APIHelpers {
    public static function createAPIResponse($is_error, $code,
    $message, $content) {

        $result = [];
        if($is_error){
            $result['success'] = false;
            $result['code'] = $code;
            $result['message'] = $message;
        }else{
            $result['success'] = true;
            $result['code'] = $code;
            if($content ==null){
                $result['message'] = $message;
            }else{
                $result['data'] = $content;
            }
        }
        return $result;
    }
}
```



2. Modify Your Controller (Example: **TeacherController**)

```
class TeacherController extends Controller
{

    public function index()
    {
        $teachers = Teacher::all();
        $response = APIHelpers::createAPIResponse(false, 200, '', $teachers);
        return response()->json($response, 200);
    }

    public function show(string $id)
    {
        $teachers = Teacher::find($id);
        $response = APIHelpers::createAPIResponse(false, 200, '', $teachers);
        return response()->json($response, 200);
    }

    public function store(Request $request)
    {
        $teacher = new Teacher();
        $teacher->teacher_code = $request->teacher_code;
        $teacher->teacher_name = $request->teacher_name;
        $teacher->teacher_dob = $request->teacher_dob;
        $teacher->teacher_email = $request->teacher_email;
        $teacher->teacher_phone = $request->teacher_phone;
        $teacher->address = $request->address;
        $teacher->gender_id = $request->gender_id;
        $teacher_save = $teacher->save();
        if($teacher_save){
            $response = APIHelpers::createAPIResponse(false, 202, 'Teacher added successfully!', null);
            return response()->json($response, 202);
        }else{
            $response = APIHelpers::createAPIResponse(true, 400, 'Teacher creation failed!', null);
            return response()->json($response, 400);
        }
    }

    public function update(Request $request, string $id)
    {
        $teacher = Teacher::find($id);
        $teacher->teacher_code = $request->teacher_code;
        $teacher->teacher_name = $request->teacher_name;
        $teacher->teacher_dob = $request->teacher_dob;
        $teacher->teacher_email = $request->teacher_email;
        $teacher->teacher_phone = $request->teacher_phone;
        $teacher->address = $request->address;
        $teacher->gender_id = $request->gender_id;
        $teacher_update = $teacher->save();
        if($teacher_update){
            $response = APIHelpers::createAPIResponse(false, 200, 'Teacher updated successfully!', null);
            return response()->json($response, 200);
        }else{
            $response = APIHelpers::createAPIResponse(true, 400, 'Teacher update failed!', null);
            return response()->json($response, 400);
        }
    }

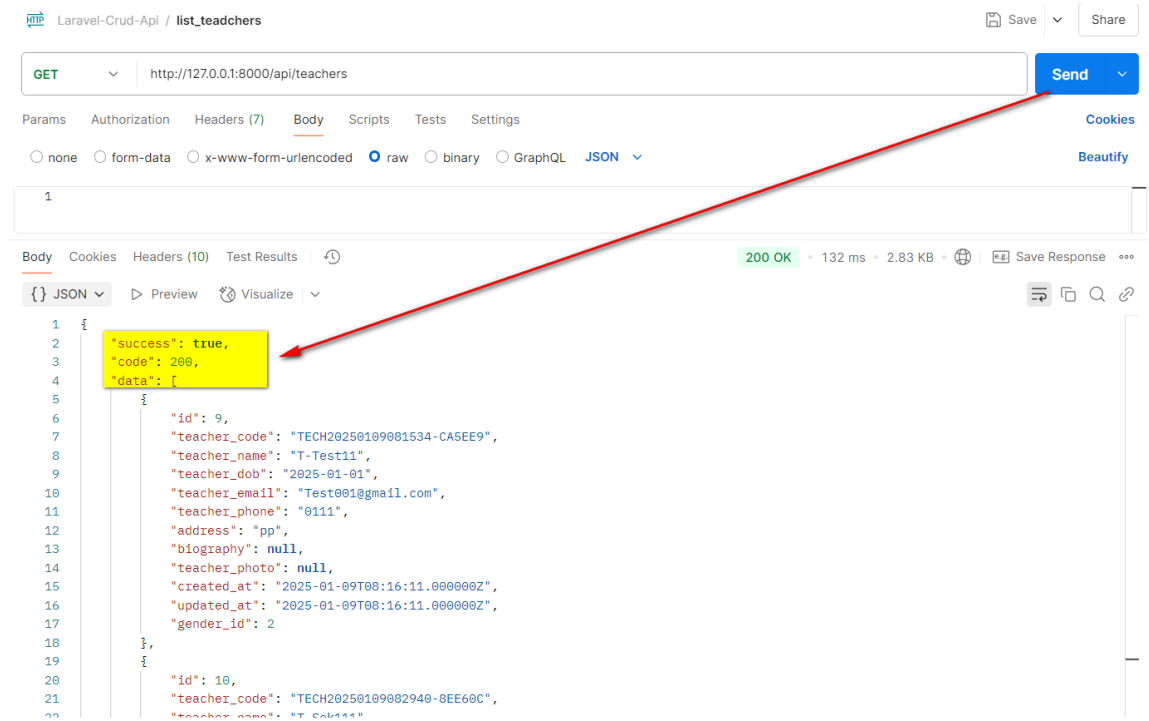
    public function destroy(string $id)
    {
        $teacher = Teacher::find($id);
        $teacher_delete = $teacher->delete();
        if($teacher_delete){
            $response = APIHelpers::createAPIResponse(false, 200, 'Teacher deleted successfully!', null);
            return response()->json($response, 200);
        }else{
            $response = APIHelpers::createAPIResponse(true, 400, 'Teacher delete failed!', null);
            return response()->json($response, 400);
        }
    }
}
```

3. Routing

```
Route::apiResource('/teachers', TeacherController::class);
```

4. Open & Run with Postman

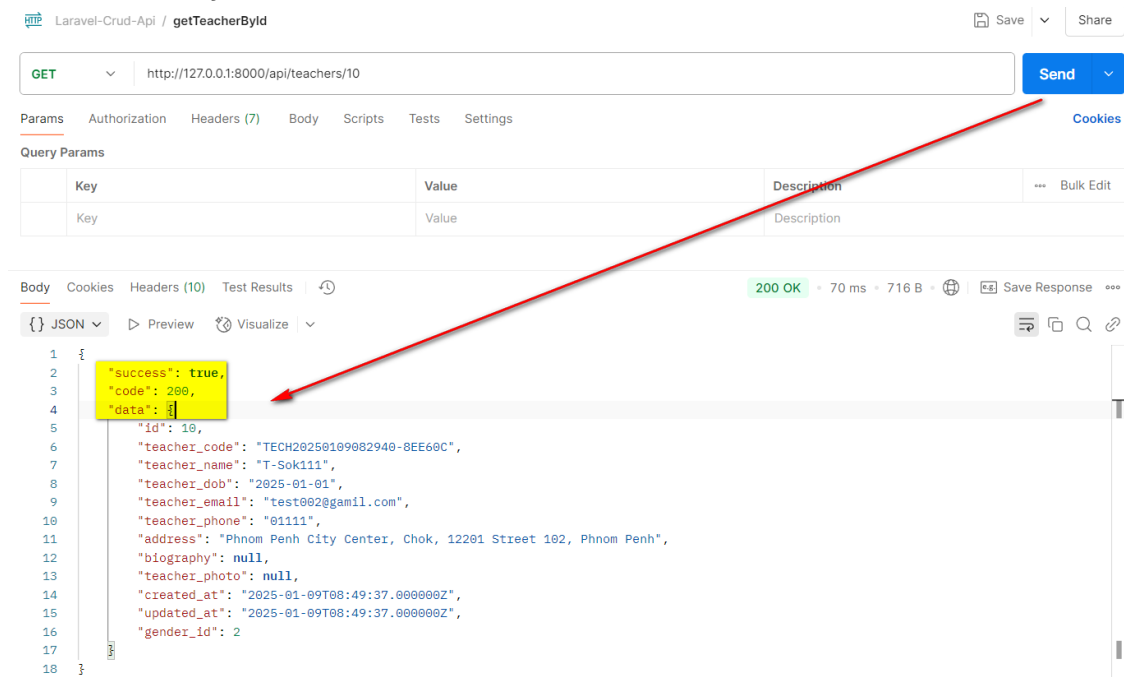
✓ Get-all-teacher-lists



Postman interface showing a GET request to `http://127.0.0.1:8000/api/teachers`. The response is a 200 OK status with a JSON body. A red arrow points to the `"success": true` field in the response body.

```
{
  "success": true,
  "code": 200,
  "data": [
    {
      "id": 9,
      "teacher_code": "TECH20250109081534-CA5EE9",
      "teacher_name": "T-Test11",
      "teacher_dob": "2025-01-01",
      "teacher_email": "Test001@gmail.com",
      "teacher_phone": "0111",
      "address": "pp",
      "biography": null,
      "teacher_photo": null,
      "created_at": "2025-01-09T08:16:11.000000Z",
      "updated_at": "2025-01-09T08:16:11.000000Z",
      "gender_id": 2
    },
    {
      "id": 10,
      "teacher_code": "TECH20250109082940-8EE60C",
      "teacher_name": "T-Sok111",
      "teacher_dob": "2025-01-01",
      "teacher_email": "test002@gmail.com",
      "teacher_phone": "0111",
      "address": "Phnom Penh City Center, Chok, 12201 Street 102, Phnom Penh",
      "biography": null,
      "teacher_photo": null,
      "created_at": "2025-01-09T08:49:37.000000Z",
      "updated_at": "2025-01-09T08:49:37.000000Z",
      "gender_id": 2
    }
  ]
}
```

✓ Get-teacher-ById



Postman interface showing a GET request to `http://127.0.0.1:8000/api/teachers/10`. The response is a 200 OK status with a JSON body. A red arrow points to the `"success": true` field in the response body.

```
{
  "success": true,
  "code": 200,
  "data": {
    "id": 10,
    "teacher_code": "TECH20250109082940-8EE60C",
    "teacher_name": "T-Sok111",
    "teacher_dob": "2025-01-01",
    "teacher_email": "test002@gmail.com",
    "teacher_phone": "0111",
    "address": "Phnom Penh City Center, Chok, 12201 Street 102, Phnom Penh",
    "biography": null,
    "teacher_photo": null,
    "created_at": "2025-01-09T08:49:37.000000Z",
    "updated_at": "2025-01-09T08:49:37.000000Z",
    "gender_id": 2
  }
}
```

✓ Inserting -teacher

Laravel-Crud-API / **addNewTeacher** Save Share

POST ▼ http://127.0.0.1:8000/api/teachers Send ▼

Params Authorization Headers (9) **Body** • Scripts Tests Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   "teacher_code": "TECH20250109082940-8EE60B",
3   "teacher_name": "T-Sok11122",
4   "teacher_dob": "2025-04-01",
5   "teacher_email": "test0023@gamil.com",
6   "teacher_phone": "01111",
7   "address": "Phnom Penh City Center, Chok, 12201 Street 102, Phnom Penh",
8   "gender_id": 2
9 }
```

Body Cookies Headers (10) Test Results ↺ 202 Accepted • 49 ms • 380 B Save Response ⋮

JSON ▼ Preview Visualize ▼

```
1 {
2   "success": true,
3   "code": 202,
4   "message": "Teacher added successfully!"
5 }
```

✓ Updating -teacher

Laravel-Crud-API / **updateTeacher** Save Share

PATCH ▼ http://127.0.0.1:8000/api/teachers/7 Send ▼

Params Authorization Headers (9) **Body** • Scripts Tests Settings Cookies Beautify

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

```
1 {
2   "teacher_code": "TECH20250109082940-8EE60D",
3   "teacher_name": "T-Sok111233",
4   "teacher_dob": "2025-04-10",
5   "teacher_email": "test33@gamil.com",
6   "teacher_phone": "016554477",
7   "address": "Phnom Penh City Center, Chok, 12201 Street 102, Phnom Penh",
8   "gender_id": 2
9 }
```

Body Cookies Headers (10) Test Results ↺ 200 OK • 76 ms • 376 B Save Response ⋮

JSON ▼ Preview Visualize ▼

```
1 {
2   "success": true,
3   "code": 200,
4   "message": "Teacher updated successfully!"
5 }
```

✓ Deleting -teacher

Laravel-Crud-API / **deleteTeacher** Save Share

DELETE ▼ http://127.0.0.1:8000/api/teachers/8 Send ▼

Params Authorization Headers (7) **Body** • Scripts Tests Settings Cookies

Query Params

Key	Value	Description	⋮	Bulk Edit
Key	Value	Description		

Body Cookies Headers (10) Test Results ↺ 200 OK • 79 ms • 376 B Save Response ⋮

JSON ▼ Preview Visualize ▼

```
1 {
2   "success": true,
3   "code": 200,
4   "message": "Teacher deleted successfully!"
5 }
```


Improved Code with Robust Error Handling

```
public function destroy(string $id)
{
    $teacher = Teacher::find($id);

    if (!$teacher) {
        // Teacher not found - this is the most common reason for "else" issues
        $response = APIHelpers::createAPIResponse(true, 404, 'Teacher not found!', null);
        return response()->json($response, 404);
    }


    try {
        $teacher_delete = $teacher->delete();

        if ($teacher_delete) {
            $response = APIHelpers::createAPIResponse(false, 200, 'Teacher deleted successfully!', null);
            return response()->json($response, 200);
        } else {
            // Database error during deletion
            $response = APIHelpers::createAPIResponse(true, 500, 'Teacher delete failed!', null);
            return response()->json($response, 500);
        }
    } catch (\Exception $e) {
        // Catch any exceptions (e.g., database errors)
        $response = APIHelpers::createAPIResponse(true, 500, 'Server error: ' . $e->getMessage(), null);
        return response()->json($response, 500);
    }
}
```

[HTTP](#) Laravel-Crud-API / deleteTeacher

 Save  Share

DELETE  http://127.0.0.1:8000/api/teachers/8


Send 

Params Authorization Headers (7) Body Scripts Tests Settings






Cookies

Query Params

Key	Value	Description	...	Bulk Edit
Key	Value	Description		

Body Cookies Headers (10) Test Results 

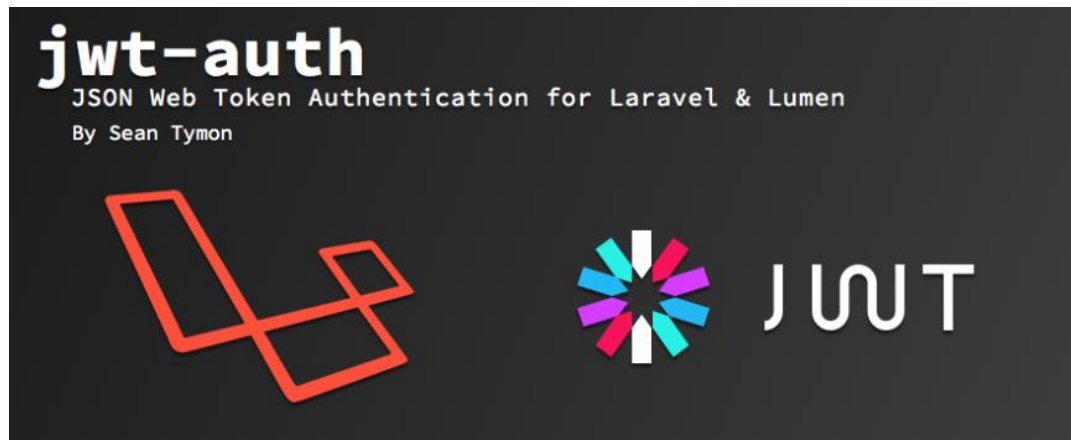
404 Not Found • 72 ms • 373 B •   Save Response ...

 JSON   Preview  Visualize 

```
1  {
2    "success": false,
3    "code": 404,
4    "message": "Teacher not found!"
5  }
```

JSON Web Token Authentication for Laravel & Lumen



1. Laravel Installation

- **Install via composer**

```
composer require tymon/jwt-auth
```

- **Publish the config**

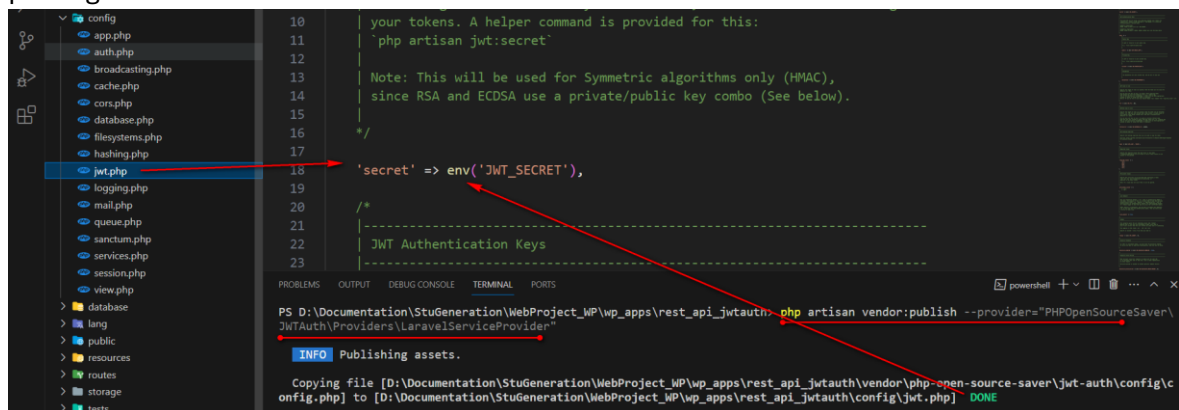
Run the following command to publish the package config file:

```
php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"
user@MSI MINGW64 /d/wamp64/www/dev/laravel/studentmgm-app
$ php artisan vendor:publish --provider="Tymon\JWTAuth\Providers\LaravelServiceProvider"

INFO Publishing assets.

Copying file [D:\wamp64\www\dev\laravel\studentmgm-app\vendor\tymon\jwt-auth\config\config.php] to [D:\wamp64\www\dev\laravel\studentmgm-app\config\jwt.php] DONE
```

You should now have a `config/jwt.php` file that allows you to configure the basics of this package.



- **Generate secret key**

I have included a helper command to generate a key for you:

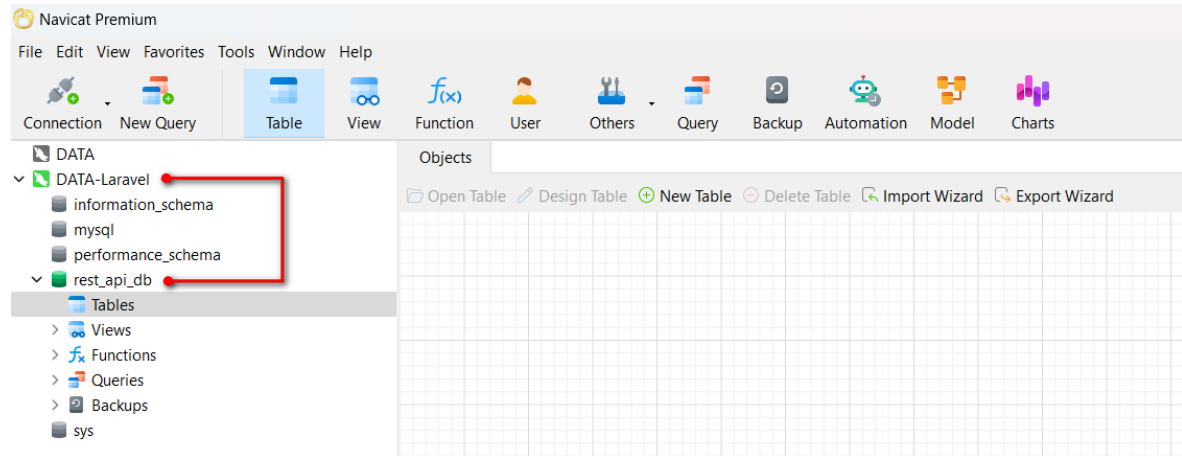
```
php artisan jwt:secret
```

This will update your `.env` file with something like `JWT_SECRET=foobar`

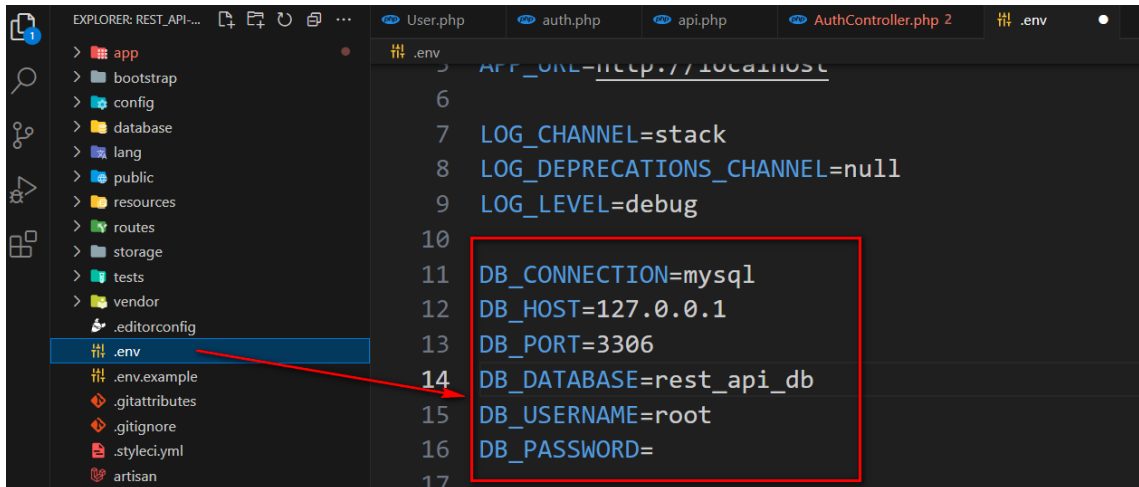
It is the key that will be used to sign your tokens. How that happens exactly will depend on the algorithm that you choose to use.

```
10
71 .env JWT_SECRET=c31A6UVtvMw1aEks77ydzXQA0DhDVqbKwtEsjC7D1px7bgt9c9MU8hk9AgmrK3L
72 .env.example
```

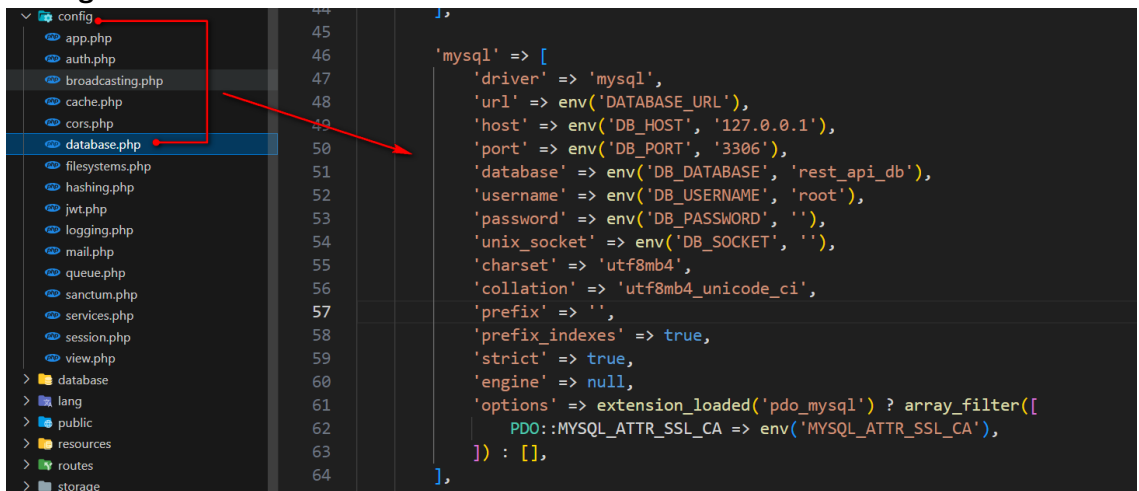
2. Database Configuration



■ config .env



■ Config -> database



- **Configure Auth guard: Change from **web** to **api****

Inside the **config/auth.php** file you will need to make a few changes to configure Laravel to use the **jwt** guard to power your application authentication.

Laravel JWT Auth

Search docs

- Home
- Laravel Installation
- Lumen Installation (incomplete)
- Quick start**
 - Update your User model
 - Configure Auth guard
 - Add some basic authentication routes
 - Create the AuthController
 - Authenticated requests
- Auth guard**
 - Configuration
 - Exception Handling
 - Resources

Configure Auth guard

Note: This will only work if you are using Laravel 5.2 and above.

Inside the **config/auth.php** file you will need to make a few changes to configure Laravel to use the **jwt** guard to power your application authentication.

Make the following changes to the file:

```
'defaults' => [
    'guard' => 'api',
    'passwords' => 'users',
],
...
'guards' => [
    'api' => [
        'driver' => 'jwt',
        'provider' => 'users',
    ],
],
```

Here we are telling the **api** guard to use the **jwt** driver, and we are setting the **api** guard as the default.

We can now use Laravel's built in Auth system, with jwt-auth doing the work behind the scenes!

- **Update Your User Model**

Your **User** model needs to implement the **Tymon\JWTAuth\Contracts\JWTSubject** interface.

```
public function getJWTIdentifier(){
    return $this->getKey();
}
public function getJWTCustomClaims(){
    return [];
}
```

Models

- Course.php
- Gender.php
- Major.php
- Permission.php
- Role.php
- Schedule.php
- Student.php
- Subject.php
- Teacher.php
- Teacherdetail.php
- User.php**

Providers

Rules

```
11 use Tymon\JWTAuth\Contracts\JWTSubject;
12
13 class User extends Authenticatable implements JWTSubject
14 {
15     use HasApiTokens, HasFactory, Notifiable;
16
17     public function getJWTIdentifier()
18     {
19         return $this->getKey();
20     }
21
22     public function getJWTCustomClaims()
23     {
24         return [];
25     }
26 }
```

- **Create the AuthController**

php artisan make:controller Api/AuthenticationController

```
<?php

namespace App\Http\Controllers\Api;

use App\Http\Controllers\Controller;
use App\Models\User;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Validator;
use Tymon\JWTAuth\Exceptions\JWTException;
use Tymon\JWTAuth\Exceptions\TokenInvalidException;
use Tymon\JWTAuth\Facades\JWTAuth;

class AuthenticationController extends Controller
{
    public function __construct()
    {
        $this->middleware('auth:api', ['except' => ['login']]);
    }

    public function __invoke(Request $request)
    {
        // Your logic here
        return response()->json(['message' => 'Invoked!']);
    }

    public function register(Request $request)
    {
        $validator = Validator::make($request->all(), [
            'name' => 'required|string|max:255',
            'email' => 'required|string|email|max:255|unique:users',
            'password' => 'required|string|min:8|confirmed',
        ]);

        if ($validator->fails()) {
            return response()->json($validator->errors(), 400);
        }

        $user = User::create([
            'name' => $request->get('name'),
            'email' => $request->get('email'),
            'password' => Hash::make($request->get('password')),
        ]);

        $token = JWTAuth::fromUser($user);
        return response()->json(compact('user', 'token'), 201);
    }

    public function login(Request $request)
    {
        $request->validate([
            'email' => 'required',
            'password' => 'required'
        ]);

        $credentials = request(['email', 'password']);
        try {
            if (! $token = auth()->attempt($credentials)) {
                return response()->json(['error' => 'Unauthorized'], 401);
            }
        } catch (JWTException $e) {
            return response()->json(['error' => 'Could not create token'], 500);
        }
        return $this->respondWithToken($token);
    }

    public function profile()
    {
        return response()->json(auth()->user());
    }

    public function logout()
    {
        auth()->logout();
        return response()->json(['message' => 'Successfully logged out!']);
    }

    public function refresh()
    {
        try {
            $newToken = JWTAuth::refresh(JWTAuth::getToken());
            return $this->respondWithToken($newToken);
        } catch (TokenInvalidException $e) {
            return response()->json(['error' => 'Token is invalid!'], 400);
        }
    }

    protected function respondWithToken($token)
    {
        return response()->json([
            'access_token' => $token,
            'token_type' => 'bearer',
            'expires_in' => JWTAuth::factory()->getTTL() * 60 // Use JWTAuth::factory()
        ], 200);
    }
}
```

- **Create Authentication Routes/Controllers**

```
Route::middleware('auth:api')->group(function () {
    Route::apiResource('/teachers', TeacherController::class);
});

// routes/api.php

Route::post('/login', [AuthenticationController::class, 'login'])->name('login');

Route::post('/register', [AuthenticationController::class, 'register'])->name('register');

Route::middleware('auth:api')->get('/profile', , [AuthenticationController::class, 'profile'])->name('profile');

Route::middleware('auth:api')->post('/logout', , [AuthenticationController::class, 'logout'])->name('logout');

Route::middleware('auth:api')->post('/refresh', , [AuthenticationController::class, 'refresh'])->name('refresh');
```

- **Check your postman**

Register

The screenshot shows the REST Client interface with a POST request to `http://127.0.0.1:8000/api/register`. The request body is a JSON object with the following content:

```
{
  "name": "Mr.Sok",
  "email": "sok222@gmail.com",
  "password": "sok12345678",
  "password_confirmation": "sok12345678"
}
```

The response is a 201 Created status with a JSON object containing user information and a token:

```
{
  "user": {
    "name": "Mr.Sok",
    "email": "sok222@gmail.com",
    "updated_at": "2025-04-11T08:46:41.000000Z",
    "created_at": "2025-04-11T08:46:41.000000Z",
    "id": 9
  },
  "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ3c3MiOiJodHRwOi8vMTIzLjAuMC4xOjgwMDAvYXBPpL3JlZ2ZlZG9yIiwiaWF0IjoxNzQ0MzYxMjAxLjC3eHAiOjE3NDQzNjQ0MDEsImZiIjE2MTc2NDM2MTIwMiwianRpIjoiaWQvYXN3VDlCkVwQzBBTIsInN1YiI6IjkiLCJwcnV0eSIyM2JkNW40QTQ5ZjYwMGFkYjY5MzZwMmMwMDA4ZnJkYjdhNTk3NmY3In0.alvnZ5dr4JGegwhLE7y8e1csBINsKjeU6qNZFER4ZEA"
}
```

A red arrow points from the 'Send' button to the response status '201 Created'.

Login

The screenshot shows the Swagger UI interface for a REST API. At the top, the API name is 'Laravel-Crud-API / Login'. The method is 'POST' and the URL is 'http://127.0.0.1:8000/api/login'. The request body is a JSON object with the following fields: 'email' (sok222@gmail.com) and 'password' (sok12345678). The response is a JSON object with the following fields: 'access_token' (eyJ3c3MiOiJodHRwOi8vMTI3LjAuMC4xOjgwMDAvYXBpL2xvZ21uIiwiaWF0IjoxNzQ0MzYxNjIwLjE4aW9Ej3NDQzNjUyMjA5Im5iZiI6MTc0NDNDMTYyMwCianRpdjoiOiEoxb0c2WFV3aXpXTVB0aCisInM1YiI6IjkiLjC3wcnYiOiIyMzI3NjNmM4OTQ5ZjYwMGFKyJm5ZTcwMmM0MDA4ZjZkYjdhNTk3NmY3In0i.MkLPtvV6wNewSbucqaNSf3_V6H5Ju5eQxnRY-YkiCmg'), 'token_type' (bearer), and 'expires_in' (3600).

Swagger UI for the `profile` endpoint. The URL is `http://127.0.0.1:8000/api/profile`. The `Authorization` tab is selected, showing a `Bearer Token` type. A warning message states: "Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about variables." The `Token` field contains a long alphanumeric string, which is highlighted with a red box. A red arrow points from this box to the `id` field in the JSON response body. The response is a JSON object with fields: `id`, `name`, `email`, `email_verified_at`, `level`, `active`, `language`, `created_at`, and `updated_at`. The status bar at the bottom shows `200 OK` and `340 ms`.

[illegible]

The screenshot shows the Swagger UI for the Laravel-Crud-API. The endpoint being tested is `POST http://127.0.0.1:8000/api/logout`. The request body is empty. The response is a `401 Unauthorized` status with a message: `"message": "Unauthenticated."`. A red arrow points to the response message.

Teacher lists (no token)

Laravel-Crud-API / list_teachers

SaveShare

GEThttp://127.0.0.1:8000/api/teachers

Send

ParamsAuthorizationHeaders (9)BodyScriptsTestsSettingsCookies

Auth Type

Bearer Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

Heads up! These parameters hold sensitive data. To keep this data secure while working in a collaborative environment, we recommend using variables. Learn more about [variables](#).

Token

Token

BodyCookies (2)Headers (8)Test Results

401 Unauthorized47 ms297 BSave Response

{ } JSONPreviewVisualize

```
1 {
2   "message": "Unauthenticated."
3 }
```