**Backend con Laravel**

Primero creamos un proyecto como normalmente lo haríamos:

**composer create-project laravel/laravel nombre\_api**

Luego instalamos la siguiente dependencia dentro del proyecto nuevo:   
php artisan install:api

Creamos un modelo Persona:

php artisan make:model Persona -mcrsf –api

**Para el Modelo:**

<?php

namespace App\Models;

use Illuminate\Database\Eloquent\Factories\HasFactory;

use Illuminate\Database\Eloquent\Model;

class Persona extends Model

{

    /\*\* @use HasFactory<\Database\Factories\PersonaFactory> \*/

    use HasFactory;

    protected $table = "personas";

    protected $fillable = [

        'nombres',

        'apellidos',

        'ci',

        'direccion',

        'telefono',

        'email',

    ];

}

**Creamos los respectivos migraciones, factories y seeders:**

<?php

use Illuminate\Database\Migrations\Migration;

use Illuminate\Database\Schema\Blueprint;

use Illuminate\Support\Facades\Schema;

return new class extends Migration

{

    /\*\*

     \* Run the migrations.

     \*/

    public function up(): void

    {

        Schema::create('personas', function (Blueprint $table) {

            $table->id();

            $table->string('nombres', 50);

            $table->string('apellidos', 50);

            $table->string('ci', 12)->unique();

            $table->string('direccion', 100);

            $table->string('telefono', 15);

            $table->string('email', 100)->unique();

            $table->timestamps();

        });

    }

    /\*\*

     \* Reverse the migrations.

     \*/

    public function down(): void

    {

        Schema::dropIfExists('personas');

    }

};

<?php

namespace Database\Factories;

use Illuminate\Database\Eloquent\Factories\Factory;

/\*\*

 \* @extends \Illuminate\Database\Eloquent\Factories\Factory<\App\Models\Persona>

 \*/

class PersonaFactory extends Factory

{

    /\*\*

     \* Define the model's default state.

     \*

     \* @return array<string, mixed>

     \*/

    public function definition(): array

    {

        return [

            'nombres' => $this->faker->firstName(),

            'apellidos' => $this->faker->lastName(),

            'ci' => $this->faker->unique()->numerify('##########'),

            'direccion' => $this->faker->address(),

            'telefono' =>  $this->faker->unique()->numerify('########'),

            'email' => $this->faker->unique()->safeEmail(),

        ];

    }

}

<?php

namespace Database\Seeders;

use App\Models\User;

use Illuminate\Database\Console\Seeds\WithoutModelEvents;

use Illuminate\Database\Seeder;

class DatabaseSeeder extends Seeder

{

    use WithoutModelEvents;

    /\*\*

     \* Seed the application's database.

     \*/

    public function run(): void

    {

        // User::factory(10)->create();

        User::factory()->create([

            'name' => 'Test User',

            'email' => 'test@example.com',

        ]);

        $this->call(

            PersonaSeeder::class

        );

    }

}

Ejecutamos la migración con:  
php artisan migrate:fresh –seed

**Controladores:**<?php

namespace App\Http\Controllers;

use App\Models\Persona;

use Illuminate\Http\Request;

class PersonaController extends Controller

{

    /\*\*

     \* Display a listing of the resource.

     \*/

    public function index()

    {

        return response()->json(Persona::all());

    }

    /\*\*

     \* Store a newly created resource in storage.

     \*/

    public function store(Request $request)

    {

        $persona = Persona::create($request->all());

        return response()->json($persona, 201);

    }

    /\*\*

     \* Display the specified resource.

     \*/

    public function show(Persona $persona)

    {

        return response()->json($persona);

    }

    /\*\*

     \* Update the specified resource in storage.

     \*/

    public function update(Request $request, Persona $persona)

    {

        $persona->update($request->all());

        return response()->json($persona);

    }

    /\*\*

     \* Remove the specified resource from storage.

     \*/

    public function destroy(Persona $persona)

    {

        $persona->delete();

        return response()->json($persona, 204);

    }

}

Ahora para las Routes, al ser backend se usa el archivo **api.php**:

<?php

use App\Http\Controllers\PersonaController;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Route;

Route::get('/user', function (Request $request) {

    return $request->user();

})->middleware('auth:sanctum');

Route::get('/user', function (Request $request) {

    return $request->user();

})->middleware('auth:sanctum');

Route::apiResource('personas', PersonaController::class);

Y para acceder al recurso, usando postman:

<http://127.0.0.1:8000/api/personas>

**Authenticacion y Authorizacion:**

Instalamos la librería:

composer require firebase/php-jwt

Y añadimos lo siguiente en las variables de entorno:

**JWT\_SECRET=secret**

**JWT\_ALGORITHM=HS256**

Ahora creamos el Middleware:  
php artisan make:middleware JwtMiddleware

Y dentro del middleware ponemos lo siguiente:

<?php

namespace App\Http\Middleware;

use Closure;

use Firebase\JWT\JWT;

use Firebase\JWT\Key;

use Illuminate\Http\Request;

use Symfony\Component\HttpFoundation\Response;

class JwtMiddleware

{

    /\*\*

     \* Handle an incoming request.

     \*

     \* @param  \Closure(\Illuminate\Http\Request): (\Symfony\Component\HttpFoundation\Response)  $next

     \*/

    public function handle(Request $request, Closure $next): Response

    {

        try {

            $autorizacion = $request->header('Authorization');

            $jwt = substr($autorizacion, 7);

            $key = env('JWT\_SECRET');

            $algoritmo = env('JWT\_ALGORITHM');

            $datos = JWT::decode($jwt,  new Key($key, $algoritmo));

            $request->attributes->add(['usuario' => $datos->data]);

        } catch (\Exception $e) {

            return response()->json(['status' => 'No autorizado ' . $e->getMessage()], 401);

        }

        return $next($request);

    }

}

Ahora debemos proteger las rutas:

Route::apiResource('personas', PersonaController::class)->middleware(JwtMiddleware::class);

Ahora debemos crear un controlador para el Login y así recibir un token:

php artisan make:controller LoginController

<?php

namespace App\Http\Controllers;

use App\Models\Persona;

use Firebase\JWT\JWT;

use Illuminate\Http\Request;

use Illuminate\Support\Facades\Hash;

class LoginController extends Controller

{

    public function login(Request $request)

    {

        $usuario = Persona::where('email', $request->email)->first();

        if ($usuario == null) {

            return response()->json([

                'mensaje' => 'usuario invalido',

            ], 400);

        }

        if (Hash::check($request->password, $usuario->password)) {

            $key = env('JWT\_SECRET');

            $algoritmo = env('JWT\_ALGORITHM');

            $time = time();

            $token = array(

                'iat' => $time, // Tiempo que inició el token

                'exp' => $time + (1200 \* 60), // Tiempo que expirará el token (+1 hora)

                'data' => [ // información del usuario

                    'user\_id' => $usuario->id,

                ],

            );

            $jwt = JWT::encode($token, $key, $algoritmo);

            return response()->json([

                'mensaje' => 'Se logró autenticar al usuario',

                'token' => $jwt,

                'type' => 'bearer',

                'expires' => $time + (1200 \* 60),

                'usuario' => $usuario

            ], 200);

        } else {

            return response()->json([

                'mensaje' => 'Contraseña invalida',

                'status' => 400

            ], 400);

        }

    }

}

Y añadimos el controlador del Login:

Route::post('login', [LoginController::class, 'login']);

Usando:

<http://127.0.0.1:8000/api/login>

Y con el body:

{

    "email": "test@example.com",

    "password": "password"

}

Nos genera un token:

{

    "mensaje": "Se logró autenticar al usuario",

    "token": "eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJpYXQiOjE3NjExNzA3NDgsImV4cCI6MTc2MTI0Mjc0OCwiZGF0YSI6eyJ1c2VyX2lkIjoxfX0.Hpwx1M3B-yF5tTb5fYLtqSuU1Fny6yAA2FmLIq9RZ5M",

    "type": "bearer",

    "expires": 1761242748,

    "usuario": {

        "id": 1,

        "name": "Test User",

        "email": "test@example.com",

        "email\_verified\_at": "2025-10-22T21:36:01.000000Z",

        "created\_at": "2025-10-22T21:36:02.000000Z",

        "updated\_at": "2025-10-22T21:36:02.000000Z"

    }

}

Y Ahora para poder realizar solicitudes simplemente es de agregar el token en authentication y se podrá acceder a los recursos protegidos.