

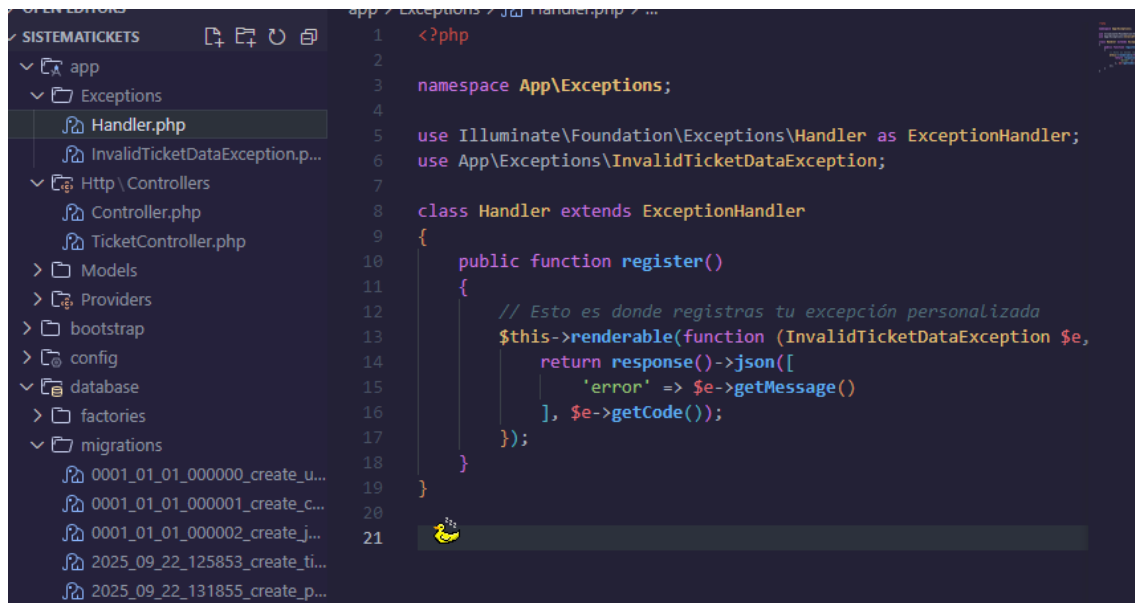


SEMINARIO DE COMPLEMENTACIÓN PRÁCTICA III



GUÍA

PRÁCTICA #6 – SEMINARIO DE COMPLEMENTACIÓN PRÁCTICA III – 601

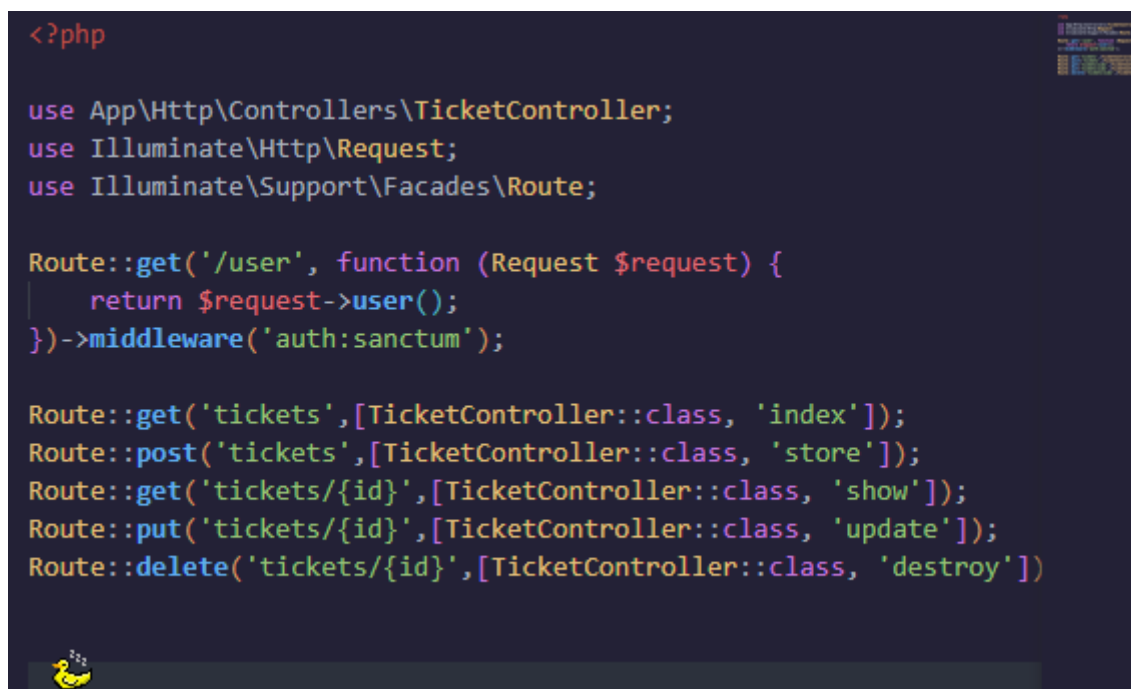


```
<?php
namespace App\Exceptions;

use Illuminate\Foundation\Exceptions\Handler as ExceptionHandler;
use App\Exceptions\InvalidTicketDataException;

class Handler extends ExceptionHandler
{
    public function register()
    {
        // Esto es donde registras tu excepción personalizada
        $this->renderable(function (InvalidTicketDataException $e,
            return response()->json([
                'error' => $e->getMessage()
            ], $e->getStatusCode());
    });
}
}
```

Se crearon las Exceptions



```
<?php

use App\Http\Controllers\TicketController;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Route;

Route::get('/user', function (Request $request) {
    return $request->user();
})->middleware('auth:sanctum');

Route::get('tickets',[TicketController::class, 'index']);
Route::post('tickets',[TicketController::class, 'store']);
Route::get('tickets/{id}',[TicketController::class, 'show']);
Route::put('tickets/{id}',[TicketController::class, 'update']);
Route::delete('tickets/{id}',[TicketController::class, 'destroy'])
```

Estos son los rutados de la api para que funcione el postman

HTTP <http://127.0.0.1:8000/api/Patients> Save

POST <http://127.0.0.1:8000/api/tickets> Send

Params Auth Headers (8) **Body** Pre-req. Tests Settings Cookies

raw JSON Beautify

```
1  
2  "description": "hola",  
3  "priority": "low"  
4  
5  
6  
7  
8  
9  
10  
11  
12  
13
```

Body 500 Internal Server Error 1274 ms 1.18 MB Save Response

Pretty Raw **Preview** Visualize

Internal Server Error

App\Exceptions\InvalidTicketDataExceptio

Datos invalidos para crear el ticket

LARAVEL 12.30.1 PHP 8.2.12 **UNHANDLED** **CODE 422**

acá como vemos que cuando no están todos los datos sale el mensaje personalizado que pusimos y ya no salga el predeterminado

```

class TicketController extends Controller
{
    //
    // Mostrar todos Los tickets
    public function index()
    {
        $tickets = Ticket::all();
        return response()->json($tickets);
    }
    // Crear un nuevo ticket
    public function store(Request $request)
    {
        // Validación de datos
        $rules = [
            'title' => 'required|string|max:255',
            'description' => 'required|string',
            'priority' => 'required|in:low,medium,high'
        ];
        $validator = Validator::make($request->all(), $rules);
        if ($validator->fails()) {
            throw new InvalidTicketDataException();
        }

        // Crear el ticket
        try {
            $ticket = Ticket::create([
                'title' => $request->title,
                'description' => $request->description,
                'priority' => $request->priority
            ]);
            return response()->json(['message' => 'Ticket creado con éxito', 'ticket' => $ticket], 201);
        } catch (\Exception $e) {
            return response()->json(['error' => 'Error al crear el ticket: ' . $e->getMessage()], 500);
        }
    }

    // Ver detalles de un ticket
    public function show($id)
    {
        try {
            $ticket = Ticket::findOrFail($id);
            return response()->json($ticket);
        } catch (\Exception $e) {
            return response()->json(['error' => 'Ticket no encontrado'], 404);
        }
    }
}

```

```

public function update(Request $request, $id)
{
    $rules = [
        'title' => 'required|string|max:255',
        'description' => 'required|string',
        'priority' => 'required|in:low,medium,high'
    ];
    $validator = Validator::make($request->all(), $rules);

    if ($validator->fails()) {
        throw new InvalidTicketDataException(json_encode($validator->errors()));
    }

    try {
        $ticket = Ticket::findOrFail($id);
        $ticket->update([
            'title' => $request->title,
            'description' => $request->description,
            'priority' => $request->priority
        ]);
        return response()->json(['message' => 'Ticket actualizado con éxito', 'ticket' => $ticket],
            200);
    } catch (Exception $e) {
        return response()->json(['error' => 'Ticket no encontrado o error al actualizar: ' .
            $e->getMessage()], 404);
    }
}

public function destroy($id)
{
    try {
        $ticket = Ticket::findOrFail($id);
        $ticket->delete();
        return response()->json(['message' => 'Ticket eliminado con éxito'], 200);
    } catch (Exception $e) {
        return response()->json(['error' => 'Ticket no encontrado o error al eliminar: ' .
            $e->getMessage()], 404);
    }
}

```

aquí el código de todo del GET GIT PUT y DELETE para el postman y funcione correctamente

HTTP

http://127.0.0.1:8000/api/Patients

Save

GET

http://127.0.0.1:8000/api/tickets

Send

Params

Auth

Headers (8)

Body

Pre-req.

Tests

Settings

Cookies

raw

JSON

Beautify

1

2

3

4

5

6

7

8

9

10

11

12

13

14

"title": "hola2.0",

"description": "hola",

"priority": "low"

Body

200 OK

212 ms

727 B

Save Response

Pretty



Raw

Preview

Visualize

[{"id":1,"title":"hola2.0","description":"hola","priority":"low","created_at":"2025-09-22T13:55:52.000000Z","updated_at":"2025-09-22T15:19:40.000000Z"}, {"id":2,"title":"Problemas con la conexion","description":"El sistema no se conecta a internet","priority":"high","created_at":"2025-09-22T13:58:16.000000Z","updated_at":"2025-09-22T13:58:16.000000Z"}, {"id":3,"title":"o\u00f1o","description":"hola","priority":"high","created_at":"2025-09-22T15:07:36.000000Z","updated_at":"2025-09-22T15:07:36.000000Z"}]

GET

 **http://127.0.0.1:8000/api/Patients**  Save

POST

http://127.0.0.1:8000/api/tickets

Send

ParamsAuthHeaders (8)BodyPre-req. Tests SettingsCookies

rawJSONBeautify

1

2

3

4

5

6

7

8

9

10


11

12

13

14

Body

 201 Created 167 ms 435 B Save Response

PrettyRawPreviewVisualize

```
{
  "message": "Ticket creado con \r\n \u00e9xito",
  "ticket": {
    "title": "hola2.",
    "description": "hola",
    "priority": "low",
    "updated_at": "2025-09-22T17:17:10.000000Z",
    "created_at": "2025-09-22T17:17:10.000000Z",
    "id": 4
  }
}
```

POST

HTTP

http://127.0.0.1:8000/api/Patients

Save

PUT

http://127.0.0.1:8000/api/tickets/1

Send

Params

Auth

Headers (8)

Body

Pre-req.

Tests

Settings

Cookies

raw

JSON

Beautify

1

2

3

4

5

6

7

8

9

10

11

12

13

14

Body

200 OK

171 ms

426 B

Save Response

Pretty

Raw

Preview

Visualize

{

"message": "Ticket actualizado con \u00e9xito",

"ticket": {

"id": 1,

"title": "GAAAA",

"description": "hola",

"priority": "low",

"created_at": "2025-09-22T13:55:52.000000Z",

"updated_at": "2025-09-22T17:17:34.000000Z"

}

}

PUT

DELETE

http://127.0.0.1:8000/api/tickets/1

Send

ParamsAuthHeaders (8)BodyPre-req. Tests SettingsCookies

rawJSONBeautify

1

2

3

4

5

6

7

8

9

10

11

12

13

14

```
"title": "GAAAA",
"description": "hola",
"priority": "low"
```

Body200 OK167 ms266 BSave Response

PrettyRawPreviewVisualize

{"message": "Ticket eliminado con \u00e9xito"}

DELETE