

## CASOS DE PRUEBAS EJECUTADOS Y EVIDENCIADOS

### - EMPLOYEE

#### Caso de Prueba 1: Crear un nuevo empleado

- **Endpoint:** /api/v1/employees
- **Método HTTP:** POST
- **Descripción:** Registrar un nuevo empleado en el sistema.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de petición:**

```
{  
    "name": "Carlos Pérez",  
    "email": "carlos.perez@empresa.com",  
    "password": "SecurePass123",  
    "department": "Soporte Técnico"  
}
```

- **Código de Respuesta Esperado:** 201 Created
- **Respuesta esperada:**

```
{  
    "id": 2,  
    "name": "Carlos Pérez",  
    "email": "carlos.perez@empresa.com",  
    "role": "USER",  
    "department": "Soporte Técnico"  
}
```

- **Prueba exitosa si:** Se recibe 201 Created y el empleado se almacena con los datos enviados.
- **Pasos de Ejecución de prueba:**
  - a) Agregar el cuerpo de la petición para crear usuario:

POST Create employee

Método HTTP: POST  
Endpoint: {{baseURL}} /api/v1/employees

```

1 {
2   "name": "Carlos Pérez",
3   "email": "carlos.perez@empresa.com",
4   "password": "123",
5   "department": "Soporte Técnico"
6 }
    
```

Cuerpo de solicitud

Response | History

Send

1:21 p.m.  
23/09/2025

b) Enviar la petición y devuelve el usuario creado con código 201

POST Create employee

Método HTTP: POST  
Endpoint: {{baseURL}} /api/v1/employees

```

1 {
2   "name": "Carlos Pérez",
3   "email": "carlos.perez@empresa.com",
4   "password": "SecurePass123",
5   "department": "Soporte Técnico"
6 }
    
```

Body | Cookies | Headers (11) | Test Results | 201 ms | 451 B | Save Response | ...

201 Created

2:18 p.m.  
23/09/2025

Respuesta

```

1 {
2   "id": 2,
3   "name": "Carlos Pérez",
4   "email": "carlos.perez@empresa.com",
5   "role": "USER",
6   "department": "Soporte Técnico"
7 }
    
```

- **¿Prueba Superada?:** Sí, Aprobada
- **Observaciones:** La API respondió con 201 creando el usuario nuevo

## Caso de Prueba 2: Error al crear empleado con email inválido

- **Endpoint:** /api/v1/employees
- **Método HTTP:** POST
- **Descripción:** Intentar registrar un empleado con un correo en formato incorrecto.
- **Cuerpo de petición:**

{

```

    "name": "Juan Pérez",
    "email": "juan.perez",
    "password": "TuPasswordSegura123",
    "department": "Soporte"
}

```

- **Código de Respuesta Esperado:** 400 Bad Request

- **Respuesta esperada:**

```

{
  "error": "Formato de email inválido."
}

```

- **Prueba exitosa si:** Se recibe 400 Bad Request con mensaje de validación adecuado.

- **Pasos de Ejecución de prueba:**

- Agregar el cuerpo de la petición para crear usuario pero con un correo electrónico invalido:

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections, environments, and history. The main area shows a POST request to the endpoint `/api/v1/employees`. The 'Body' tab is selected, showing the following JSON payload:

```

1: {
2:   "name": "Juan Pérez",
3:   "email": "juan.perez",
4:   "password": "TuPasswordSegura123",
5:   "department": "Soporte"
6: }

```

This payload is highlighted with a red box and labeled 'Cuerpo de la petición'. To the right of the body, there's a 'Send' button also highlighted with a red box. Below the request, there's a preview area showing a calendar for Wednesday, September 24, 2025, with the date 24 highlighted. The status bar at the bottom indicates 'Miércoles, 24 de septiembre de 2025' and 'Configuración de la fecha y la hora'.

- Enviar la petición y el error con el código 400:

The screenshot shows the Postman interface with a collection named 'TICKET\_SYSTEM\_EMPLOYEES'. A POST request is being made to `/api/v1/employees`. The request body is set to `JSON` and contains the following data:

```

1 {
2   "name": "Juan Pérez",
3   "email": "juan.perez",
4   "password": "TuPasswordSegura123",
5   "department": "Soporte"
6 }

```

The response status is **Código HTTP: 400 Bad Request**, and the response body is:

```

1 {
2   "email": "Formato de email inválido."
3 }

```

A calendar window is overlaid on the bottom right, showing the date as Wednesday, September 24, 2025.

### Caso de Prueba 3: Actualizar un empleado existente

- Endpoint:** `/api/v1/employees/{id}`
- Método HTTP:** PUT
- Descripción:** Modificar los datos de un empleado ya registrado.
- Parámetros:**
  - `id: 2`
- Cuerpo de petición:**

```
{
  "name": "Camilo Quintero",
  "email": "camilo.quintero@example.com",
  "role": "ADMIN",
  "department": "RRHH"
}
```

- Código de Respuesta Esperado:** 200 OK
- Respuesta esperada:**

```
{
```

```

"id":2,
"name":"Camilo Quintero",
"email":"camilo.quintero@example.com",
"role":"ADMIN",
"department":"RRHH"
}

```

- **Prueba exitosa si:** Devuelve 200 OK con los datos actualizados.
- **Pasos de Ejecución de prueba:**

a) Agregar el cuerpo de la petición para editar un empleado existente:

The screenshot shows the Postman interface with a collection named 'My Workspace'. A PUT request is selected for the 'TICKET\_SYSTEM\_EMPLOYEES' endpoint, specifically for employee ID 2. The request URL is `PUT {{base_url}}/api/v1/employees/2`. The 'Body' tab is active, showing the following JSON payload:

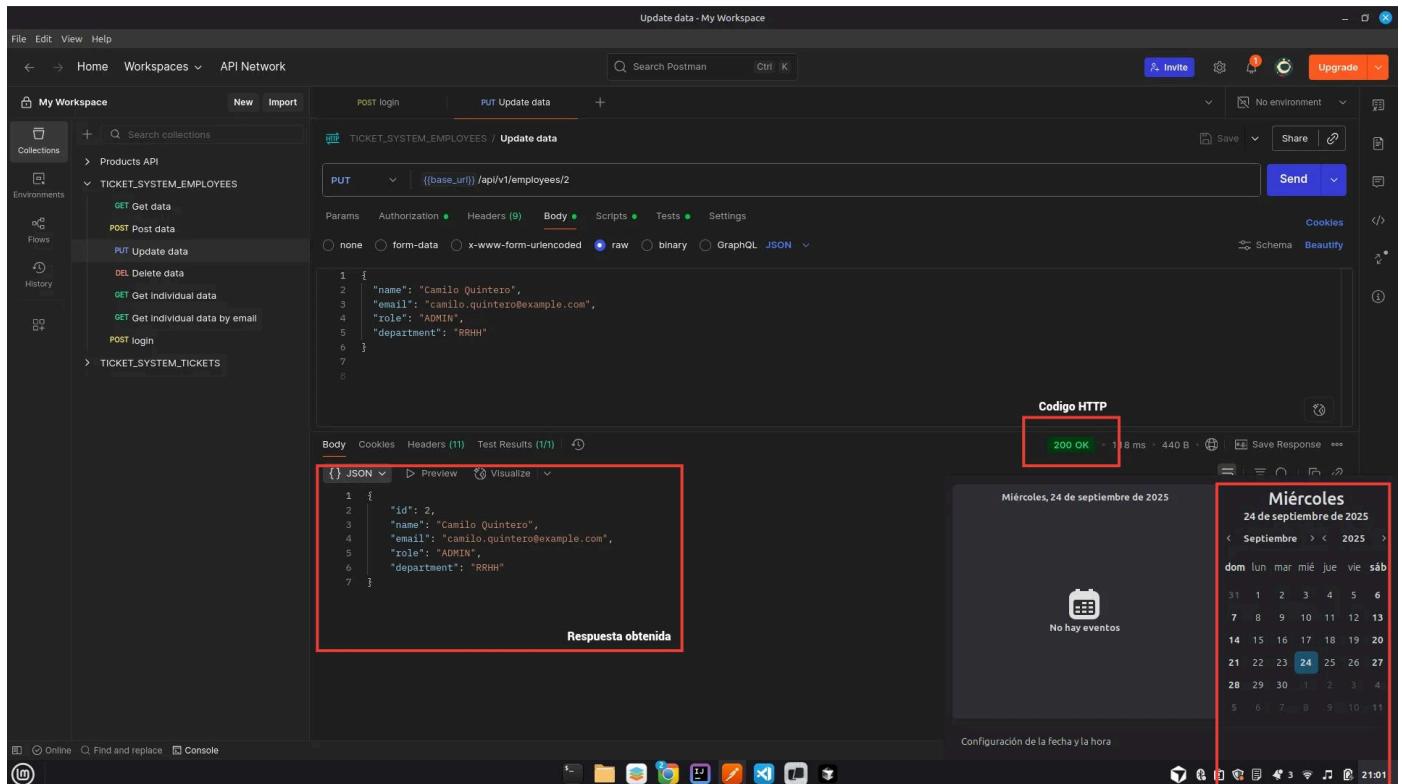
```

1 {
2   "name": "Camilo Quintero",
3   "email": "camilo.quintero@example.com",
4   "role": "ADMIN",
5   "department": "RRHH"
6 }
7
8

```

A red box highlights the 'Cuerpo de la petición' (Body of the request) area. The 'Send' button is also highlighted in blue. Below the request, there is a response preview showing a calendar for Wednesday, September 24, 2025. The date 24 is highlighted in blue. The response message says 'No hay eventos' (No events).

b) Enviar la petición y devuelve el usuario actualizado con código 200:



## Caso de Prueba 4: Error al actualizar empleado inexistente

- **Endpoint:** /api/v1/employees/{id}
- **Método HTTP:** PUT
- **Descripción:** Intentar actualizar un empleado que no existe.
- **Parámetros:**
  - id: 999
- **Cuerpo de petición:**

```
{
  "name": "Fantasma",
  "email": "fantasma@email.com",
  "role": "USER",
  "department": "Ventas"
}
```

- **Código de Respuesta Esperado:** 404 Not Found
- **Respuesta esperada:**

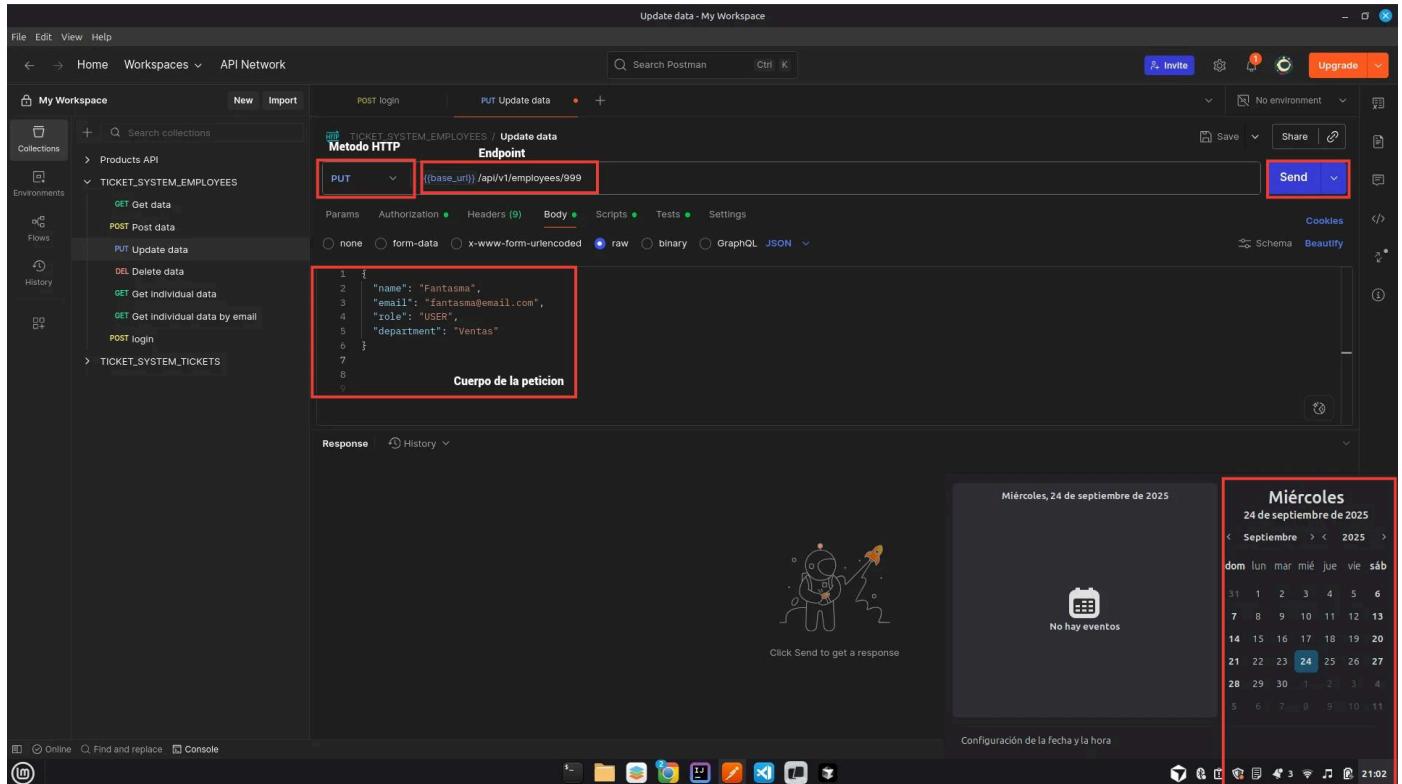
```
{
```

"error":"Empleado no encontrado con ID: 999"

}

- **Prueba exitosa si:** Devuelve 404 Not Found con mensaje claro.
- **Pasos de Ejecución de prueba:**

a) Agregar el cuerpo de la petición para editar un empleado inexistente:

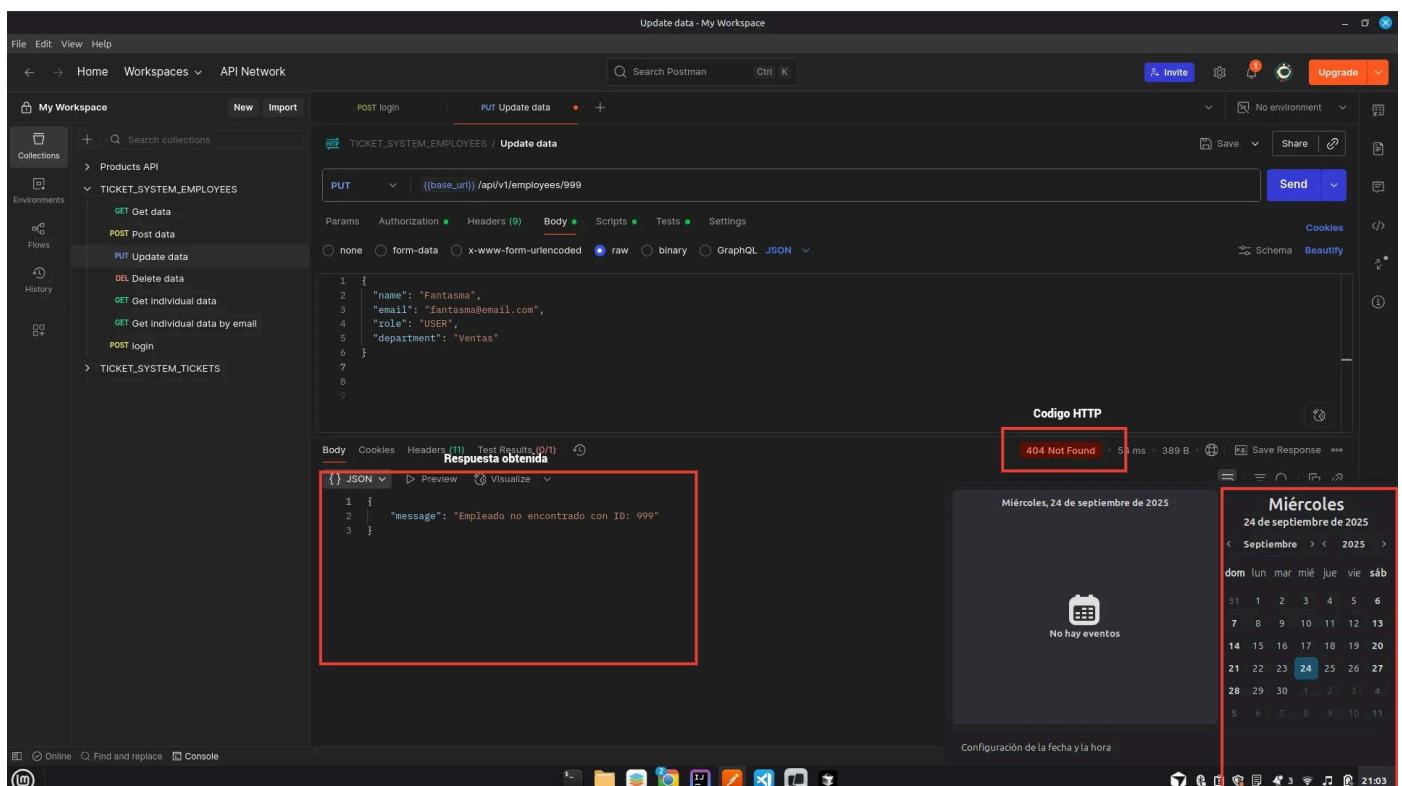


The screenshot shows the Postman interface with a collection named "TICKET\_SYSTEM\_EMPLOYEES". A PUT request is selected with the endpoint `{base_url}/api/v1/employees/999`. The "Body" tab is open, showing a raw JSON payload:

```
1 {
2   "name": "Fantasma",
3   "email": "fantasma@email.com",
4   "role": "USER",
5   "department": "Ventas"
6 }
```

The "Send" button is highlighted in red. To the right, a calendar window is open, showing the date as Wednesday, September 24, 2025. The calendar highlights the 24th of September.

b) Enviar la petición y devuelve el error 404:



The screenshot shows the same setup as the previous one, but the response has changed. The "Body" tab displays the following JSON response:

```
1 {
2   "message": "Empleado no encontrado con ID: 999"
3 }
```

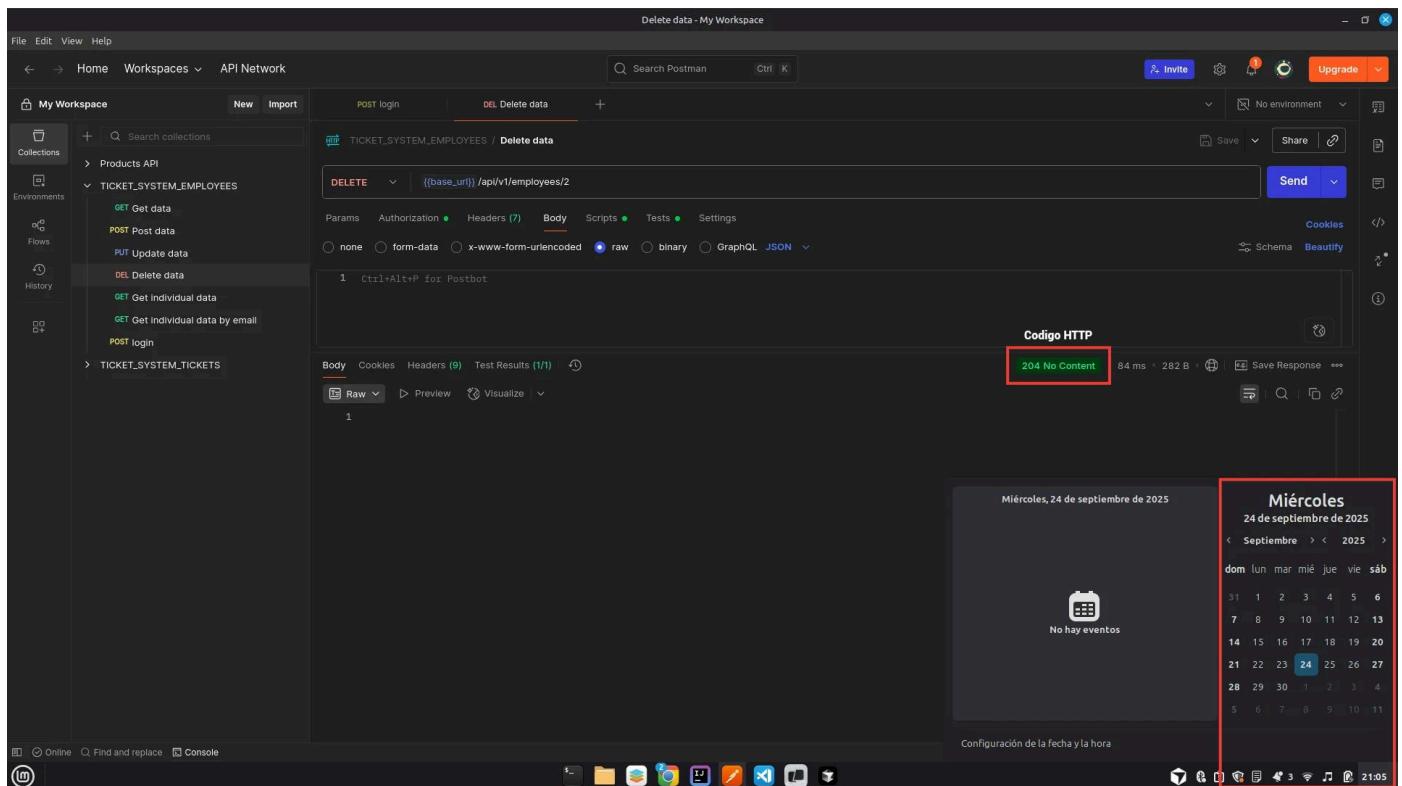
The "Send" button is highlighted in red. The calendar window to the right remains the same, showing the date as Wednesday, September 24, 2025.

## Caso de Prueba 5: Eliminar empleado existente

- **Endpoint:** /api/v1/employees/{id}
- **Método HTTP:** DELETE
- **Descripción:** Eliminar un empleado del sistema.
- **Parámetros:**
  - id: 2
- **Código de Respuesta Esperado:** 204 No Content
- **Respuesta esperada:** Vacía.
- **Prueba exitosa si:** Devuelve 204 No Content y el empleado deja de existir en el sistema.
- **Pasos de Ejecución de prueba:**
  - a) Agregar el ID de un empleado en concreto para eliminarlo:

The screenshot shows the Postman application interface. In the center, there's a collection named 'TICKET\_SYSTEM\_EMPLOYEES' containing several API endpoints. One endpoint is selected, showing a 'DELETE' method and the URL {{base\_url}} /api/v1/employees/2. The 'Body' tab is selected, showing 'raw' JSON: { "id": 2 }. To the right of the URL field, a 'Send' button is highlighted with a red box. Below the URL field, there are tabs for Params, Authorization, Headers, Body, Scripts, Tests, and Settings. The Body tab has a 'raw' radio button selected. At the bottom left, there's a 'Response' section with an illustration of an astronaut launching a rocket. On the right side, there's a calendar overlay titled 'Miércoles, 24 de septiembre de 2025'. The calendar shows the month of September 2025, with the date 24 highlighted. The status bar at the bottom right shows the date as 'Miércoles, 24 de septiembre de 2025' and the time as '21:04'.

- b) Enviar la petición y el código 204:



## Caso de Prueba 6: Error al eliminar empleado inexistente

- **Endpoint:** /api/v1/employees/{id}
- **Método HTTP:** DELETE
- **Descripción:** Intentar eliminar un empleado que no existe.
- **Parámetros:**
  - id: 999
- **Código de Respuesta Esperado:** 404 Not Found
- **Respuesta esperada:**

```
{
  "error": "Empleado no encontrado."
}
```

- **Prueba exitosa si:** Devuelve 404 Not Found.
- **Pasos de Ejecución de prueba:**
  - Agregar el ID de un empleado inexistente en el sistema para eliminarlo:

The screenshot shows the Postman interface with a dark theme. A DELETE request is selected with the URL `{base_url}/api/v1/employees/999`. The 'Send' button is highlighted with a red box. The response area shows a calendar for Wednesday, September 24, 2025, with the date 24 highlighted. The status bar at the bottom indicates 'Miércoles, 24 de septiembre de 2025'.

b) Enviar la petición y devuelve el código 404:

The screenshot shows the Postman interface with a dark theme. A DELETE request is selected with the URL `{base_url}/api/v1/employees/999` and 'Bearer Token' as the auth type. The token field contains a redacted string. The response area shows a 404 Not Found error with the message "Empleado no encontrado con ID: 999". The status bar at the bottom indicates 'Miércoles, 24 de septiembre de 2025'.

## Caso de Prueba 7: Obtener empleado por ID

- **Endpoint:** /api/v1/employees/{id}
- **Método HTTP:** GET
- **Descripción:** Consultar los datos de un empleado específico.

- **Parámetros:**

- id: 1

- **Código de Respuesta Esperado:** 200 OK

- **Respuesta esperada:**

```
{
  "id":1,
  "name":"Admin",
  "email":"admin@acme.com",
  "role":"ADMIN","department":"IT"
}
```

- **Prueba exitosa si:** Devuelve 200 OK con los datos correctos.

- **Pasos de Ejecución de prueba:**

- Agregar el ID de un empleado existente en el sistema para obtener su información:

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections like 'My Workspace', 'TICKET\_SYSTEM\_EMPLOYEES' (which is expanded), and 'TICKET\_SYSTEM\_TICKETS'. In the main area, a request is being prepared for the 'GET Individual data' endpoint under 'TICKET\_SYSTEM\_EMPLOYEES'. The 'Method HTTP' dropdown is set to 'GET', and the 'Endpoint' field contains the URL: `http://{{base_url}}/api/v1/employees/1`. A red box highlights the 'Send' button at the bottom right of the request panel. Below the request, there's a 'Response' section with a small illustration of an astronaut launching a rocket. To the right of the response, a calendar is displayed for Wednesday, September 24, 2025. The calendar shows the month of September with days numbered from 1 to 30. The date '24' is highlighted with a red box. A red box also highlights the entire calendar area. At the bottom of the screen, the Windows taskbar is visible with various icons.

- Enviar la petición y devuelve la información del empleado con el código 200:

The screenshot shows the Postman interface with a collection named 'TICKET\_SYSTEM\_EMPLOYEES'. A GET request is made to `/api/v1/employees/1`. The response body is a JSON object:

```

1 {
2   "id": 1,
3   "name": "Admin",
4   "email": "admin@acme.com",
5   "role": "ADMN",
6   "department": "IT"
7 }
  
```

The status code is 200 OK. To the right, there is a calendar view for September 2025, showing the date 24 highlighted.

## Caso de Prueba 8: Error al consultar empleado inexistente por ID

- Endpoint:** /api/v1/employees/{id}
- Método HTTP:** GET
- Descripción:** Intentar obtener un empleado que no existe.
- Parámetros:**
  - id: 999
- Código de Respuesta Esperado:** 404 Not Found
- Respuesta esperada:**

```
{
  "error": "Empleado no encontrado con ID: 999"
}
```

- Prueba exitosa si:** Devuelve 404 Not Found.
- Pasos de Ejecución de prueba:**
  - Agregar el ID de un empleado inexistente en el sistema para obtener su información:

Get Individual data - My Workspace

POST Login    GET Get individual data

**Endpoint**

Method HTTP: GET    URL: {{base\_url}} /api/v1/employees/999

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.

Response

Click Send to get a response.

Miércoles, 24 de septiembre de 2025

Calendar view showing No hay eventos (No events).

Configuración de la fecha y la hora

b) Enviar la petición y devuelve el código 404:

Get Individual data - My Workspace

POST Login    GET Get individual data

**Endpoint**

Method HTTP: GET    URL: {{base\_url}} /api/v1/employees/999

Params    Authorization    Headers (7)    Body    Scripts    Tests    Settings

Auth Type: Bearer Token

Token: [REDACTED]

The authorization header will be automatically generated when you send the request. Learn more about **Bearer Token** authorization.

Body    Cookies    Headers (1)    Test Results    Respuesta obtenida

Respuesta obtenida

Código HTTP: 404 Not Found

136 ms · 389 B · Save Response

1    {  
2    |    "message": "Empleado no encontrado con ID: 999"  
3    }

Miércoles, 24 de septiembre de 2025

Calendar view showing No hay eventos (No events).

Configuración de la fecha y la hora

## Caso de Prueba 9: Listar todos los empleados

- **Endpoint:** /api/v1/employees
- **Método HTTP:** GET
- **Descripción:** Obtener la lista de todos los empleados registrados.

- **Parámetros:** Ninguno.
- **Código de Respuesta Esperado:** 200 OK
- **Respuesta esperada:**

```
[
  {
    "id": 1,
    "name": "Admin",
    "email": "admin@acme.com",
    "role": "ADMIN",
    "department": "IT"
  },
  {
    "id": 2,
    "name": "Juan Pérez",
    "email": "juan.perez@ecma.com",
    "role": "USER",
    "department": "Soporte"
  }
]
```

- **Prueba exitosa si:** Devuelve 200 OK con una lista válida.
- **Pasos de Ejecución de prueba:**
  - a) Escoger la ruta respectiva para obtener todos los empleados registrados:

The screenshot shows the Postman interface with a collection named 'TICKET\_SYSTEM\_EMPLOYEES'. A GET request is selected with the URL `{base_url}/api/v1/employees`. The 'Send' button at the top right is highlighted with a red box.

b) Enviar la petición y devuelve la información de todos empleado con el código 200:

The screenshot shows the Postman interface after sending the request. The response body is displayed in JSON format, showing two employees with IDs 1 and 2. The status code '200 OK' is highlighted with a red box.

```

1  [
2   {
3     "id": 1,
4     "name": "Admin",
5     "email": "admin@acme.com",
6     "role": "ADMIN",
7     "department": "IT"
8   },
9   {
10    "id": 2,
11    "name": "Juan Pérez",
12    "email": "juan.perez@acme.com",
13    "role": "USER",
14    "department": "Soporte"
15  }
16 ]
  
```

**Response Body:**

**Código HTTP:**

## Caso de Prueba 10: Obtener empleado por email

- **Endpoint:** /api/v1/employees/email/{email}
- **Método HTTP:** GET
- **Descripción:** Consultar un empleado a partir de su correo electrónico.

- **Parámetros:**

- email: juan.perez@ecma.com

- **Código de Respuesta Esperado:** 200 OK

- **Respuesta esperada:**

```
{
  "id": 2,
  "name": "Juan Pérez",
  "email": "juan.perez@ecma.com",
  "role": "USER",
  "department": "Soporte"
}
```

- **Prueba exitosa si:** Devuelve 200 OK con los datos correctos.

- **Pasos de Ejecución de prueba:**

- Agrear el correo electrónico por la ruta de un empleado registrado:

The screenshot shows the Postman interface with a successful API call. The collection is 'TICKET\_SYSTEM\_EMPLOYEES' and the endpoint is 'GET Get individual data'. The URL is set to `{{base_url}}/api/v1/employees/email/juan.perez@ecma.com`. The 'Send' button is highlighted with a red box. The response pane shows the JSON data from the previous code block. A calendar overlay is visible at the bottom right, showing the date as 'Jueves, 25 de septiembre de 2025'.

- Enviar la petición y devuelve la información del empleado con el código 200:

The screenshot shows the Postman interface with a collection named 'My Workspace'. A specific test case titled 'Get Individual data' is selected. The request method is GET, and the URL is `"/api/v1/employees/email/juan.perez@ecma.com"`. The response status is 200 OK, and the JSON body contains the following data:

```

1 {
2   "id": 2,
3   "name": "Juan Pérez",
4   "email": "juan.perez@ecma.com",
5   "role": "USER",
6   "department": "Soporte"
7 }

```

A red box highlights the JSON response body. In the bottom right corner, there is a calendar view for September 25, 2025, with a red box highlighting the date 25.

## Caso de Prueba 11: Error al obtener empleado inexistente por email

- Endpoint:** /api/v1/employees/email/{email}
- Método HTTP:** GET
- Descripción:** Intentar obtener un empleado con un correo no registrado.
- Parámetros:**
  - email: noexiste@email.com
- Código de Respuesta Esperado:** 404 Not Found
- Respuesta esperada:**

```
{
  "error": "Empleado no encontrado con el email: noexiste@email.com"
}
```

- Prueba exitosa si:** Devuelve 404 Not Found.
- Pasos de Ejecución de prueba:**
  - Agregar el correo electrónico por la ruta de un empleado inexistente:

The screenshot shows the Postman interface with a collection named 'TICKET\_SYSTEM\_EMPLOYEES'. A GET request is selected with the URL {{base\_url}} /api/v1/employees/email/noexiste@email.com. The 'Send' button is highlighted with a red box.

b) Enviar la petición y devuelve el error con el código 404:

The screenshot shows the Postman interface with the same collection and request. The response body is displayed in JSON format, showing a single key-value pair: "message": "Empleado no encontrado con el email: noexiste@email.com". The status code '404 Not Found' is highlighted with a red box.

## - 6.2. Ticket

### Caso de Prueba 1: Crear un nuevo ticket

- **Endpoint:** /api/v1/tickets

- **Método HTTP:** POST
- **Descripción:** Registra un nuevo ticket en el sistema.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):**

```
{  
    "employeeId": 2,  
    "title": "Solicitud de nueva funcionalidad",  
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
    "categoryId": 1,  
    "priority": "BAJA"  
}
```

- **Código de Respuesta Esperado:** 201 Created
- **Cuerpo de la Respuesta Esperada:**

```
{  
    "id": 2,  
    "employeeId": 2,  
    "category": {  
        "id": 1,  
        "name": "TIC",  
        "description": "Se solicita agregar un botón de exportar datos en el reporte mensual"  
    },  
    "title": "Solicitud de nueva funcionalidad",  
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
    "priority": "BAJA",  
    "state": "ABIERTO",  
    "creationDate": "2025-10-04T12:52:21.984967",  
}
```

```
"closingDate": null
```

```
}
```

- **Prueba exitosa si:** Se recibe 201 Created y el ticket contiene todos los datos enviados más el ID generado y estado inicial.
- **Pasos de Ejecución de prueba:**
  - a) Agregar el cuerpo de un tiquete válido:

The screenshot shows the Postman application interface. In the center, there is a 'Post data' tab under a 'TICKET\_SYSTEM\_TICKETS' collection. The 'Endpoint' is set to 'POST {{base\_url}}/api/v1/tickets'. The 'Body' tab is selected, showing a JSON payload:

```
1 "employeeId": 1,
2 "title": "Solicitud de nueva funcionalidad",
3 "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
4 "categoryId": 1,
5 "priority": "BAJA"
```

A red box highlights this JSON payload. To the right, the 'Send' button is also highlighted with a red box. Below the request, there is a response placeholder with a cartoon character and a calendar view for Saturday, October 4, 2025, showing no events. The bottom status bar indicates the date as Sábado, 4 de octubre de 2025.

- b) Enviar la petición y devuelve el nuevo tiquete creado con el código 204:

The screenshot shows the Postman interface with a successful API call. The request URL is `POST {{base_url}}/api/v1/tickets`. The response body is highlighted with a red box and contains the following JSON:

```

1 {
2   "id": 2,
3   "employeeId": 2,
4   "category": {
5     "id": 1,
6     "name": "TIC",
7     "description": "Manejo de soporte para software o hardware"
8   },
9   "title": "Solicitud de nueva funcionalidad",
10  "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
11  "priority": "BAJA",
12  "state": "ABIERTO",
13  "creationDate": "2025-10-04T12:52:21.984967",
14  "closingDate": null
15 }

```

The response status is **201 Created** with a response time of 737 ms and 682 B. The response body is also highlighted with a red box and labeled **Respuesta obtenida**.

## Caso de Prueba 2: Error al crear ticket sin título

- Endpoint:** /api/v1/tickets
- Método HTTP:** POST
- Descripción:** Intentar crear un ticket sin título obligatorio.
- Parámetros:** Ninguno.
- Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- Cuerpo de la Solicitud (Body):**

```
{
  "employeeId": 2,
  "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
  "categoryId": 1,
  "priority": "BAJA"
}
```

- Código de Respuesta Esperado:** 400 Bad Request

- **Cuerpo de la Respuesta Esperada:**

```
{
  "error": "El título no puede ser nulo o vacío."
}
```

- **Prueba exitosa si:** La API rechaza la petición con 400 Bad Request indicando el error de validación.

- **Pasos de Ejecución de prueba:**

- Agrear el cuerpo de un tiquete pero sin título:

The screenshot shows the Postman application interface. In the center, there's a request configuration window for a POST method to the endpoint `http://{{base_url}}/api/v1/tickets`. The 'Body' tab is selected, showing a JSON payload with several fields: `"employeeId": 2, "description": "Se solicita agregar un botón de exportar datos en el reporte mensual", "categoryId": 1, "priority": "BAJA"`. Below the body, a red box highlights the text "Cuerpo de la petición". To the right of the body, the "Send" button is also highlighted with a red box. At the bottom of the request window, there's a note: "Click Send to get a response". In the bottom right corner of the screen, there's a small calendar window showing the date "Sábado, 4 de octubre de 2025" and a message "No hay eventos". A red box highlights this calendar window. The overall interface is dark-themed.

- Enviar la petición y devuelve el nuevo tiquete creado con el código 204:

The screenshot shows the Postman interface with a failed POST request to `/api/v1/tickets`. The response body contains the error message: `"title": "El título no puede ser nulo o vacío."`

### Caso de Prueba 3: Actualizar estado de un ticket

- **Endpoint:** `/api/v1/tickets/{id}/state`
- **Método HTTP:** PUT
- **Descripción:** Modificar el estado de un ticket existente.
- **Parámetros:**
  - id (path) → Identificador único del ticket.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):**

```
{
  "state": "EN_PROGRESO"
}
```

- **Código de Respuesta Esperado:** 200 OK
- **Cuerpo de la Respuesta Esperada:**

```
{
```

```

    "id": 2,
    "employeeId": 2,
    "category": {
        "id": 1,
        "name": "TIC",
        "description": "Manejo de soporte para software o hardware"
    }
    "title": "Solicitud de nueva funcionalidad",
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
    "priority": "ALTA",
    "state": "EN_PROCESO",
    "creationDate": "2025-10-04T12:52:21.984967",
    "closingDate": null
}

```

- **Prueba exitosa si:** El estado se actualiza correctamente y la API devuelve 200 OK con el ticket actualizado.
- **Pasos de Ejecución de prueba:**
  - Se envía en el path del endpoint el id del ticket y en el cuerpo de petición con el dato del nuevo estado:

The screenshot shows a Postman interface with a collection named 'My Workspace' containing various API endpoints for 'TICKET\_SYSTEM'. The 'PUT Update data' endpoint is selected. In the 'Body' tab, the JSON payload is set to raw and contains the following code:

```

1 "state": "EN_PROGRESO"
2
3 Cuerpo de solicitud

```

The 'Send' button is highlighted with a red box. Below the Postman window, a calendar overlay is visible, showing the date Saturday, October 4, 2025, with the number 4 highlighted in blue.

- Enviar la petición y devuelve el tiquete con el nuevo estado y con el código 200:

The screenshot shows the Postman interface with a collection named 'TICKET\_SYSTEM\_TICKETS'. A PUT request is selected with the URL `{base_url}/api/v1/tickets/2/state`. The request body contains the JSON `{"state": "EN_PROGRESO"}`. The response status is 200 OK, and the response body is displayed in a red-bordered box:

```

1 {
2   "id": 2,
3   "employeeId": 2,
4   "category": {
5     "id": 1,
6     "name": "TIC",
7     "description": "Manejo de soporte para software o hardware"
8   },
9   "title": "Solicitud de nueva funcionalidad",
10  "description": "Se solicita agregar un bot\u00f3n de exportar datos en el reporte mensual",
11  "priority": "BAJA",
12  "state": "EN_PROGRESO",
13  "creationDate": "2025-10-04T12:52:21.984967",
14  "closingDate": null
15 }

```

A red box highlights the response body with the label 'Respuesta obtenida'.

- **¿Prueba Superada?:** Sí, Aprobada
- **Observaciones:** El estado se actualiza y devuelve 200 OK.

#### Caso de Prueba 4: Actualizar estado de un tiquete con valor invalido

- **Endpoint:** /api/v1/tickets/{id}/state
- **Método HTTP:** PUT
- **Descripción:** Modificar el estado de un ticket existente pero con valor invalido.
- **Parámetros:**
  - id (path) → Identificador único del ticket.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):**

```
{
  "state": "VALOR_INVALIDO"
}
```

- **Código de Respuesta Esperado:** 400 Bad Request

- **Cuerpo de la Respuesta Esperada:**

```
{
```

```
    error:
```

```
}
```

- **Prueba exitosa si:** El estado no se actualiza correctamente y la API devuelve el código 400 Bad Request con el error arrojado.
- **Pasos de Ejecución de prueba:**
  - a) Se envía en el path del endpoint el id del ticket y en el cuerpo de petición con el dato del nuevo estado invalido:
  - b) Enviar la petición y devuelve el error dado con el código 400:
- **¿Prueba Superada?:** Sí, Aprobada
- **Observaciones:** El estado no se actualiza y devuelve 400 Bad Request.

#### **Caso de Prueba 5: Actualizar estado de un tiquete inexistente**

- **Endpoint:** /api/v1/tickets/{id}/state

- **Método HTTP:** PUT

- **Descripción:** Modificar el estado de un ticket inexistente.

- **Parámetros:**

- id (path) → Identificador único del ticket.

- **Headers:**

- Authorization: Bearer <token>
  - Content-Type: application/json

- **Cuerpo de la Solicitud (Body):**

```
{
```

```
    "state": "EN_PROGRESO"
```

```
}
```

- **Código de Respuesta Esperado:** 404 Not Found

- **Cuerpo de la Respuesta Esperada:**

```
{
```

```
    "error": "Tiquete no encontrado con ID: 99"
```

```
}
```

- **Prueba exitosa si:** El estado no se actualiza correctamente y la API devuelve el código 400 Not Found con el error arrojado.
- **Pasos de Ejecución de prueba:**
  - Se envía en el path del endpoint el id del ticket inexistente y en el cuerpo de petición con el dato del nuevo estado:

The screenshot shows a Postman collection named "TICKET\_SYSTEM\_TICKETS". A test named "PUT Update data" is selected. The request URL is `PUT {{base_url}}/api/v1/tickets/99/state`. The body is set to raw JSON with the value `{ "state": "EN_PROGRESO" }`. The response pane shows a success message: `Estado actualizado: EN_PROGRESO`.

- Enviar la petición y devuelve el error dado con el código 404:

The screenshot shows the same Postman setup as the previous one, but the test result is failing. The response body shows an error message: `{ "error": "Ticket no encontrado con ID: 99" }`.

- **¿Prueba Superada?:** Sí, Aprobada

- **Observaciones:** El estado no se actualiza y devuelve 404 Not Found.

## Caso de Prueba 6: Eliminar un ticket

- **Endpoint:** /api/v1/tickets/{id}
- **Método HTTP:** DELETE
- **Descripción:** Elimina un ticket existente por ID.
- **Parámetros:**
  - id (path) → ID del ticket a eliminar.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 204 No Content
- **Cuerpo de la Respuesta Esperada:** Vacío.
- **Prueba exitosa si:** El ticket es eliminado y la API devuelve 204 No Content.
- **Pasos de Ejecución de prueba:**
  - Se envía en el path del endpoint el id del ticket existente:

The screenshot shows the Postman application interface. A DELETE request is being prepared against the endpoint `{[base_url]}/api/v1/tickets/3`. The 'Send' button is highlighted with a red box. In the bottom right corner, there is a calendar interface showing the date **Sábado, 4 de octubre de 2025**, which is also highlighted with a red box. The calendar includes navigation arrows for the month and year, and a grid of days from Monday to Sunday with corresponding dates.

- Enviar la petición y devuelve el código 204:

The screenshot shows the Postman interface with a collection named "My Workspace". A specific DELETE request is selected, targeting the endpoint `/api/v1/tickets/3`. The response status is **204 No Content**, which is highlighted with a red box. A calendar window is overlaid on the bottom right, showing the date **Sábado, 4 de octubre de 2025**.

- **¿Prueba Superada?:** Sí, Aprobada
- **Observaciones:** El tiquete se elimina y devuelve 204 No Content.

### Caso de Prueba 7: Eliminar un ticket inexistente

- **Endpoint:** `/api/v1/tickets/{id}`
- **Método HTTP:** DELETE
- **Descripción:** Elimina un ticket inexistente por ID.
- **Parámetros:**
  - id (path) → ID del ticket a eliminar.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 404 Not Found.
- **Cuerpo de la Respuesta Esperada:**

```
{
    "error": "Tiquete no encontrado con el ID: 3"
}
```
- **Prueba exitosa si:** El ticket no es eliminado y la API devuelve 404 Not Found.
- **Pasos de Ejecución de prueba:**
  - a) Se envía en el path del endpoint el id del ticket inexistente:

The screenshot shows the Postman interface with a dark theme. On the left, the sidebar lists collections and environments. In the center, a DELETE request is being made to `[[base_url]]/api/v1/tickets/3`. The 'Body' tab is selected, showing the raw JSON body: `{}`. The 'Send' button at the top right is highlighted with a red box. The response pane shows a placeholder message: 'Click Send to get a response'. To the right, a calendar for Saturday, October 4, 2025, is displayed, showing 'No hay eventos'.

b) Enviar la petición y devuelve el código 404:

This screenshot is similar to the previous one but shows the result of sending the request. The response body is now a JSON object with an 'error' key: `{"error": "Tiquete no encontrado con ID: 3"}`. The 'Codigo HTTP' section shows '404 Not Found'. The rest of the interface is identical to the first screenshot.

- **¿Prueba Superada?:** Sí, Aprobada
- **Observaciones:** El tiquete no se elimina y devuelve 404 Not Found.

#### Caso de Prueba 8: Consultar un ticket existente por ID

- **Endpoint:** `/api/v1/tickets/{id}`

- **Método HTTP:** GET
- **Descripción:** Obtener la información de un ticket específico.
- **Parámetros:**

- id (path) → ID del ticket.

- **Headers:**

- Authorization: Bearer <token>

- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 200 OK
- **Cuerpo de la Respuesta Esperada:**

```
{
  "id": 2,
  "employeeId": 2,
  "category": {
    "id": 1,
    "name": "TIC",
    "description": "Manejo de soporte para software o hardware"
  },
  "title": "Solicitud de nueva funcionalidad",
  "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
  "priority": "BAJA",
  "state": "EN_PROCESO",
  "creationDate": "2025-10-04T12:52:21.984967",
  "closingDate": null
}
```

- **Prueba exitosa si:** La API devuelve 200 OK y los datos del ticket corresponden al ID solicitado.
- **Pasos de Ejecución de prueba:**
  - a) Se envía en el path del endpoint el id del ticket existente:

The screenshot shows the Postman interface with a collection named 'My Workspace'. A GET request is selected with the URL `{[base_url]}/api/v1/tickets/2`. The 'Send' button is highlighted with a red box. The response area is empty, showing a placeholder message: 'Click Send to get a response'. To the right, a calendar for Saturday, October 4, 2025, is displayed, and the status bar at the bottom right shows the date as 'Sábado, 4 de octubre de 2025'.

b) Enviar la petición y devuelve el código 200:

The screenshot shows the same Postman setup as above, but now the response body is visible. It contains a JSON object representing a ticket with ID 2. The status code '200 OK' is highlighted with a red box in the top right corner of the response area. The response body is as follows:

```

1  {
2    "id": 2,
3    "employeeId": 2,
4    "category": {
5      "id": 1,
6      "name": "TIC",
7      "description": "Manejo de soporte para software o hardware"
8    },
9    "title": "Solicitud de nueva funcionalidad",
10   "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
11   "priority": "BAJA",
12   "state": "EN_PROGRESO",
13   "creationDate": "2025-10-04T12:52:21.984967",
14   "closingDate": null
15 }

```

The status bar at the bottom right still shows 'Sábado, 4 de octubre de 2025'.

- **¿Prueba Superada?:** Sí, Aprobada
- **Observaciones:** El tiquete muestra su información y devuelve 200 OK.

### Caso de Prueba 9: Consultar un ticket inexistente por ID

- **Endpoint:** /api/v1/tickets/{id}

- **Método HTTP:** GET
- **Descripción:** Obtener la información de un ticket inexistente específico.
- **Parámetros:**

- id (path) → ID del ticket.

- **Headers:**

- Authorization: Bearer <token>

- **Cuerpo de la Solicitud (Body):** No aplica.

- **Código de Respuesta Esperado:** 404 Not Found.

- **Cuerpo de la Respuesta Esperada:**

```
{
}
```

```
"error": "Tiquete no encontrado con el ID: 3"
```

```
}
```

- **Prueba exitosa si:** La API devuelve 404 Not Found y un error arrojado.

- **Pasos de Ejecución de prueba:**

- Se envía en el path del endpoint el id del ticket existente:

The screenshot shows the Postman interface with a failed API request. The collection is 'My Workspace' and the endpoint is 'TICKET\_SYSTEM\_TICKETS / Get data'. The method is 'GET' and the URL is '({{base\_url}}) /api/v1/tickets/3'. The 'Send' button is highlighted with a red box. The response pane shows an error message: 'Sábado, 4 de octubre de 2025' and 'No hay eventos'.

- Enviar la petición y devuelve el código 200:

The screenshot shows a Postman workspace titled "Get data - My Workspace". In the center, there's a collection named "TICKET\_SYSTEM\_TICKETS" with a "GET Get data" endpoint selected. The URL is set to `"/{base_url}/api/v1/tickets/3"`. The "Headers" tab shows "Content-Type: application/json". The "Body" tab is selected, showing the raw JSON response: 

```
{}
{
  "error": "Tiquete no encontrado con ID: 3"
}
```

A red box highlights the error message "Tiquete no encontrado con ID: 3". Below the response, it says "Respuesta obtenida". To the right, a calendar interface shows Saturday, October 4, 2025. The calendar grid for October 2025 is visible, with the 4th highlighted in blue.

- **¿Prueba Superada?:** Sí, Aprobada
- **Observaciones:** El tiquete no muestra su información y devuelve 404 Not Found.

## Caso de Prueba 10: Listar todos los tickets

- **Endpoint:** /api/v1/tickets
- **Método HTTP:** GET
- **Descripción:** Obtener una lista con todos los tickets registrados en el sistema.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 200 OK
- **Cuerpo de la Respuesta Esperada:**

```
[
  {
    "id": 2,
    "employeeId": 2,
    "category": {
      "id": 1,
```

```

    "name": "TIC",
    "description": "Manejo de soporte para software o hardware"
},
{
    "title": "Solicitud de nueva funcionalidad",
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
    "priority": "BAJA",
    "state": "EN_PROGRESO",
    "creationDate": "2025-10-04T12:52:21.984967",
    "closingDate": null
},
{
    "id": 4,
    "employeeId": 2,
    "category": {
        "id": 1,
        "name": "TIC",
        "description": "Manejo de soporte para software o hardware"
    },
    "title": "Error de login",
    "description": "Se solicita corregir el sistema de login",
    "priority": "ALTA",
    "state": "ABIERTO",
    "creationDate": "2025-10-09T12:29:06.974869",
    "closingDate": null
}
]

```

- **Prueba exitosa si:** La API devuelve 200 OK y una lista con todos los tickets existentes en el sistema.
- **Pasos de Ejecución de prueba:**
  - a) Se ejecuta la ruta correspondiente para obtener todos los tiquetes:

The screenshot shows the Postman interface with a collection named 'TICKET\_SYSTEM\_TICKETS'. A specific endpoint 'GET Get data' is selected. The URL field contains '{[base\_url]}/api/v1/tickets'. The 'Send' button at the top right is highlighted with a red box.

b) Enviar la petición y devuelve el código 200:

The screenshot shows the Postman interface with the same setup. The response body is displayed in JSON format, showing a list of tickets. The entire JSON response is highlighted with a large red box.

```

The Anniversary
[{"id": 2, "employeeId": 2, "category": {"id": 1, "name": "TIC", "description": "Manejo de soporte para software o hardware"}, "title": "Solicitud de nueva funcionalidad", "description": "Se solicita agregar un bot\u00f3n de exportar datos en el reporte mensual", "priority": "BAJA", "state": "EN_PROGRESO", "creationDate": "2025-10-04T12:52:21.984967", "closingDate": null}, {"id": 4, "employeeId": 2, "category": {"id": 1, "name": "TIC", "description": "Manejo de soporte para software o hardware"}, "title": "Error de login", "description": "Se solicita corregir el sistema de login", "priority": "ALTA", "state": "ABIERTO", "creationDate": "2025-10-09T12:29:06.974869", "closingDate": null}
]
  
```

- **¿Prueba Superada?:** S\u00ed, Aprobada
- **Observaciones:** Se entregan todos los tiquetes registrados con su informaci\u00f3n y devuelve 200 OK.

### Caso de Prueba 11: Listar tickets con par\u00e1metros de consulta incorrecto

- **Endpoint:** /api/v1/tickets

- **Método HTTP:** GET
- **Descripción:** Verificar la respuesta del sistema cuando se realiza una solicitud con parámetros no definidos o estructura adicional al listar todos los tiquetes.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 404 Bad Request OK

- **Ejemplo de Respuesta:**

```
{
  "error": "Valor de parámetro inválido"
}
```

- **Prueba exitosa si:** La API devuelve 404 Bad Request y envía un error con el parámetro erróneo.
- **Pasos de Ejecución de Prueba:**
  - Enviar la solicitud:

The screenshot shows the Postman application interface. On the left, there's a sidebar with collections, environments, and a history section. The main workspace shows a collection named 'TICKET\_SYSTEM\_TICKETS' with several endpoints listed. One endpoint, 'GET /api/v1/tickets', is selected and highlighted with a red box. The 'Method' dropdown also has a red box around it, showing 'GET'. To the right of the endpoint details, there's a 'Send' button which is also highlighted with a red box. Below the endpoint details, there's a 'Body' tab with options like 'none', 'form-data', 'x-www-form-urlencoded', 'raw' (which is selected), 'binary', 'GraphQL', and 'JSON'. The 'raw' tab contains the text '1 Ctrl+Alt+P to Ask AI'. At the bottom of the workspace, there's a 'Response' tab and a 'Configuración de la fecha y la hora' (Date and Time Configuration) window. The date in the configuration window is set to 'Jueves, 9 de octubre de 2025' (Thursday, October 9, 2025). The date 9 is highlighted with a red box in the calendar grid.

- Enviar la solicitud:
- Respuesta 200 OK y que se devuelve la lista de tiquetes (vacía o con datos):

The screenshot shows the Postman interface with a successful API call. The URL is `GET {{base_url}}/api/v1/tickets`. The response body is a JSON array of tickets:

```

[{"id": 2, "employeeId": 2, "category": {"id": 1, "name": "TIC", "description": "Manejo de soporte para software o hardware"}, "title": "Solicitud de nueva funcionalidad", "description": "Se solicita agregar un bot\u00f3n de exportar datos en el reporte mensual", "priority": "BAJA", "state": "EN_PROGRESO", "creationDate": "2025-10-04T12:52:21.984967", "closingDate": null}, {"id": 4, "employeeId": 2, "category": {"id": 1, "name": "TIC", "description": "Manejo de soporte para software o hardware"}, "title": "Error de login", "description": "Se solicita corregir el sistema de login", "priority": "ALTA", "state": "ABIERTO", "creationDate": "2025-10-09T12:29:06.974869", "closingDate": null}]
  
```

The status bar at the bottom right shows the date as Jueves, 9 de octubre de 2025.

**¿Prueba Superada?:** No, Desaprobada

**Observaciones:** El sistema no genera un error 400 (Bad Request).

### Caso de Prueba 12: Listar tickets por estado

- Endpoint:** /api/v1/tickets/state/{state}
- M\u00f3dulo HTTP:** GET
- Descripci\u00f3n:** Obtener una lista de tickets filtrados por estado.
- Par\u00e1metros:**
  - state (path) → Estado del ticket (ej: ABIERTO, EN\_PROCESO, CERRADO).
- Headers:**
  - Authorization: Bearer <token>
- Cuerpo de la Solicitud (Body):** No aplica.
- C\u00f3digo de Respuesta Esperado:** 200 OK
- Cuerpo de la Respuesta Esperada:**

```
[
  {
    "id": 4,
    "employeeId": 2,
    "category": {
```

```

    "id": 1,
    "name": "TIC",
    "description": "Manejo de soporte para software o hardware"
},
{
    "title": "Error de login",
    "description": "Se solicita corregir el sistema de login",
    "priority": "ALTA",
    "state": "ABIERTO",
    "creationDate": "2025-10-09T12:29:06.974869",
    "closingDate": null
}
]

```

- **Prueba exitosa si:** La API devuelve 200 OK con solo los tickets que tienen el estado solicitado.
- **Pasos de Ejecución de prueba:**

a) Se ejecuta la ruta pasando el estado de los tiquetes que queremos consultar:

The screenshot shows the Postman interface with a collection named 'TICKET\_SYSTEM\_TICKETS'. A GET request is selected with the URL `{[base_url]}/api/v1/tickets/state/ABIERTO`. The 'Authorization' tab shows 'Bearer Token' selected, and a token value is present. The 'Send' button is highlighted with a red box. The response pane shows a calendar for October 2025 with the date 'Jueves, 9 de octubre de 2025' and a note 'No hay eventos'.

b) Enviar la petición y devuelve el código 200:

The screenshot shows the Postman interface with a collection named 'TICKET\_SYSTEM\_TICKETS'. A specific GET request is selected with the URL `:(base_url) /api/v1/tickets/state/ABIERTO`. The 'Authorization' header is set to 'Bearer Token'. The response body is displayed in JSON format, showing a single ticket record. The status bar at the bottom indicates a 200 OK response.

- **¿Prueba Superada?:** Sí, Aprobada
- **Observaciones:** Se entregan todos los tickets registrados con el estado requerido y devuelve 200 OK.

### Caso de Prueba 13: Listar tickets por estado incorrecto

- **Endpoint:** /api/v1/tickets/state/{state}
- **Método HTTP:** GET
- **Descripción:** Verificar la respuesta del sistema cuando se consulta un estado inválido del enum.
- **Parámetros:**
  - state (path) → Estado del ticket (Incorrecto).
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 404 Bad Request
- **Cuerpo de la Respuesta Esperada:**

```
{
  "error": "Valor inválido 'ESTADO_INCORRECTO' para el parámetro 'state'. Valores válidos: ABIERTO, EN_PROGRESO, RESUELTO, CERRADO"
}
```
- **Prueba exitosa si:** La API devuelve 404 Not Found y un error indicando donde fallo.

- **Pasos de Ejecución de prueba:**

- Se ejecuta la ruta pasando el estado incorrecto de los tiquetes que queremos consultar:

The screenshot shows the Postman interface with a collection named 'My Workspace'. A new POST request is being created for the endpoint `/api/v1/tickets/state/ESTADO_INCORRECTO`. The 'Send' button is highlighted with a red box.

- Enviar la petición y devuelve el código 404:

The screenshot shows the Postman interface after sending the request. The response body is displayed with the following JSON content:

```
{
  "error": "Valor inválido 'ESTADO_INCORRECTO' para el parámetro 'state'. Valores válidos: ABIERTO, EN_PROGRESO, RESUELTO, CERRADO"
}
```

The status code is shown as **Código HTTP: 400 Bad Request**. The calendar window on the right shows October 9, 2025, with a red box highlighting the date.

- **¿Prueba Superada?:** Sí, Aprobada

- **Observaciones:** El sistema devuelve 404 Not Found cuando no existen tiquetes asociados al estado consultado.

## Caso de Prueba 13: Consultar comentarios de un ticket

- **Endpoint:** GET /api/v1/tickets/{id}/comments
- **Método HTTP:** GET
- **Descripción:** Obtener todos los comentarios asociados a un ticket.
- **Parámetros:**

- id (path) → ID del ticket.

- **Headers:**

- Authorization: Bearer <token>

- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 200 OK
- **Cuerpo de la Respuesta Esperada:**

```
[  
 {  
   "id": 1,  
   "ticketId": null,  
   "employee": {  
     "id": 2,  
     "name": "Juan Pérez",  
     "email": "juan.perez@ecma.com",  
     "role": "USER",  
     "department": "Soporte"  
   },  
   "text": "El problema fue revisado y se está realizando la corrección correspondiente.",  
   "creationDate": "2025-10-09T16:54:11.144683"  
 },  
 {  
   "id": 2,  
   "ticketId": null,  
   "employee": {  
     "id": 2,  
     "name": "Juan Pérez",  
     "email": "juan.perez@ecma.com",  
     "role": "USER",  
     "department": "Soporte"  
   }  
 }]
```

```

        "email": "juan.perez@ecma.com",
        "role": "USER",
        "department": "Soporte"
    },
    "text": "Se validó la información del ticket y se confirmó que el error persiste en el módulo de autenticación.",
    "creationDate": "2025-10-09T16:57:02.357183"
}
]

```

- **Prueba exitosa si:** La API devuelve 200 OK con todos los comentarios relacionados al ticket indicado.
- **Ejecución de la prueba y resultado obtenido:**
  - Se envía en el path del endpoint el id del ticket:

The screenshot shows the Postman interface with a successful API call to `http://localhost:8080/api/v1/2/comments`. The 'Send' button is highlighted in blue. The response window displays a calendar for October 2025, with the date `Jueves, 9 de octubre de 2025` highlighted. The calendar shows the following days:

dom	lun	mar	mié	jue	vie	sáb
28	29	30	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	31	1
2	3	4	5	6	7	8

- Nos arroja la lista de comentarios relacionados con el tiquete junto con el código 200:

The screenshot shows the Postman interface with a successful API call. The response body is displayed as JSON, and the date picker in the bottom right corner is also highlighted.

## Caso de Prueba 15: Obtener registros por ticket

- Endpoint:** GET /api/tickets/{id}/tickets-records
- Método HTTP:** GET
- Descripción:** Consultar el historial de cambios asociados a un ticket.
- Parámetros:**
  - ticketId (path) → ID del ticket a consultar.
  - from (query) → Fecha de inicio.
  - to (query) → Fecha de fin.
- Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- Cuerpo de la Solicitud (Body):** No aplica.
- Código de Respuesta Esperado:** 200 OK
- Cuerpo de la Respuesta Esperada:**

```
[  
{  
  "id": 45,  
  "ticketId": 123,  
  "employee": {  
    "id": 2,  
    "name": "Juan Pérez",  
    "email": "juan.perez@ecma.com",  
    "role": "USER",  
    "department": "Soporte"  
  },  
  "text": "El problema fue revisado y se está realizando la corrección correspondiente.",  
  "creationDate": "2025-10-09T16:54:11.144683"  
},  
{  
  "id": 2,  
  "ticketId": null,  
  "employee": {  
    "id": 2,  
    "name": "Juan Pérez",  
    "email": "juan.perez@ecma.com",  
    "role": "USER",  
    "department": "Soporte"  
  },  
  "text": "Se validó la información del ticket y se confirmó que el error persiste en el módulo.",  
  "creationDate": "2025-10-09T16:57:02.357183"  
}]
```

```

    "previousState": "Abierto",
    "nextState": "En progreso",
    "changedDate": "2025-09-10T14:30:00"
},
{
    "id": 46,
    "ticketId": 123,
    "previousState": "En progreso",
    "nextState": "Cerrado",
    "changedDate": "2025-09-11T10:15:00"
}
]

```

- Prueba exitosa si:** La API devuelve código 200 y el historial de cambios del ticket en formato de lista.
- Ejecución de la prueba y resultado obtenido:**
  - Se envía en el path del endpoint el id del ticket y si quieras por rango de fecha pones los 2 parametros de consulta “from” y “to”:

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'My Workspace' containing collections like 'API TRACKS', 'API USUARIOS SPRINGBOOT', 'Cloud API', 'FinanzApp API', 'Portfolio', 'Portfolio Backend Java', and 'Ticket System'. Under 'Ticket System', there are 'Ticket' and 'Employee' sections with various methods: 'POST Create ticket', 'GET Obtener ticket', 'PUT Actualizar estado ticket y regis...', 'GET Listar ticketRecords de ticket', 'POST Create employee', 'PUT Update employee', 'POST Login', and 'GET Category'. The main workspace shows a 'Listar ticketRecords de ticket' collection. A specific 'Endpoint' is selected: 'Método HTTP: GET' with the URL path {{baseURL}} /api/v1/tickets/2/tickets-record?from=2025-09-10&to=2025-09-23. The 'Params' tab shows two optional query parameters: 'from' set to '2025-09-10' and 'to' set to '2025-09-23'. The 'Send' button is highlighted with a red box. The bottom right corner of the interface shows the date and time: '8:16 p. m. 23/09/2025'.

- Respuesta de los ticketRecords de un ticket en un rango de fecha determinado

The screenshot shows the Postman interface with a successful API call. The URL is `{{baseUrl}}/api/v1/tickets/1/tickets-record?from=2025-09-10&to=2025-09-23`. The response code is `200 OK`. The response body is a JSON array containing three objects, each representing a ticket state change:

```

[ {
    "id": 1,
    "previousState": "ABIERTO",
    "nextState": "EN_PROGRESO",
    "changedDate": "2025-09-23T15:23:33.830224"
},
{
    "id": 2,
    "previousState": "ABIERTO",
    "nextState": "EN_PROGRESO",
    "changedDate": "2025-09-23T15:27:43.809354"
},
{
    "id": 4,
    "previousState": "EN_PROGRESO",
    "nextState": "RESUELTO",
    "changedDate": "2025-09-23T19:40:03.000181"
}
]

```

- Muestra de los datos de la respuesta del endpoint:

```
[ { "id": 1, "previousState": "ABIERTO", "nextState": "EN_PROGRESO", "changedDate": "2025-09-23T15:23:33.830224" },
```

```
{ "id": 2, "previousState": "ABIERTO", "nextState": "EN_PROGRESO", "changedDate": "2025-09-23T15:27:43.809354" },
```

```
{ "id": 4, "previousState": "EN_PROGRESO", "nextState": "RESUELTO", "changedDate": "2025-09-23T19:40:03.000181" },
```

```
{ "id": 5, "previousState": "RESUELTO", "nextState": "EN_PROGRESO", "changedDate": "2025-09-23T19:41:23.893695" },
```

```
{ "id": 6, "previousState": "EN_PROGRESO", "nextState": "RESUELTO", "changedDate": "2025-09-23T19:56:50.02772" } ]
```

- ¿Prueba Superada?: Sí, Aprobada

- Observaciones:** Devuelve lista de todos los registros de cambio de estado de un ticket pero sin el "ticketId".

## Caso de Prueba 10: Error al obtener registros de un ticket inexistente

- Endpoint:** GET /api/tickets/{id}/tickets-records
- Método HTTP:** GET
- Descripción:** Consultar el historial de un ticket que no existe.
- Parámetros:**
  - ticketId (path) → ID inexistente.
  - from (query) → Fecha de inicio.

- o to (query) → Fecha de fin.
- **Headers:**
    - o Authorization: Bearer <token>
    - o Content-Type: application/json
  - **Cuerpo de la Solicitud (Body):** No aplica.
  - **Código de Respuesta Esperado:** 404 Not Found
  - **Cuerpo de la Respuesta Esperada:**

```
{
  "error": "Ticket no encontrado"
}
```

- **Prueba exitosa si:** La API devuelve código 404 y un mensaje indicando que el ticket no existe.
- **Ejecución de la prueba y resultado obtenido:**
  - Se pone un id de ticket inexistente en el path del endpoint

The screenshot shows the Postman interface with the following details:

- Método HTTP:** GET, Endpoint: {{baseURL}} /api/v1/tickets/3/tickets-records?from=2025-09-10&to=2025-09-23
- Query Params:**

Key	Value	Description	Bulk Edit
from	2025-09-10		
to	2025-09-23		
- Código HTTP:** 404 Not Found
- Respuesta:** {"message": "Ticket no encontrado con ID: 3"}
- Timestamp:** 8:48 p.m. 23/09/2025

- **¿Prueba Superada?:** Sí, Aprobada
- **Observaciones:** Devuelve error 404 porque no encontró un tiquete con ese ID

### Caso de Prueba 11: Error al consultar rango de fechas inválido

- **Endpoint:** GET /api/tickets/{id}/tickets-records/range?from=2025-99-99&to=2025-09-15
- **Método HTTP:** GET
- **Descripción:** Intentar consultar registros con una fecha mal formada.

- **Parámetros:**

- from inválido.
- to válido.

- **Headers:**

- Authorization: Bearer <token>
- Content-Type: application/json

- **Cuerpo de la Solicitud (Body):** No aplica.

- **Código de Respuesta Esperado:** 400 Bad Request

- **Cuerpo de la Respuesta Esperada:**

{

"error": "Parámetros de fecha inválidos"

}

- **Prueba exitosa si:** La API devuelve código 400 con un mensaje indicando error en el formato de fecha.

- **Ejecución de la prueba y resultado obtenido:**

- Se pone un id de ticket inexistente en el path del endpoint

The screenshot shows a Postman collection named 'My Workspace' containing several API endpoints. The selected endpoint is 'Listar ticketRecords de ticket' (Method: GET). The URL is {{baseURL}} /api/v1/tickets/1/tickets-record?from=2025-11-11x&to=2025-09-23. In the 'Params' tab, the 'from' parameter is set to '2025-11-11x'. The response status is 400 Bad Request, and the response body is: { "error": "Formato de fecha inválido. Usa el formato yyyy-MM-dd." }. The timestamp at the bottom right is 1:12 p.m. 8/10/2025.

- **¿Prueba Superada?:**Aprobada

- **Observaciones:** Parece validar el formato de la fecha pero en vez de devolver error 400 devuelve error de no autorizado, hay conflicto entre la validación de fecha de los parametros de consulta y la capa de seguridad.

### - 6.3 Category

#### Caso de Prueba 1: Listar todas las categorías

- **Endpoint:** GET /api/v1/categories
- **Método HTTP:** GET
- **Descripción:** Devuelve una lista con todas las categorías registradas en la base de datos.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 200 OK
- **Cuerpo de la Respuesta Esperada (muestra):**

```
[  
 {  
   "id": 3,  
   "name": "A",  
   "description": "Problemas relacionados con conectividad"  
 },  
 {  
   "id": 4,  
   "name": "Red",  
   "description": "Problemas relacionados con conectividad"  
 }]
```

- **Prueba exitosa si:** La API devuelve 200 OK con la lista completa de categorías.
- **Pasos de ejecución de la prueba**
  - A) Agregar la petición para listar las categorías

The screenshot shows the Postman application interface. In the top navigation bar, 'Home' and 'Workspaces' are selected. The main area displays an 'Endpoint' named 'Springboot / Eventos / post'. A red box highlights the 'GET' method and the URL 'http://localhost:8080/api/v1/categories'. Below the URL, the 'Body' tab is selected, showing 'raw' JSON input. The status bar at the bottom indicates a successful response: '200 OK' with a duration of '72 ms' and a size of '647 B'. The bottom right corner shows the date and time: '5:30 p.m. 8/10/2025'.

## B) Enviar petición y obtener el código 200

This screenshot shows the same Postman session after sending the request. The 'Body' tab is highlighted with a red box, displaying the JSON response. The response body contains three categories: { "id": 3, "name": "A", "description": "Problemas relacionados con conectividad"}, { "id": 4, "name": "Red", "description": "Problemas relacionados con conectividad"}, and { "id": 5, "name": "Rede", "description": "Problemas relacionados con conectividad"}. To the right of the response body, the text 'Cuerpo de la respuesta' is written. Above the response body, the status bar shows '200 OK' with a duration of '72 ms' and a size of '647 B'. The bottom right corner again shows the date and time: '5:33 p.m. 8/10/2025'.

### Respuesta obtenida (muestra):

```
[  
 {  
   "id": 3,  
   "name": "A",  
   "description": "Problemas relacionados con conectividad"  
 }]
```

```

    "description": "Problemas relacionados con conectividad"
},
{
    "id": 4,
    "name": "Red",
    "description": "Problemas relacionados con conectividad"
}
]

```

**Prueba superada?** Si, aprobada.

### Caso de Prueba 2: Obtener una categoría por ID

- **Endpoint:** GET /api/v1/categories/{id}
- **Método HTTP:** GET
- **Descripción:** Obtiene una categoría según su ID.
- **Parámetros:**
  - id (path) → ID de la categoría.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:**
  - 200 OK si se encuentra la categoría.
- **Cuerpo de la Respuesta Esperada (ejemplo éxito):**

```
{
    "id": 4,
    "name": "Red",
    "description": "Problemas relacionados con conectividad"
}
```

- **Prueba exitosa si:** La API devuelve 200 OK con la categoría correspondiente.

- **Pasos de ejecución de la prueba**

C) Agregar la petición con el path para obtener la categoría deseada

The screenshot shows the Postman application interface. In the center, there is a 'Método HTTP' section with a dropdown menu set to 'GET'. Next to it is a red box highlighting the 'Endpoint' field, which contains the URL 'http://localhost:8080/api/v1/categories/4'. Below this, under the 'Body' tab, there is a single line of text: '1 Ctrl+Alt+P to Ask AI'. At the bottom right of the interface, there is a timestamp '6:05 p.m.' and a date '8/10/2025'.

D) Enviar petición y obtener el código 200

The screenshot shows the Postman application interface after sending the request. The 'Body' tab is selected, and the response is displayed as JSON. The JSON object has one key-value pair: 'id': 4. Below the JSON, the text 'Cuerpo de la respuesta' is visible. At the top right, the status code '200 OK' is highlighted with a red box. At the bottom right, there is a timestamp '6:08 p.m.' and a date '8/10/2025'.

```
{
  "id": 4
}
```

**Respuesta obtenida (muestra):**

```
{
  "id": 4,
  "name": "Red",
```

- "description": "Problemas relacionados con conectividad"
  - }
- **Prueba superada?** Si, aprobada.

### Caso de Prueba 3: Error al obtener una categoría por ID no existente

- **Endpoint:** GET /api/v1/categories/{id}
- **Método HTTP:** GET
- **Descripción:** Obtiene una categoría según su ID.
- **Parámetros:**
  - id (path) → ID de la categoría.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:**
  - 200 OK si se encuentra la categoría.
- **Cuerpo de la Respuesta Esperada (ejemplo éxito):**

{

"error": "No se ha encontrado la categoría con id: 999"

}

- **Prueba exitosa si:** La API devuelve 404 NOT FOUND con el error correspondiente.
- **Pasos de ejecución de la prueba**
  - E) Agregar la petición con el path para obtener la categoría deseada

The screenshot shows the Postman interface with a collection named "Springboot / Eventos". A new endpoint named "post" has been created. The "Method" dropdown is set to "GET" and the "Endpoint" field contains the URL "http://localhost:8080/api/v1/categories/999". The "Body" tab is selected, showing the raw JSON response: { "error": "No se encontro la categoria con el id: 999" }. The status bar at the bottom indicates a successful "200 OK" response with a duration of 815 ms and a size of 598 B. The timestamp in the bottom right corner is 4:33 p.m. on 10/10/2025.

## F) Enviar petición y obtener el código 200

The screenshot shows the Postman interface with the same setup as the previous one. A red box highlights the "GET" method and the URL "http://localhost:8080/api/v1/categories/999". In the "Body" tab, the JSON response is displayed: { "error": "No se encontro la categoria con el id: 999" }. The status bar at the bottom shows a "404 Not Found" error with a duration of 577 ms and a size of 395 B. The timestamp in the bottom right corner is 4:33 p.m. on 10/10/2025.

### Respuesta obtenida (muestra):

```
{  
  "error": "No se ha encontrado la categoría con id: 999"  
}
```

- Prueba superada? Si, aprobada.

#### Caso de Prueba 4: Crear una categoría

- **Endpoint:** POST /api/v1/categories
- **Método HTTP:** POST
- **Descripción:** Permite registrar una nueva categoría en el sistema.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):**

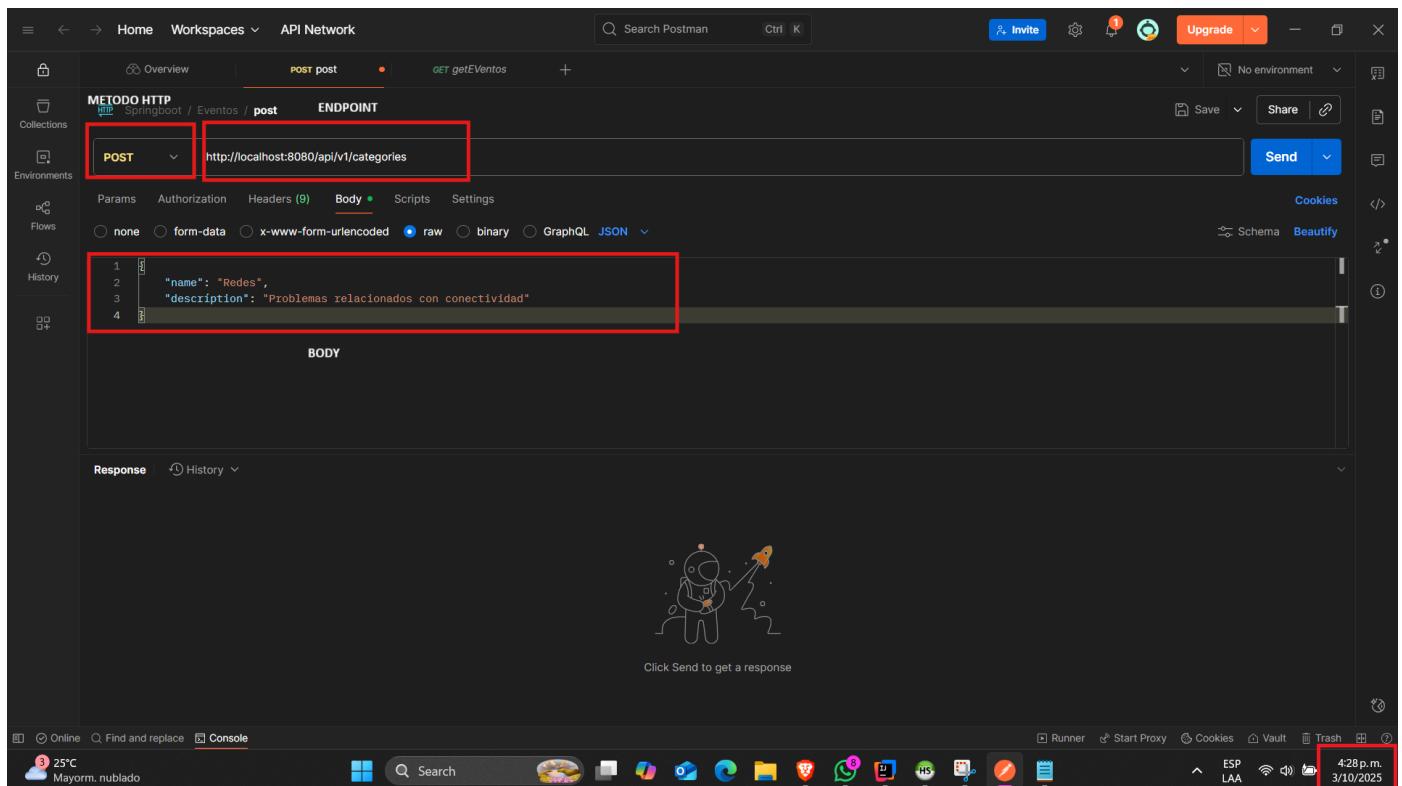
```
{  
  "name": "Redes",  
  "description": "Problemas relacionados con conectividad"  
}
```

- **Código de Respuesta Esperado:**
  - 200 Ok si la categoría se creó correctamente.
- **Cuerpo de la Respuesta Esperada (Ejemplo):**

```
{  
  "id": 1,  
  "name": "Redes",  
  "description": "Problemas relacionados con conectividad"  
}
```

- **Prueba exitosa si:** La API devuelve 200 Ok y los datos de la nueva categoría.
- **Pasos de ejecución de la prueba**
  - G) Agregar el cuerpo con la petición para agregar una nueva categoría

Screenshot of Postman interface showing a POST request to http://localhost:8080/api/v1/categories with a JSON body containing {"name": "Redes", "description": "Problemas relacionados con conectividad"}.



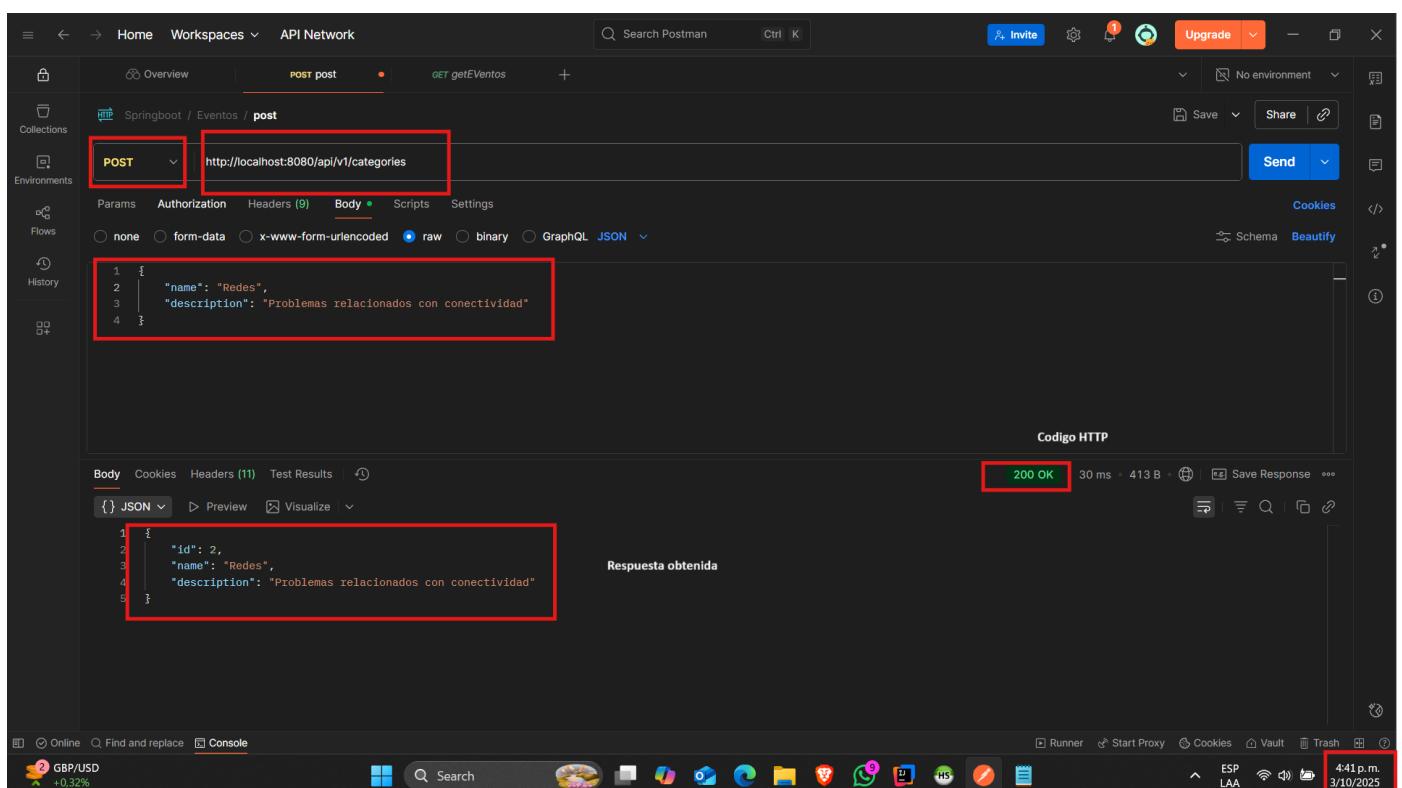
The screenshot shows the Postman interface with a dark theme. At the top, it says "Home Workspaces API Network". Below that, there's a search bar and some status icons. The main area shows a POST request to "http://localhost:8080/api/v1/categories". The "Body" tab is selected, showing a raw JSON payload:

```
1 {
2   "name": "Redes",
3   "description": "Problemas relacionados con conectividad"
4 }
```

At the bottom right of the interface, there's a timestamp: "4:28 p.m. 3/10/2025".

#### H) Enviar petición y obtener el código 200

Screenshot of Postman interface showing a successful POST request to http://localhost:8080/api/v1/categories, resulting in a 200 OK response with the category data.



The screenshot shows the Postman interface with a dark theme. At the top, it says "Home Workspaces API Network". Below that, there's a search bar and some status icons. The main area shows a POST request to "http://localhost:8080/api/v1/categories". The "Body" tab is selected, showing a raw JSON payload:

```
1 {
2   "name": "Redes",
3   "description": "Problemas relacionados con conectividad"
4 }
```

Below the request, the "Test Results" section shows a "200 OK" status with a timestamp: "4:41 p.m. 3/10/2025". The "Body" tab is expanded, showing the response data:

```
1 {
2   "id": 2,
3   "name": "Redes",
4   "description": "Problemas relacionados con conectividad"
5 }
```

At the bottom right of the interface, there's a timestamp: "4:41 p.m. 3/10/2025".

#### Respuesta obtenida:

```
{
  "id": 2,
  "name": "Redes",
  "description": "Problemas relacionados con conectividad"
```

```
}
```

**Prueba superada?** Si, aprobada.

#### Caso de Prueba 5: Error al crear una categoria con el nombre ya existente

- **Endpoint:** POST /api/v1/categories
- **Método HTTP:** POST
- **Descripción:** Intenta registrar una nueva categoria con un nombre ya existente.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):**

```
{
```

```
  "name": "Redes",  
  "description": "Problemas relacionados con conectividad"  
}
```

- **Código de Respuesta Esperado:**
  - 409 Conflict si ya existe una categoría con ese nombre.
- **Cuerpo de la Respuesta Esperada (Ejemplo):**

```
{  
  "error": "Ya existe una categoría con nombre: Redes"  
}
```

- **Prueba exitosa si:** La API devuelve 409 y el mensaje de error.
- **Pasos de ejecución de la prueba**
  - I) Agregar el cuerpo con la petición para agregar una nueva categoría

The screenshot shows the Postman interface with a POST request to the endpoint `http://localhost:8080/api/v1/categories`. The request body is set to raw JSON, containing the following data:

```
1 {  
2   "name": "Redes",  
3   "description": "Problemas relacionados con conectividad"  
4 }
```

#### J) Enviar petición y obtener el código 409

The screenshot shows the Postman interface with a POST request to the endpoint `http://localhost:8080/api/v1/categories`. The request body is set to raw JSON, containing the following data:

```
1 {  
2   "name": "Redes",  
3   "description": "Problemas relacionados con conectividad"  
4 }
```

The response code is **409 Conflict**, indicating that the operation failed because the resource already exists. The response body is:

```
{  
  "error": "Ya existe una categoría con nombre: Redes"  
}
```

#### Respuesta obtenida:

```
{  
  "error": "Ya existe una categoría con nombre: Redes"  
}
```

**Prueba superada? Si.**

## Caso de Prueba 6: Error al crear una categoria con un token invalido o de un rol distinto

- **Endpoint:** POST /api/v1/categories
- **Método HTTP:** POST
- **Descripción:** Intenta registrar una nueva categoria con un nombre ya existente.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):**

```
{  
  "name": "Redes",  
  "description": "Problemas relacionados con conectividad"  
}
```

- **Código de Respuesta Esperado:**
  - 401 Unauthorized.
- **Cuerpo de la Respuesta Esperada (Ejemplo):**

```
{  
  "error": "No tienes autorización para acceder a este recurso"  
}
```

- **Prueba exitosa si:** La API devuelve 401 y el mensaje de error.
- **Pasos de ejecución de la prueba**
  - K) Agregar el cuerpo con la petición para agregar una nueva categoría

The screenshot shows the Postman interface with a POST request to the endpoint `http://localhost:8080/api/v1/categories`. The request method is selected as `POST`, and the URL is specified. The `Body` tab is active, showing a raw JSON payload:

```
1 {  
2   "name": "Redes",  
3   "description": "Problemas relacionados con conectividad"  
4 }
```

## L) Enviar petición y obtener el código 401

The screenshot shows the Postman interface with the same POST request to `http://localhost:8080/api/v1/categories`. The response status is `401 Unauthorized`. The response body is displayed as:

```
{  
  "error": "No tienes autorización para acceder a este recurso"  
}
```

**Respuesta obtenida:**

```
{  
  "error": "No tienes autorización para acceder a este recurso"  
}
```

**Prueba superada? Si.**

## Caso de Prueba 7: Eliminar una categoría

- **Endpoint:** DELETE /api/v1/categories/{id}
- **Método HTTP:** DELETE
- **Descripción:** Elimina una categoría existente según su ID.
- **Parámetros:**
  - id (path) → ID de la categoría a eliminar.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:**
  - 204 No Content si se eliminó exitosamente.
- **Prueba exitosa si:** La API devuelve 204 No Content al eliminar la categoría indicada.
- **Pasos de ejecución de la prueba**
  - M) Se envia el path en el endpoint con el id de la categoria deseada a eliminar

The screenshot shows the Postman application interface. In the center, there's a 'DELETE' button and a URL input field containing 'http://localhost:8080/api/v1/categories/2'. The URL field is highlighted with a red box. The 'Body' tab is selected at the bottom. At the bottom right, there's a timestamp '6:34 p.m. 7/10/2025' which is also highlighted with a red box.

N) Enviar petición y obtener el código 204

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8080/api/v1/categories/2`. The response code is **204 No Content**, and the timestamp is **6:38 p.m. 7/10/2025**.

### Respuesta obtenida:

No aplica (respuesta 204)

Prueba superada? Si.

### Caso de Prueba 8: Error al eliminar una categoría no existente

- **Endpoint:** `DELETE /api/v1/categories/{id}`
- **Método HTTP:** `DELETE`
- **Descripción:** Trata de eliminar una categoría cuyo id no existe.
- **Parámetros:**
  - `id` (path) → ID de la categoría a eliminar.
- **Headers:**
  - `Authorization: Bearer <token>`
- **Cuerpo de la Solicitud (Body):**

```
{  
  "error": "No se ha encontrado la categoría con id: 999"  
}
```
- **Código de Respuesta Esperado:**
  - `404 Not Found` si no existe la categoría.
- **Prueba exitosa si:** La API devuelve `404 Not Found` al intentar eliminar la categoría indicada.
- **Pasos de ejecución de la prueba**

O) Se envia el path en el endpoint con el id de la categoria deseada a eliminar

The screenshot shows the Postman interface with a DELETE request to the endpoint `http://localhost:8080/api/v1/categories/999`. The response is a 204 No Content status with a duration of 923 ms and a size of 282 B. The raw response body is empty.

P) Enviar petición y obtener el código 404

The screenshot shows the Postman interface with a DELETE request to the endpoint `http://localhost:8080/api/v1/categories/999`. The response is a 404 Not Found status with a duration of 956 ms and a size of 398 B. The raw response body contains the JSON object: `{"error": "No se ha encontrado la categoría con id: 999"}`.

Respuesta obtenida:

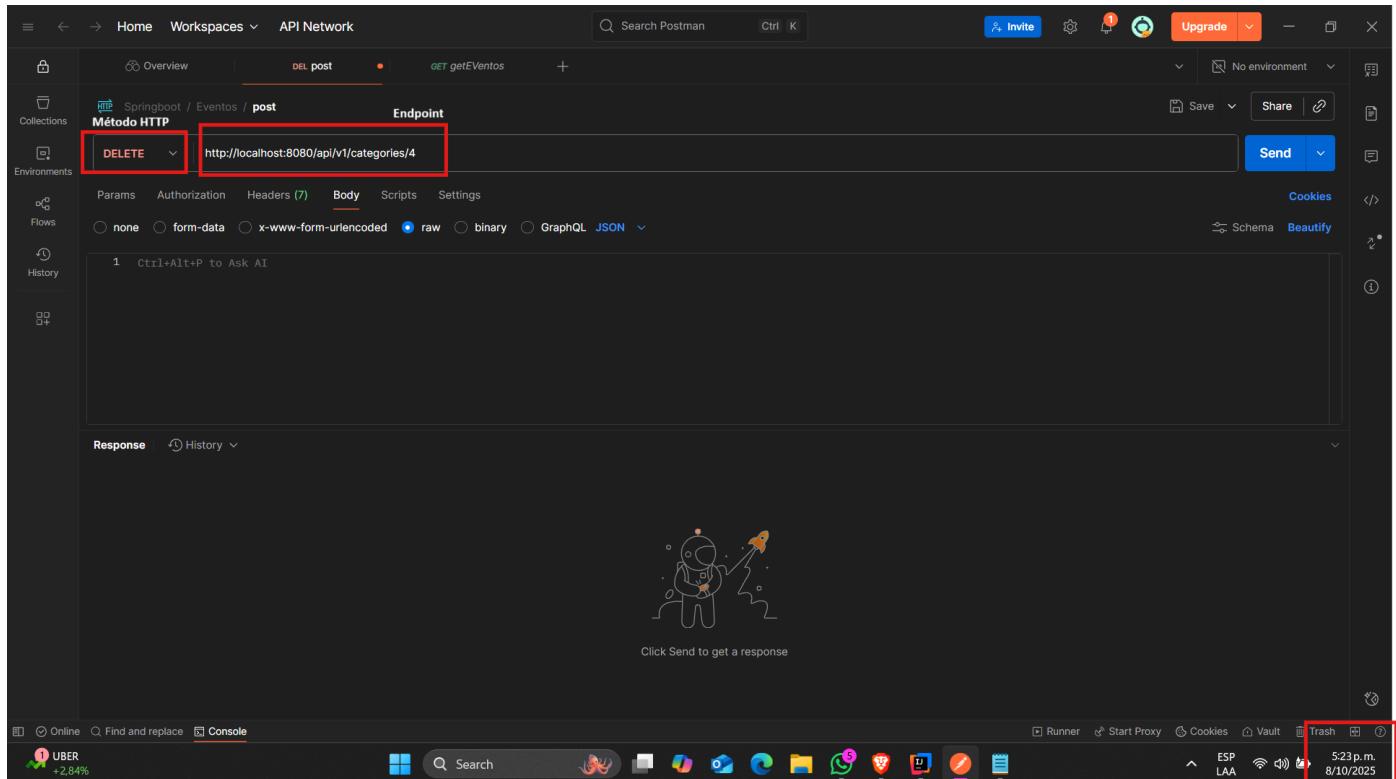
```
{  
  "error": "No se ha encontrado la categoría con id: 999"  
}
```

## Prueba superada? Si.

### Caso de Prueba 9: Error al eliminar una categoría con un token invalido o distinto

- **Endpoint:** DELETE /api/v1/categories/{id}
- **Método HTTP:** DELETE
- **Descripción:** Trata de eliminar una categoria con un token sin autorizacion.
- **Parámetros:**
  - id (path) → ID de la categoría a eliminar.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):**

```
{  
    "error": "No tienes autorización para acceder a este recurso"  
}
```
- **Código de Respuesta Esperado:**
  - 401 Unauthorized si el token no es valido.
- **Prueba exitosa si:** La API devuelve 401 Content al intentar eliminar la categoría indicada con el token invalido.
- **Pasos de ejecución de la prueba**
  - Q) Se envia el path en el endpoint con el id de la categoria deseada a eliminar



The screenshot shows the Postman application interface. In the main workspace, there is a collection named "API Network". A specific request is selected, showing a "DELETE" method and the URL "http://localhost:8080/api/v1/categories/4". The "Body" tab is selected, showing an empty JSON object. The "Headers" tab contains a single header: "Authorization: Bearer <token>". The "Params" tab is empty. The "Settings" tab is also empty. At the bottom of the request card, there is a note: "Click Send to get a response". The status bar at the bottom of the screen shows the date and time as "8/10/2025 5:23 p.m." and the battery level as "+2,84%".

R) Enviar peticion y obtener el codigo 401

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8080/api/v1/categories/4`. The response body contains the JSON object: `{ "error": "No tienes autorización para acceder a este recurso" }`. The status code is 401 Unauthorized.

### Respuesta obtenida:

```
{
    "error": "No tienes autorización para acceder a este recurso"
}
```

**Prueba superada? Si.**

### Caso de Prueba 10: Actualizar una categoría

- **Endpoint:** PUT /api/v1/categories/{id}
- **Método HTTP:** PUT
- **Descripción:** Actualiza los datos de una categoría existente.
- **Parámetros:**
  - id (path) → ID de la categoría a actualizar.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):**

```
{
    "name": "Hardware y periféricos",
    "description": "Problemas relacionados con dispositivos físicos"
}
```

- **Código de Respuesta Esperado:**

- 200 OK si la actualización fue exitosa.

- **Cuerpo de la Respuesta Esperada:**

{

```
"id": 3,
```

```
"name": "Hardware y periféricos",
```

```
"description": "Problemas relacionados con dispositivos físicos"
```

}

- **Prueba exitosa si:** La API devuelve 200 OK con la categoría actualizada.

- **Pasos de ejecución de la prueba**

- S) Se envia el path en el endpoint con el id de la categoria deseada a actualizar junto con el cuerpo de la solicitud

The screenshot shows the Postman interface with a PUT request to the endpoint `http://localhost:8080/api/v1/categories/3`. The request body is a JSON object containing the updated category details: `{"name": "Hardware y periféricos", "description": "Problemas relacionados con dispositivos físicos"}`. The response at the bottom indicates a successful 200 OK status.

- T) Enviar petición y obtener el código 200

The screenshot shows the Postman interface with the following details:

- Method:** PUT
- URL:** http://localhost:8080/api/v1/categories/3
- Body (JSON):**

```

1 {
2   "name": "Hardware y periféricos",
3   "description": "Problemas relacionados con dispositivos físicos"
4 }
5

```
- Response Status:** 200 OK
- Response Body (JSON):**

```

1 {
2   "id": 3,
3   "name": "Hardware y periféricos",
4   "description": "Problemas relacionados con dispositivos físicos"
5 }

```

### Respuesta obtenida:

```
{
  "id": 3,
  "name": "Hardware y periféricos",
  "description": "Problemas relacionados con dispositivos físicos"
}
```

Prueba superada? Si.

### Caso de Prueba 11: Error al actualizar una categoría no existente

- **Endpoint:** PUT /api/v1/categories/{id}
- **Método HTTP:** PUT
- **Descripción:** Actualiza los datos de una categoría existente.
- **Parámetros:**
  - id (path) → ID de la categoría a actualizar.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):**

{

```
"name": "Hardware y periféricos",
"description": "Problemas relacionados con dispositivos físicos"
}
```

- **Código de Respuesta Esperado:**

- 404 NOT FOUND si no se encuentra la categoria con el id suministrado.

- **Cuerpo de la Respuesta Esperada:**

```
{
  "error": "No se ha encontrado la categoría con id: 999"
}
```

- **Prueba exitosa si:** La API devuelve 404 NOT FOUND con el mensaje de error.

- **Pasos de ejecución de la prueba**

- U) Se envia el path en el endpoint con el id de la categoria deseada a actualizar junto con el cuerpo de la solicitud

The screenshot shows the Postman interface with a PUT request configuration. The method is selected as 'PUT' and the endpoint is 'http://localhost:8080/api/v1/categories/999'. The 'Body' tab is active, showing a raw JSON payload:

```
{
  "name": "Hardware y periféricos",
  "description": "Problemas relacionados con dispositivos físicos"
}
```

The 'Cuerpo de la solicitud' (Request Body) label is visible next to the JSON code. The status bar at the bottom indicates a successful '200 OK' response.

V) Enviar petición y obtener el código 404

The screenshot shows the Postman interface with a failed API call. The request is a PUT to `http://localhost:8080/api/v1/categories/999`. The body contains JSON data: `{ "name": "Hardware y periféricos", "description": "Problemas relacionados con dispositivos físicos" }`. The response code is 404 Not Found, and the response body is `{"error": "No se ha encontrado la categoría con id: 999"}`.

### Respuesta obtenida:

```
{  
  "error": "No se ha encontrado la categoría con id: 999"  
}
```

Prueba superada? Si.

#### - 6.4. Comment

##### Caso de Prueba 1: Obtener todos los comentarios de un ticket

Endpoint: GET /comments/ticket/{ticketId}

Método HTTP: GET

Descripción: Obtener la lista de comentarios asociados a un ticket específico.

Parámetros:

- ticketId (Path): ID del ticket cuyos comentarios se desean obtener.

Headers:

- Authorization: Bearer <token>
- Content-Type: application/json

Cuerpo de la Solicitud (Body): No aplica.

## Códigos de Respuesta Esperados: 200 OK

### Cuerpo de la Respuesta Esperada (Muestra)

```
{  
    "id": 1,  
  
    "ticket": {  
        "id": 1,  
  
        "employeeId": null,  
  
        "category": {  
            "id": 4,  
  
            "name": "Red",  
  
            "description": "Problemas relacionados con conectividad"  
        },  
  
        "title": "Solicitud de nueva funcionalidad",  
  
        "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
  
        "priority": "BAJA",  
  
        "state": "ABIERTO",  
  
        "creationDate": "2025-10-19T14:50:59.014026",  
  
        "closingDate": null  
    },  
  
    "employee": {  
        "id": 2,  
  
        "name": "Carlos Pérez",  
  
        "email": "carlos.perez@empresa.com",  
  
        "role": "USER",  
  
        "department": "Soporte Técnico"  
    },  
  
    "text": "Prueba de comentario",  
  
    "creationDate": "2025-10-19T14:52:44.243587"  
},  
{  
    "id": 2,
```

```
"ticket": {  
    "id": 1,  
    "employeeId": null,  
    "category": {  
        "id": 4,  
        "name": "Red",  
        "description": "Problemas relacionados con conectividad"  
    },  
    "title": "Solicitud de nueva funcionalidad",  
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
    "priority": "BAJA",  
    "state": "ABIERTO",  
    "creationDate": "2025-10-19T14:50:59.014026",  
    "closingDate": null  
},  
    "employee": {  
        "id": 2,  
        "name": "Carlos Pérez",  
        "email": "carlos.perez@empresa.com",  
        "role": "USER",  
        "department": "Soporte Técnico"  
    },  
    "text": "Prueba de comentario",  
    "creationDate": "2025-10-19T14:56:16.465813"  
}
```

### Prueba exitosa si:

- La API devuelve **200 OK** con una lista de comentarios pertenecientes al ticketId.
- Se envia la solicitud con el cuerpo de la solicitud

The screenshot shows the Postman interface with a failed API call. The URL in the endpoint field is highlighted with a red box. The status bar at the bottom right shows the date and time as 3:21 p.m. 19/10/2025.

Body Cookies Headers (13) Test Results

400 Bad Request 72 ms 450 B Save Response

3:21 p.m. 19/10/2025

- Enviar petición y obtener el código 200

The screenshot shows the Postman interface with a successful API call. The URL in the endpoint field is highlighted with a red box. The status bar at the bottom right shows the date and time as 3:24 p.m. 9/10/2025.

Body Cookies Headers (14) Test Results

200 OK 195 ms 2.6 KB Save Response

Cuerpo de la respuesta

```
[{"id": 1, "ticket": {"id": 1, "employeeId": null, "category": {"id": 4, "name": "Red", "description": "Problemas relacionados con conectividad"}, "title": "Solicitud de nueva funcionalidad", "description": "Se solicita agregar un bot\u00f3n de exportar datos en el reporte mensual", "priority": "BAJA", "state": "ABIERTO", "creationDate": "2025-10-19T14:50:59.014026", "closingDate": null}, "employee": {"id": 2, "name": "Carlos P\u00f3rez", "email": "carlos.perez@empresa.com", "role": "USER", "department": "Soporte T\u00e9cnico"}, "text": "Prueba de comentario", "creationDate": "2025-10-19T14:52:44.243587"}]
```

25°C Mayorn, nublado

3:24 p.m. 9/10/2025

### Respuesta obtenida (muestra):

```
{  
  "id": 1,  
  "ticket": {  
    "id": 1,
```

```
"employeeId": null,  
"category": {  
    "id": 4,  
    "name": "Red",  
    "description": "Problemas relacionados con conectividad"  
},  
"title": "Solicitud de nueva funcionalidad",  
"description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
"priority": "BAJA",  
"state": "ABIERTO",  
"creationDate": "2025-10-19T14:50:59.014026",  
"closingDate": null  
},  
"employee": {  
    "id": 2,  
    "name": "Carlos Pérez",  
    "email": "carlos.perez@empresa.com",  
    "role": "USER",  
    "department": "Soporte Técnico"  
},  
"text": "Prueba de comentario",  
"creationDate": "2025-10-19T14:52:44.243587"  
},  
{  
    "id": 2,  
    "ticket": {  
        "id": 1,  
        "employeeId": null,  
        "category": {  
            "id": 4,  
            "name": "Red",  
            "description": "Problemas relacionados con conectividad"  
},  
        "title": "Solicitud de nueva funcionalidad",  
        "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
        "priority": "BAJA",  
        "state": "ABIERTO",  
        "creationDate": "2025-10-19T14:50:59.014026",  
        "closingDate": null  
    },  
    "comment": {  
        "text": "Prueba de comentario",  
        "creationDate": "2025-10-19T14:52:44.243587"  
    }  
}
```

```
        "description": "Problemas relacionados con conectividad"  
    },  
    "title": "Solicitud de nueva funcionalidad",  
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
    "priority": "BAJA",  
    "state": "ABIERTO",  
    "creationDate": "2025-10-19T14:50:59.014026",  
    "closingDate": null  
},  
"employee": {  
    "id": 2,  
    "name": "Carlos Pérez",  
    "email": "carlos.perez@empresa.com",  
    "role": "USER",  
    "department": "Soporte Técnico"  
},  
"text": "Prueba de comentario",  
"creationDate": "2025-10-19T14:56:16.465813"  
}  
• Prueba superada? Si.
```

## Caso de Prueba 2: Error al obtener comentarios de un ticket inexistente

- **Endpoint:** GET /comments/ticket/{ticketId}
- **Método HTTP:** GET
- **Descripción:** Error al obtener comentarios de un ticket inexistente
- **Parámetros:**
  - id (path): ID del ticket.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json

- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 404 NOT FOUND
- **Cuerpo de la Respuesta Esperada:**

```
{
  "error": "Tiquete no encontrado con ID: 999"
}
```

- **Prueba exitosa si:** La API devuelve 200 OK y retorna la información del comentario solicitado.
- **Pasos de ejecución de la prueba**
  - W) Se envia la solicitud con el cuerpo de la solicitud

The screenshot shows the Postman application interface. In the center, there's a request configuration window for an endpoint named 'getEventos'. The method is set to 'GET' and the URL is 'http://localhost:8080/api/v1/comments/ticket/999'. The 'Body' tab is selected, showing the JSON response from the previous step: {"error": "Tiquete no encontrado con ID: 999"}. Below the request window, the status bar displays '200 OK', '195 ms', '2.6 KB', and the date '19/10/2025'. At the bottom right of the status bar, the time is shown as '3:29 p.m.'.

X) Enviar petición y obtener el código 404

The screenshot shows the Postman interface with a failed API call. The endpoint is set to 'GET' and the URL is 'http://localhost:8080/api/v1/comments/ticket/999'. The response status is '404 Not Found'. The body of the response is a JSON object with a single key 'error': "Tiquete no encontrado con ID: 999".

### Respuesta obtenida:

```
{  
  "error": "Tiquete no encontrado con ID: 999"  
}
```

- Prueba superada? Si.

### Caso de Prueba 3: Crear un nuevo comentario

- **Endpoint:** POST /comments
- **Método HTTP:** POST
- **Descripción:** Registrar un nuevo comentario en el sistema.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):**

```
{  
  "ticketId": 1,  
  "employeeId": 2,  
  "text": "Prueba de comentario"
```

```
}



- Código de Respuesta Esperado: 201 Created
- Cuerpo de la Respuesta Esperada:



{

    "id": 4,

    "ticket": {

        "id": 1,
        "employeed": null,
        "category": {
            "id": 4,
            "name": "Red",
            "description": "Problemas relacionados con conectividad"
        },
        "title": "Solicitud de nueva funcionalidad",
        "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
        "priority": "BAJA",
        "state": "ABIERTO",
        "creationDate": "2025-10-19T14:50:59.014026",
        "closingDate": null
    },
    "employee": {

        "id": 2,
        "name": "Carlos Pérez",
        "email": "carlos.perez@empresa.com",
        "role": "USER",
        "department": "Soporte Técnico"
    },
    "text": "Prueba de comentario",
    "creationDate": "2025-10-19T14:58:40.922284"
}
```

- **Prueba exitosa si:** La API devuelve 200 Created y retorna el comentario creado con su ID asignado.
- **Pasos de ejecución de la prueba**
  - Y) Se envia la solicitud con el cuerpo de la solicitud

The screenshot shows the Postman interface with a successful POST request to the endpoint `http://localhost:8080/api/v1/comments`. The request method is set to `POST`, and the URL is specified in the `Endpoint` field. The `Body` tab is selected, showing a JSON payload with three fields: `ticketId`, `employeeId`, and `text`. The response status is `200 OK` with a response time of `719 ms` and a size of `982 B`.

```
{
  "ticketId": 1,
  "employeeId": 2,
  "text": "Prueba de comentario"
}
```

Z) Enviar petición y obtener el código 200

The screenshot shows the Postman interface after sending the POST request. The response status is `200 OK` with a response time of `719 ms` and a size of `982 B`. The response body is a detailed JSON object representing the created comment, including fields like `id`, `ticket`, `employee`, and `text`.

```
{
  "id": 4,
  "ticket": {
    "id": 1,
    "employeeId": null,
    "category": {
      "id": 4,
      "name": "Red",
      "description": "Problemas relacionados con conectividad"
    },
    "title": "Solicitud de nueva funcionalidad",
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
    "priority": "BAJA",
    "state": "ABIERTO",
    "creationDate": "2025-10-19T14:59:59.014026",
    "closingDate": null
  },
  "employee": {
    "id": 2,
    "name": "Carlos Pérez",
    "email": "carlos.perez@empresa.com",
    "role": "USER",
    "department": "Soporte Técnico"
  },
  "text": "Prueba de comentario",
  "creationDate": "2025-10-19T14:58:40.922284"
}
```

**Respuesta obtenida:**

```
{
```

```
  "id": 4,
```

```

"ticket": {
    "id": 1,
    "employeeId": null,
    "category": {
        "id": 4,
        "name": "Red",
        "description": "Problemas relacionados con conectividad"
    },
    "title": "Solicitud de nueva funcionalidad",
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",
    "priority": "BAJA",
    "state": "ABIERTO",
    "creationDate": "2025-10-19T14:50:59.014026",
    "closingDate": null
},
"employee": {
    "id": 2,
    "name": "Carlos Pérez",
    "email": "carlos.perez@empresa.com",
    "role": "USER",
    "department": "Soporte Técnico"
},
"text": "Prueba de comentario",
"creationDate": "2025-10-19T14:58:40.922284"
}

```

- **Prueba superada? Si.**

#### **Caso de Prueba 5: Error al crear comentario sin texto**

- **Endpoint:** POST /comments
- **Método HTTP:** POST
- **Descripción:** Intentar registrar un comentario sin el campo obligatorio text.

- **Parámetros:** Ninguno.

- **Headers:**

- Authorization: Bearer <token>
- Content-Type: application/json

- **Cuerpo de la Solicitud (Body):**

```
{
```

```
  "ticketId": 1,
```

```
  "employeeId": 2
```

```
}
```

- **Código de Respuesta Esperado:** 400 Bad Request

- **Cuerpo de la Respuesta Esperada:**

```
{
```

```
  "error": "Comment text is required"
```

```
}
```

- **Prueba exitosa si:** La API devuelve 400 Bad Request y explica que el campo text es obligatorio.

- **Pasos de ejecución de la prueba**

AA) Se envia la solicitud con el cuerpo de la solicitud

The screenshot shows the Postman application interface. A POST request is being prepared to the endpoint `http://localhost:8080/api/v1/comments`. The request method is set to `POST`, and the URL is specified in the `Endpoint` field. The `Body` tab is selected, showing a JSON payload:

```
{
  "ticketId": 1,
  "employeeId": 2
}
```

The response status bar at the bottom indicates a `400 Bad Request` status with a timestamp of `3:18 p.m. 19/10/2025`.

BB) Enviar petición y obtener el código 400

The screenshot shows the Postman interface with a POST request to `http://localhost:8080/api/v1/comments`. The request body is a JSON object with `ticketId: 1` and `employeeId: 2`. The response code is **400 Bad Request**, and the response body is `{"text": "El texto no puede ser nulo o vacío."}`. The timestamp in the bottom right corner is 3:09 p.m. on 19/10/2025.

### Respuesta obtenida:

```
{  
  "text": "El texto no puede ser nulo o vacío."  
}
```

- **Prueba superada? Si.**

### Caso de Prueba 6: Eliminar un comentario

- **Endpoint:** `DELETE /comments/{id}`
- **Método HTTP:** `DELETE`
- **Descripción:** Eliminar un comentario existente del sistema.
- **Parámetros:**
  - `id` (path): ID del comentario a eliminar.
- **Headers:**
  - `Authorization: Bearer <token>`
  - `Content-Type: application/json`
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** `204 No Content`

- **Cuerpo de la Respuesta Esperada:** No aplica.
- **Prueba exitosa si:** La API devuelve 204 No Content y el comentario desaparece del sistema.
- Se envia la solicitud con path de la solicitud

The screenshot shows the Postman interface with a DELETE request to `http://localhost:8080/api/v1/comments/1`. The method dropdown is set to `DELETE` and the URL field is highlighted with a red box. The response status is `204 No Content`, indicating that the comment was successfully deleted. The timestamp in the bottom right corner is 3:40 p.m. on 19/10/2025.

- Enviar peticion y obtener el codigo 204

The screenshot shows the Postman interface with the same DELETE request. The response status is explicitly labeled as `204 No Content` in the status bar. The raw response body is shown as an empty string. The timestamp in the bottom right corner is 3:39 p.m. on 19/10/2025.

### Respuesta obtenida:

{

"error": "No se encontro el comentario con el id: 999"

}

- **Prueba superada? Si.**

#### Caso de Prueba 7: Error al eliminar comentario inexistente

- **Endpoint:** DELETE /comments/{id}
- **Método HTTP:** DELETE
- **Descripción:** Intentar eliminar un comentario que no existe en el sistema.
- **Parámetros:**
  - id (path): ID inexistente.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 404 Not Found
- **Cuerpo de la Respuesta Esperada:**

{

    "error": "Comment not found"

}

- **Prueba exitosa si:** La API devuelve 404 Not Found y explica que el comentario no existe.
- **Pasos de ejecución de la prueba**
  - CC) Se envia la solicitud con el path de la solicitud

Método HTTP: DELETE

Endpoint: http://localhost:8080/api/v1/comments/999

Body: raw

Test Results: 404 Not Found, 92 ms, 485 B, 19/10/2025

DD) Enviar petición y obtener el código 404

Método HTTP: DELETE

Endpoint: http://localhost:8080/api/v1/comments/999

Body: raw

Código HTTP: 404 Not Found

Cuerpo de la respuesta:

```
{  
  "error": "No se encontro el comentario con el id: 999"  
}
```

Test Results: 404 Not Found, 24 ms, 485 B, 19/10/2025

Respuesta obtenida:

```
{  
  "error": "No se encontro el comentario con el id: 999"  
}
```

- Prueba superada? Si.

---

- **6.5. Assignment**

#### Caso de Prueba 1: Crear una nueva asignación

- **Endpoint:** POST /api/v1/assignments
- **Método HTTP:** POST
- **Descripción:** Crear una nueva asignación de un ticket a un empleado.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):**

```
{  
  "ticketId": 1,  
  "employeeId": 3  
}
```

- **Código de Respuesta Esperado:** 201 Created
- **Cuerpo de la Respuesta Esperada:**

```
{  
  "id": 1,  
  "ticket": {  
    "id": 1,  
    "title": "Error en login"  
  },  
  "employee": {  
    "id": 3,  
    "name": "John Doe"  
  },  
  "assignmentDate": "2025-09-14T12:00:00"  
}
```

- **Prueba exitosa si:** La API devuelve 201 Created y los datos coinciden con la asignación creada.
- **Ejecución de prueba y resultado obtenido:**
  - Cuerpo de petición para asignar ticket a agente

The screenshot shows the Postman interface with the following details:

- Método HTTP:** POST
- Endpoint:** {{baseUrl}}/api/v1/assignments
- Body (raw JSON):**

```
{
  "ticketId": 1,
  "employeeId": 3
}
```

This JSON object is highlighted with a red box and labeled "Cuerpo de la petición".
- Send button:** A blue button at the top right of the request panel.
- Response panel:** Shows a placeholder message: "Click Send to get a response".
- Time stamp:** 3:07 p.m. 23/09/2025 is shown in the bottom right corner.

- Resultado obtenido al enviar la petición.

The screenshot shows the Postman interface with the following details:

- Método HTTP:** POST
- Endpoint:** {{baseUrl}}/api/v1/assignments
- Body (raw JSON):**

```
{
  "id": 2,
  "ticket": {
    "id": 4,
    "employeeId": 2,
    "category": {
      "id": 1,
      "name": "Software",
      "description": "Errores, bugs o problemas de instalación y uso de aplicaciones."
    },
    "title": "Aplicación de contabilidad se cierra inesperadamente",
    "description": "Al abrir el módulo de facturación en la aplicación de contabilidad, esta se cierra de forma automática sin mostrar mensaje de error. El problema ocurre desde la última actualización.",
    "priority": "ALTA",
    "state": "EN_PROGRESO",
    "creationDate": "2025-09-28T14:47:41.764982",
    "closingDate": null
  },
  "employee": {
    "id": 3,
    "name": "Alejandro Magno",
    "email": "alejandromagno@gmail.com",
    "role": "AGENT",
    "department": "Soporte Contabilidad"
  },
  "assignmentDate": "2025-09-28T15:27:44.006285"
}
```

This JSON object is highlighted with a red box.
- Status code:** 201 Created is highlighted with a red box in the top right corner of the response panel.
- Time stamp:** 3:28 p.m. 23/09/2025 is shown in the bottom right corner.

- **¿Prueba Superada?:** Si, Aprobada.
- **Observaciones:** En la respuesta de la creación de la asignación de postman tiene los datos más completos en cuanto al ticket y el agente

## Caso de Prueba 2: Error al crear asignación con datos nulos

- **Endpoint:** POST /api/v1/assignments
- **Método HTTP:** POST
- **Descripción:** Intentar crear una asignación sin ticketId o employeeld.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):**

```
{  
  "ticketId": null,  
  "employeeld": 3  
}
```

- **Código de Respuesta Esperado:** 400 Bad Request
- **Cuerpo de la Respuesta Esperada:**

```
{  
  "error": "El id del ticket no puede ser nulo."  
}
```

- **Prueba exitosa si:** La API devuelve 400 Bad Request indicando el campo faltante o inválido.
- **Ejecución de prueba y resultado obtenido:**

The screenshot shows the Postman interface with a failed API call. The request method is POST to the endpoint `(baseUrl) /api/v1/assignments`. The request body is JSON with the key `employeeId` set to 3. The response code is 400 Bad Request, and the response body is a JSON object with the message `"ticketId": "El id del ticket no puede ser nulo."`. The timestamp at the bottom right is 3:40 p.m. on 23/09/2025.

- **¿Prueba Superada?:** Si, Aprobada
- **Observaciones:** Se obtuvo correctamente el código 400 por enviar sin un id de un ticket

### Caso de Prueba 3: Listar asignaciones por empleado

- **Endpoint:** GET /api/v1/assignments/employee/{employeeId}
- **Método HTTP:** GET
- **Descripción:** Obtener todas las asignaciones de un empleado agente.
- **Parámetros:**
  - `employeeId` (path): ID del empleado.
- **Headers:**
  - `Authorization: Bearer <token>`
  - `Content-Type: application/json`
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 200 OK
- **Cuerpo de la Respuesta Esperada:**

[

{

`"id": 11,`

`"ticket": {`

```

    "id": 124,
    "title": "Problema con facturación"
},
"employee": {
    "id": 7,
    "name": "John Doe"
},
"assignmentDate": "2025-09-14T13:00:00"
}
]

```

- **Prueba exitosa si:** La API devuelve 200 OK con todas las asignaciones del empleado.
- **Ejecución de prueba y resultado obtenido:**
  - Se pone en el path del endpoint el id de un agente existente

HTTP Request Details:

- Método HTTP: GET
- Endpoint: {{baseURL}}/api/v1/assignments/employee/3
- Body: JSON

HTTP Response Details:

- Código HTTP: 200 OK
- Respuesta (Lista de asignaciones):

```

[{"id": 2, "ticket": {"id": 1, "employeeId": 2, "category": {"id": 1, "name": "Software", "description": "Errores, bugs o problemas de instalación y uso de aplicaciones."}, "title": "Aplicación de contabilidad se cierra inesperadamente", "description": "Al abrir el módulo de facturación en la aplicación de contabilidad, esta se cierra de forma automática sin mostrar mensaje de error. El problema ocurre desde la última actualización.", "priority": "ALTA", "state": "EN_PROGRESO", "creationDate": "2025-09-23T14:47:41.764932", "closingDate": null}, "employee": {"id": 3, "name": "Alejandro Magno", "email": "alejandromagno@gmail.com", "role": "AGENT", "department": "Soporte Contabilidad"}]

```

- **Muestra de datos obtenidos de la respuesta:**

[{"id": 2, "ticket": {"id": 1, "employeeId": 2, "category": {"id": 1, "name": "Software", "description": "Errores, bugs o problemas de instalación y uso de aplicaciones."}, "title": "Aplicación de contabilidad se cierra inesperadamente", "description": "Al abrir el módulo de facturación en la aplicación de contabilidad, esta se cierra de forma automática sin mostrar mensaje de error. El problema ocurre desde la última actualización.", "priority": "ALTA", "state": "EN\_PROGRESO", "creationDate": "2025-09-23T14:47:41.764932", "closingDate": null}, "employee": {"id": 3, "name": "Alejandro Magno", "email": "alejandromagno@gmail.com", "role": "AGENT", "department": "Soporte Contabilidad"}, "assignmentDate": "2025-09-23T15:27:44.006235"},

```
{"id":3,"ticket":{"id":2,"employeeId":2,"category":{"id":1,"name":"Software","description":"Errores, bugs o problemas de instalación y uso de aplicaciones."},"title":"Error al generar reportes financieros","description":"Cuando se intenta generar el reporte mensual de ingresos y egresos, el sistema muestra una pantalla en blanco y no produce el archivo PDF esperado. Este error empezó después de aplicar la última actualización.","priority":"MEDIA","state":"EN_PROGRESO","creationDate":"2025-09-23T15:50:36.432072","closingDate":null,"employee":{"id":3,"name":"Alejandro Magno","email":"alejandromagno@gmail.com","role":"AGENT","department":"Soporte Contabilidad"},"assignmentDate":"2025-09-23T15:52:03.858704"}]
```

- **¿Prueba Superada?:** Si, Aprobada
  - **Observaciones:** Se describe totalmente toda la información de ticket y empleado en cada asignación
- 

#### Caso de Prueba 4: Error al listar asignaciones de empleado inexistente

- **Endpoint:** GET /api/v1/assignments/employee/{employeeId}
- **Método HTTP:** GET
- **Descripción:** Intentar obtener asignaciones de un empleado que no existe.
- **Parámetros:**
  - employeeId (path): ID inexistente.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 404 Not Found
- **Cuerpo de la Respuesta Esperada:**

```
{  
    "error": "Agente no encontrado"  
}
```

- **Prueba exitosa si:** La API devuelve 404 Not Found con mensaje adecuado.
- **Ejecución de prueba y resultado obtenido:**
  - a) Se pone en el path del endpoint el id de un empleado inexistente

The screenshot shows the Postman interface with the following details:

- Endpoint:** {{baseUrl}} /api/v1/assignments/employee/4
- Método HTTP:** GET
- Código HTTP:** 404 Not Found
- Body (Response):**

```
{
  "message": "Empleado no encontrado."
}
```

- **¿Prueba Superada?:** Si, Aprobada
- **Observaciones:** Se envió código 404 de no encontrado el empleado correctamente

### Caso de Prueba 5: Obtener asignación por ticket

- **Endpoint:** GET /api/v1/assignments/ticket/{ticketId}
- **Método HTTP:** GET
- **Descripción:** Obtener la asignación actual de un ticket específico.
- **Parámetros:**
  - ticketId (path): ID del ticket.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 200 OK
- **Cuerpo de la Respuesta Esperada:**

{

"id": 12,

"ticket": {

"id": 125,

```

    "title": "Fallo en conexión"
},
"employee": {
    "id": 8,
    "name": "Jane Smith"
},
"assignmentDate": "2025-09-14T14:30:00"
}

```

- **Prueba exitosa si:** La API devuelve 200 OK y retorna la asignación del ticket.
- **Ejecución de prueba y resultado obtenido:**
  - Se pone en el path del endpoint el id de un ticket existente

The screenshot shows the Postman interface with the following details:

- Método HTTP:** GET
- Endpoint:** `(baseUrl) /api/v1/assignments/ticket/1`
- Código Http:** 200 OK
- Body (JSON):**

```

1  {
2      "id": 2,
3      "ticket": {
4          "id": 1,
5          "employeeId": 2,
6          "category": {
7              "id": 1,
8              "name": "Software",
9              "description": "Errores, bugs o problemas de instalación y uso de aplicaciones."
10         },
11         "title": "Aplicación de contabilidad se cierra inesperadamente",
12         "description": "Al abrir el módulo de facturación en la aplicación de contabilidad, esta se cierra de forma automática sin mostrar mensaje de error. El problema ocurre desde la última actualización.",
13         "priority": "ALTA",
14         "state": "EN PROGRESO",
15         "creationDate": "2025-09-23T14:47:41.764982",
16         "closingDate": null
17     },
18     "employee": {
19         "id": 3,
20         "name": "Alejandro Magno",
21         "email": "alejandromagno@gmail.com",
22         "role": "AGENT",
23         "department": "Soporte Contabilidad"
24     },
25     "assignmentDate": "2025-09-23T15:27:44.006256"
26 }

```

- **¿Prueba Superada?:** Si, aprobado
- **Observaciones:** Se obtuvo la asignación del ticket especificado con la información completa de su agente y ticket

## Caso de Prueba 6: Error al obtener asignación de ticket inexistente

- **Endpoint:** GET /api/v1/assignments/ticket/{ticketId}
- **Método HTTP:** GET
- **Descripción:** Intentar obtener la asignación de un ticket que no existe.
- **Parámetros:**
  - ticketId (path): ID inexistente.

- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 404 Not Found
- **Cuerpo de la Respuesta Esperada:**

```
{
  "error": "Ticket o asignación no encontrada"
}
```

- **Prueba exitosa si:** La API devuelve 404 Not Found con mensaje adecuado.
- **Ejecución de prueba y resultado obtenido:**
  - Se pone en el path del endpoint el id de un ticket inexistente

The screenshot shows the Postman interface with a collection named 'My Workspace' containing various API endpoints. The current request is for 'Obtener assignment por ticket' (Assignment / Obtener assignment por ticket) using the GET method. The URL is set to `{(baseUrl)}/api/v1/assignments/ticket/3`. The 'Headers' tab shows '(8 hidden)' and the 'Body' tab is set to 'JSON'. The 'Código HTTP' section shows '404 Not Found' with a response time of '80 ms' and a size of '406 B'. The 'Respuesta Obtenida' section displays the JSON response: `1 { 2 | "message": "El ticket con id 3 no tiene una asignación activa." 3 }`. The 'Send' button is highlighted with a red box.

- **¿Prueba Superada?:** Si, aprobado.
- **Observaciones:** Se envió código 404 del ticket no encontrado correctamente o sin asignación activa

## Caso de Prueba 7: Reasignar agente

- **Endpoint:** PATCH /api/v1/assignments/reassign/{id}?employeeId={nuevolid}
- **Método HTTP:** PATCH

- **Descripción:** Cambiar el empleado asignado a un ticket existente.

- **Parámetros:**

- id (path): ID de la asignación.
- employeed (query): ID del nuevo empleado.

- **Headers:**

- Authorization: Bearer <token>
- Content-Type: application/json

- **Cuerpo de la Solicitud (Body):** No aplica.

- **Código de Respuesta Esperado:** 200 OK

- **Cuerpo de la Respuesta Esperada:**

```
{  
  "id": 12,  
  "ticket": {  
    "id": 125,  
    "title": "Fallo en conexión"  
  },  
  "employee": {  
    "id": 9,  
    "name": "Carlos Pérez"  
  },  
  "assignmentDate": "2025-09-14T15:00:00"  
}
```

- **Prueba exitosa si:** La API devuelve 200 OK y la asignación refleja el nuevo empleado.

- **Ejecución de prueba y resultado obtenido:**

- a) Se llena el path con id de ticket y parámetro de consulta para el id del nuevo agente a asignar

The screenshot shows the Postman interface with the following details:

- Método HTTP:** PATCH
- Endpoint:** {{baseUrl}} /api/v1/assignments/reassign/2?employeeId=4
- Params:** employeeId (Value: 4)
- Send button:** The "Send" button is highlighted with a red box.
- Response area:** Placeholder text "Click Send to get a response" and a small rocket illustration.
- Bottom bar:** Includes icons for Postbot, Runner, Capture requests, Desktop Agent, Cookies, Vault, and Trash, along with the date and time (5:54 p.m., 23/09/2025).

- b) Una vez se envia la petición devuelve la nueva asignación del ticket con el nuevo agente

The screenshot shows the Postman interface with the following details:

- Método HTTP:** PATCH
- Endpoint:** {{baseUrl}} /api/v1/assignments/reassign/2?employeeId=4
- Body:** JSON response containing ticket assignment details (category, title, description, employee information, etc.).
- Status code:** 200 OK (highlighted with a red box)
- Resultado obtenido:** The JSON response content is displayed in the main pane.
- Bottom bar:** Includes icons for Postbot, Runner, Capture requests, Desktop Agent, Cookies, Vault, and Trash, along with the date and time (6:15 p.m., 23/09/2025).

- **¿Prueba Superada?: Si, aprobada**
- **Observaciones:** Devuelve la asignación con el nuevo agente con código 200

## Caso de Prueba 8: Error al reasignar agente con ID inválido

- **Endpoint:** PATCH /api/v1/assignments/reassign/{id}?employeeId={nuevolid}
- **Método HTTP:** PATCH
- **Descripción:** Intentar reasignar a un empleado que no existe.

- **Parámetros:**

- id (path): ID de la asignación.
- employeed (query): ID inexistente.

- **Headers:**

- Authorization: Bearer <token>
- Content-Type: application/json

- **Cuerpo de la Solicitud (Body):** No aplica.

- **Código de Respuesta Esperado:** 404 Not Found

- **Cuerpo de la Respuesta Esperada:**

{

"error": "Asignación o empleado no encontrado"

}

- **Prueba exitosa si:** La API devuelve 404 Not Found con mensaje correcto.

- **Ejecución de prueba y resultado obtenido:**

- Se pone en el query de employeed del endpoint el id de un empleado inexistente o que no sea agente

The screenshot shows the Postman interface with the following details:

- Método HTTP:** PATCH
- Endpoint:** {{baseUrl}} /api/v1/assignments/reassign/2?employeed=5
- Params:** Key: employeed, Value: 5
- Body:** JSON (empty)
- Response Status:** 404 Not Found
- Response Body:** {
 1: {
 2: "message": "Empleado no encontrado."
 3: }
}

- **¿Prueba Superada?:** Si, aprobada

- **Observaciones:** Se devuelve el código 404 al no encontrar el empleado nuevo a reasignar

---

- **6.7. Evidence**

## Caso de Prueba 1: Crear una nueva evidencia

- **Endpoint:** POST /api/evidence
- **Método HTTP:** POST
- **Descripción:** Permite registrar una nueva evidencia asociada a un ticket.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):**

```
{  
  "ticketId": 1,  
  "url": "https://example.com/evidence1.png"  
}
```

- **Código de Respuesta Esperado:** 201 CREATED
- **Cuerpo de la Respuesta Esperada:**

```
{  
  "id": 2,  
  "ticket": {  
    "id": 1,  
    "employeeId": null,  
    "category": {  
      "id": 4,  
      "name": "Red",  
      "description": "Problemas relacionados con conectividad"  
    },  
    "title": "Solicitud de nueva funcionalidad",  
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
    "priority": "BAJA",  
    "state": "ABIERTO",  
    "status": "PENDIENTE"  
  }  
}
```

```

    "creationDate": "2025-10-19T14:50:59.014026",
    "closingDate": null
},
"url": "https://example.com/evidence1.png"
}

```

- **Prueba exitosa si:** La API devuelve 201 y la evidencia creada coincide con los datos enviados.
- **Pasos de ejecución de la prueba**

EE) Se envia la solicitud con el body

The screenshot shows the Postman interface with a POST request to the endpoint `http://localhost:8080/api/v1/evidences`. The request body is set to raw JSON, containing the following data:

```

1: {
2:   "ticketId": 1,
3:   "url": "https://example.com/evidence1.png"
4: }

```

The response status is 201 Created, with a timestamp of 4:12 p.m. on 19/10/2025.

FF) Enviar peticion y obtener el codigo 201

The screenshot shows the Postman interface with a successful POST request. The URL is `http://localhost:8080/api/v1/evidences`. The response body is displayed in JSON format, and the status code `201 Created` is highlighted with a red box.

```
{  
  "id": 2,  
  "ticket": {  
    "id": 1,  
    "employeeId": null,  
    "category": {  
      "id": 4,  
      "name": "Red",  
      "description": "Problemas relacionados con conectividad"  
    },  
    "title": "Solicitud de nueva funcionalidad",  
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
    "priority": "BAJA",  
    "state": "ABIERTO",  
    "creationDate": "2025-10-19T14:50:59.014026",  
    "closingDate": null  
  },  
  "url": "https://example.com/evidence1.png"  
}
```

## Respuesta obtenida:

```
{  
  "id": 2,  
  "ticket": {  
    "id": 1,  
    "employeeId": null,  
    "category": {  
      "id": 4,  
      "name": "Red",  
      "description": "Problemas relacionados con conectividad"  
    },  
    "title": "Solicitud de nueva funcionalidad",  
    "description": "Se solicita agregar un botón de exportar datos en el reporte mensual",  
    "priority": "BAJA",  
    "state": "ABIERTO",  
    "creationDate": "2025-10-19T14:50:59.014026",  
    "closingDate": null  
  },  
  "url": "https://example.com/evidence1.png"
```

```
}
```

- **Prueba superada? Si.**

#### Caso de Prueba 2: Error al crear evidencia sin ticketId

- **Endpoint:** POST /api/evidence
- **Método HTTP:** POST
- **Descripción:** Intentar crear evidencia sin proporcionar el ticketId.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):**

```
{
  "url": "https://example.com/evidence1.png"
}
```

- **Código de Respuesta Esperado:** 400 Bad Request
- **Cuerpo de la Respuesta Esperada:**

```
{
  "error": "La ID del ticket es obligatoria"
}
```

- **Prueba exitosa si:** La API devuelve 400 Bad Request con un mensaje de error indicando que el campo ticketId es obligatorio.
- **Pasos de ejecución de la prueba**
  - GG) Se envia la solicitud con el body

The screenshot shows the Postman interface with a successful API call. The method is set to POST, and the endpoint is `http://localhost:8080/api/v1/evidences`. The request body is a JSON object with a single key-value pair: `"url": "https://example.com/evidence1.png"`. The response status is 201 Created, and the timestamp is 4:17 p.m. on 19/10/2025.

HH) Enviar petición y obtener el código 201

The screenshot shows the Postman interface with an unsuccessful API call. The method is set to POST, and the endpoint is `http://localhost:8080/api/v1/evidences`. The request body is an empty JSON object. The response status is 400 Bad Request, and the timestamp is 4:17 p.m. on 19/10/2025. The response body contains the message: `"ticketId": "La ID del ticket es obligatoria"`.

### Respuesta obtenida:

- {
- `"ticketId": "La ID del ticket es obligatoria"`
- }
- 

### Caso de Prueba 3: Error al crear evidencia sin URL

- **Endpoint:** POST /api/evidence
- **Método HTTP:** POST
- **Descripción:** Intentar crear evidencia sin enviar el campo url.
- **Parámetros:** Ninguno.
- **Headers:**
  - Authorization: Bearer <token>
  - Content-Type: application/json
- **Cuerpo de la Solicitud (Body):**

```
{
  "ticketId": 1
}
```

- **Código de Respuesta Esperado:** 400 Bad Request
- **Cuerpo de la Respuesta Esperada:**

```
{
  "error": "La URL de evidencia no puede estar vacía"
}
```

- **Prueba exitosa si:** La API devuelve 400 Bad Request con un mensaje indicando que el campo url es obligatorio.

#### - 6.8 Notification

##### Caso de Prueba 1: Obtener notificaciones de un empleado

- **Endpoint:** GET /api/notifications/employee/{employeeId}
- **Método HTTP:** GET
- **Descripción:** Devuelve todas las notificaciones asociadas a un empleado.
- **Parámetros:**
  - employeeId (path): ID del empleado.
- **Headers:**

- Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 200 OK
- **Cuerpo de la Respuesta Esperada:**

```
[  
 {  
   "id": 45,  
   "message": "Se asignó el ticket 123",  
   "creationDate": "2025-09-10T14:30:00",  
   "isRead": false  
 },  
 {  
   "id": 46,  
   "message": "El ticket 123 fue cerrado",  
   "creationDate": "2025-09-12T10:15:00",  
   "isRead": true  
 }]  
 ]
```

- **Prueba exitosa si:** La API devuelve 200 OK con una lista de notificaciones en el formato esperado.
- **Ejecución de prueba y resultado obtenido:**
  - a) Se pone en el path del endpoint el id del empleado al cual desea ver sus notificaciones

The screenshot shows the Postman interface with the following details:

- Workspace:** My Workspace
- Collection:** Ticket System
- Endpoint:** GET /api/v1/notifications/employee/3
- Method:** GET
- Headers:** (8) - raw
- Body:** JSON - [JSON response]
- Code:** 200 OK
- Response:** JSON array of notifications (highlighted with a red box)

```

[{"id": 3, "message": "Acaban de asignarte el ticket 'Aplicaci\u00f3n de contabilidad se cierra inesperadamente' con prioridad ALTA", "creationDate": "2025-09-23T15:23:34.316481", "read": false}, {"id": 6, "message": "Acaban de asignarte el ticket 'Aplicaci\u00f3n de contabilidad se cierra inesperadamente' con prioridad ALTA", "creationDate": "2025-09-23T15:27:43.98745", "read": false}, {"id": 9, "message": "Acaban de asignarte el ticket 'Error al generar reportes financieros' con prioridad MEDIA", "creationDate": "2025-09-23T15:52:03.847601", "read": false}]
  
```

Bottom right corner shows the date and time: 9:36 p.m. 23/09/2025.

- **¿Prueba Superada?:** Si, aprobada
- **Observaciones:** Se devuelve el código 200 con la lista de notificaciones de ese empleado.

## Caso de Prueba 2: Error al obtener notificaciones de un empleado inexistente

- **Endpoint:** GET /api/notifications/employee/{employeed}
- **M\u00e9todo HTTP:** GET
- **Descripci\u00f3n:** Intenta obtener las notificaciones asociadas a un empleado inexistente
- **Par\u00e1metros:**
  - employeed (path): ID del empleado inexistente (ej 999).
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **C\u00f3digo de Respuesta Esperado:** 404 NOT FOUND
- **Cuerpo de la Respuesta Esperada:**

```
{
  "error": "Empleado no encontrado"
}
```

- **Prueba exitosa si:** La API devuelve 404 Not Found con un mensaje de error.
- **Ejecuci\u00f3n de prueba y resultado obtenido:**
  - Se pone en el path del endpoint el id del empleado inexistente

The screenshot shows the Postman interface with a collection named 'My Workspace'. A specific test case titled 'Obtener notificaciones de empleado' is selected. The method is set to GET, and the URL is {{baseURL}} /api/v1/notifications/employee/999. The response status is 404 Not Found, and the response body is a JSON object with a single key 'error' containing the value 'Empleado no encontrado con id: 999'.

- **¿Prueba Superada?:** Si, aprobada
- **Observaciones:** Se devuelve el código 404 con mensaje de empleado inexistente.

### Caso de Prueba 3: Marcar notificación como leída

- **Endpoint:** PATCH /api/notifications/{notificationId}/read
- **Método HTTP:** PATCH
- **Descripción:** Cambia el estado de una notificación a isRead = true.
- **Parámetros:**
  - notificationId (path): ID de la notificación.
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 200 OK
- **Cuerpo de la Respuesta Esperada:**

{

```

"id": 45,
"message": "Se asignó el ticket 123",
"creationDate": "2025-09-10T14:30:00",
"isRead": true
  
```

{

- **Prueba exitosa si:** La API devuelve 200 OK y la notificación aparece marcada como leída (isRead = true).
- **Ejecución de prueba y resultado obtenido:**
  - a) Se pone en el path del endpoint el id de la notificación a marcar como leída

The screenshot shows the Postman interface with the following details:

- Endpoint:** PATCH {{baseUrl}} /api/v1/notifications/1/read
- Method:** PATCH
- Headers:** Authorization (set to Bearer <token>), Content-Type (set to application/json)
- Body (JSON):**

```
{
  "id": 1,
  "message": "El estado del ticket: 'Aplicación de contabilidad se cierra inesperadamente' acaba de ser actualizado a 'EN PROGRESO'",
  "creationDate": "2025-09-23T15:23:34.022368",
  "read": true
}
```
- Response Status:** 200 OK
- Response Body:**

```
{
  "id": 1,
  "message": "El estado del ticket: 'Aplicación de contabilidad se cierra inesperadamente' acaba de ser actualizado a 'EN PROGRESO'",
  "creationDate": "2025-09-23T15:23:34.022368",
  "read": true
}
```
- Timestamp:** 9:45 p.m. 23/09/2025

- **¿Prueba Superada?:** Si, aprobada
- **Observaciones:** Se devuelve el código 200 con la notificación actualizada como leída.

#### Caso de Prueba 4: Error al marcar notificación inexistente

- **Endpoint:** PATCH /api/notifications/{notificationId}/read
- **Método HTTP:** PATCH
- **Descripción:** Intentar marcar como leída una notificación inexistente.
- **Parámetros:**
  - notificationId (path): ID no existente (ejemplo: 999).
- **Headers:**
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body):** No aplica.
- **Código de Respuesta Esperado:** 404 Not Found
- **Cuerpo de la Respuesta Esperada:**

{

"error": "Notificación no encontrada"

}

- **Prueba exitosa si:** La API devuelve 404 Not Found con un mensaje de error.
- **Ejecución de prueba y resultado obtenido:**
  - a) Se pone en el path del endpoint el id de una notificación existente

The screenshot shows the Postman interface with the following details:

- Left Sidebar:** My Workspace, Collections, Environments, Flows, History.
- Top Bar:** Home, Workspaces, API Network, Search Postman, Invite, Upgrade.
- Middle Area:**
  - HTTP Method:** PATCH
  - Endpoint:** {{baseUrl}} /api/v1/notifications/18/read
  - Headers:** Authorization (green dot), Headers (9)
  - Body:** raw (selected), JSON (dropdown), Preview, Debug with AI.
  - Response:** 404 Not Found, 55 ms, 401 B, Save Response.
  - Body Content:** {"message": "No se ha encontrado la notificación con id: 18"}
- Bottom Bar:** Online, Console, Postbot, Runner, Capture requests, Desktop Agent, Cookies, Vault, Trash, ESP LAA, 9:57 p.m., 23/09/2025.

- **¿Prueba Superada?:** Si, aprobada
- **Observaciones:** Se devuelve el código 404 con el mensaje que la notificación no existe.