PLAN DE PRUEBAS

1. Introducción

- **Nombre del Proyecto**: TicketSystem API
- **Funcionalidad a probar**: Endpoints REST expuestos por la API de TicketSystem.
- **Descripción general**:
  La API de TicketSystem gestiona el flujo de información de un sistema de tickets, incluyendo entidades como Employee, Ticket, Category, Comment, Assignment, TicketRecord, Evidence, Notification. Es fundamental garantizar que los endpoints funcionen correctamente, respetando las reglas de negocio, validaciones de datos y controles de seguridad (autenticación y autorización).

---

2. Alcance de la Prueba

- **Objetivo**: Validar que los endpoints de la API de TicketSystem respondan correctamente ante diferentes entradas y escenarios, cumpliendo con los requisitos funcionales y técnicos definidos en la documentación OpenAPI.
- **Endpoints a probar (nivel general, según OpenAPI)**:
  - POST /api/v1/auth/login
  - POST /api/v1/employees, PUT /api/v1/employees/{id}, DELETE /api/v1/employees/{id}, GET /api/v1/employees/{id}, GET /api/v1/employees, GET /api/v1/employees/email/{email}
  - POST /api/v1/tickets, PUT /api/v1/tickets/{id}/state, DELETE /api/v1/tickets/{id}, GET /api/v1/tickets/{id}, GET /api/v1/tickets, GET /api/v1/tickets/state/{state}, GET /api/v1/tickets/{id}/comments, GET /api/v1/tickets/{id}/tickets-record?from={yyyy-MM-dd}&to={yyyy-MM-dd}
  - GET /api/v1/categories, GET /api/v1/categories/{id}, POST /api/v1/categories, DELETE /api/v1/categories/{id}, PUT /api/v1/categories/{id}
  - GET /api/v1/comments/ticket/{ticketId}, POST /api/v1/comments, DELETE /api/v1/comments/{id}
  - POST /api/v1/assignments, GET /api/v1/assignments/employee/{employeeId}, GET /api/v1/assignments/ticket/{ticketId}, PATCH /api/v1/assignments/reassign/{ticketId}?employeeId={nuevoEmployeeId}
  - GET /api/v1/evidencias/ticket/{ticketId}, GET /api/v1/evidencias/{id}, POST /api/v1/evidencias, DELETE /api/v1/evidencias/{id}
  - GET /api/v1/notifications/employee/{employeeId}, PATCH /api/v1/notifications/{notificationId}/read

---

3. Ambientes de Prueba

- **Entorno de Desarrollo**: http://localhost:8080/api/v1
- **Entorno de Staging**: http://localhost:8080/api/v1
- **Entorno de Producción**: No definido aún

---

4. Criterios de Aceptación

- Retornar el **código de estado HTTP** correspondiente según la operación (200 OK, 201 Created, 204 No Content, 400 Bad Request, 401 Unauthorized, 404 Not Found, 500 Internal Server Error, etc.).
- El **cuerpo de la respuesta** debe coincidir con la especificación definida en OpenAPI (propiedades obligatorias, tipos de datos, formatos).
- Los **headers de respuesta** deben ser correctos (Content-Type: application/json, Authorization cuando aplique).
- Cumplimiento de reglas de negocio (ej: validación de campos obligatorios, unicidad de email en empleados, estado válido de tickets, etc.).
- Respuestas consistentes ante entradas válidas e inválidas.

---

5. Estrategia de Pruebas

5.1 Tipos de Pruebas a Realizar

1. **Pruebas de Solicitud Correcta (Happy Path)**
    - Validar que las solicitudes correctas devuelvan los resultados esperados.
2. **Pruebas de Error (Validación de Errores)**
    - Enviar datos incompletos, incorrectos o en formatos inválidos para confirmar que se devuelven los errores adecuados.
3. **Pruebas de Autenticación y Autorización**
    - Confirmar que los endpoints protegidos no puedan accederse sin credenciales o con permisos insuficientes.
4. **Pruebas de Integridad de Datos**
    - Verificar que la creación, actualización y eliminación de entidades impacten correctamente en las relaciones definidas (ejemplo: comentarios ligados a un ticket existente).

5. **Pruebas de Rendimiento Básico**
    - Validar que los endpoints respondan en un tiempo aceptable bajo condiciones normales de carga.

---

6. Detalles de la Prueba

- **6.1. Employee**

Caso de Prueba 1: Crear un nuevo empleado

- **Endpoint**: POST /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": 101,
```

```
  "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.

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

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


```
{
  "name": "Empleado Prueba",
  "email": "correo_invalido",
  "password": "12345",
  "department": "Ventas"
}
```


- **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.

Caso de Prueba 3: Actualizar un empleado existente

- **Endpoint**: PUT /api/v1/employees/{id}
- **Método HTTP**: PUT
- **Descripción**: Modificar los datos de un empleado ya registrado.
- **Parámetros**:
    - id: 101
- **Cuerpo de petición**:

```
{
  "name": "Carlos Pérez Actualizado",
  "email": "carlos.perez@empresa.com",
  "role": "ADMIN",
  "department": "TI"
}
```

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

```
{
  "id": 101,
  "name": "Carlos Pérez Actualizado",
  "email": "carlos.perez@empresa.com",
  "role": "ADMIN",
  "department": "TI"
}
```

- **Prueba Exitosa si**: Devuelve 200 OK con los datos actualizados.

Caso de Prueba 4: Error al actualizar empleado inexistente

- **Endpoint**: PUT /api/v1/employees/{id}
- **Método HTTP**: PUT
- **Descripción**: Intentar actualizar un empleado que no existe.
- **Parámetros**:
    - id: 9999
- **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."
}
```

- **Prueba Exitosa si**: Devuelve 404 Not Found con mensaje claro.

Caso de Prueba 5: Eliminar empleado existente

- **Endpoint**: DELETE /api/v1/employees/{id}
- **Método HTTP**: DELETE

- **Descripción**: Eliminar un empleado del sistema.
- **Parámetros**:
    - id: 101
- **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.

Caso de Prueba 6: Error al eliminar empleado inexistente

- **Endpoint**: DELETE /api/v1/employees/{id}
- **Método HTTP**: DELETE
- **Descripción**: Intentar eliminar un empleado que no existe.
- **Parámetros**:
    - id: 9999
- **Código de Respuesta Esperado**: 404 Not Found
- **Respuesta esperada**:

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

- **Prueba Exitosa si**: Devuelve 404 Not Found.

Caso de Prueba 7: Obtener empleado por ID

- **Endpoint**: GET /api/v1/employees/{id}
- **Método HTTP**: GET
- **Descripción**: Consultar los datos de un empleado específico.
- **Parámetros**:
    - id: 101

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


```
{
  "id": 101,
  "name": "Carlos Pérez",
  "email": "carlos.perez@empresa.com",
  "role": "USER",
  "department": "Soporte Técnico"
}
```


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

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

- **Endpoint**: GET /api/v1/employees/{id}
- **Método HTTP**: GET
- **Descripción**: Intentar obtener un empleado que no existe.
- **Parámetros**:
  - id: 9999
- **Código de Respuesta Esperado**: 404 Not Found
- **Respuesta esperada**:


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


- **Prueba Exitosa si**: Devuelve 404 Not Found.

Caso de Prueba 9: Listar todos los empleados

- **Endpoint**: GET /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": 101,
    "name": "Carlos Pérez",
    "email": "carlos.perez@empresa.com",
    "role": "USER",
    "department": "Soporte Técnico"
  },
  {
    "id": 102,
    "name": "María López",
    "email": "maria.lopez@empresa.com",
    "role": "ADMIN",
    "department": "Recursos Humanos"
  }
]
```


- **Prueba Exitosa si**: Devuelve 200 OK con una lista válida.

Caso de Prueba 10: Obtener empleado por email

- **Endpoint**: GET /api/v1/employees/email/{email}

- **Método HTTP**: GET
- **Descripción**: Consultar un empleado a partir de su correo electrónico.
- **Parámetros**:
    - email: carlos.perez@empresa.com
- **Código de Respuesta Esperado**: 200 OK
- **Respuesta esperada**:


```
{
  "id": 101,
  "name": "Carlos Pérez",
  "email": "carlos.perez@empresa.com",
  "role": "USER",
  "department": "Soporte Técnico"
}
```


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

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

- **Endpoint**: GET /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."
}
```

- **Prueba Exitosa si**: Devuelve 404 Not Found.

- **6.2. Ticket**

Caso de Prueba 1: Crear un nuevo ticket

- **Endpoint**: POST /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": 1,
  "title": "Error en inicio de sesión",
  "description": "El usuario no puede acceder con credenciales válidas",
  "categoryId": 3,
  "priority": "ALTA"
}
```

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

```
{
  "id": 101,
  "employee": {
```

```
    "id": 1,
    "name": "Juan Pérez"
  },
  "category": {
    "id": 3,
    "name": "Autenticación"
  },
  "title": "Error en inicio de sesión",
  "description": "El usuario no puede acceder con credenciales válidas",
  "priority": "ALTA",
  "state": "ABIERTO",
  "creationDate": "2025-09-14T14:30:00",
  "closingDate": null
}
```


- **Prueba Exitosa si**: Se recibe 201 Created y el ticket contiene todos los datos enviados más el ID generado y estado inicial.

Caso de Prueba 2: Error al crear ticket sin título

- **Endpoint**: POST /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": 1,
  "description": "Error en inicio de sesión",
  "categoryId": 3,
  "priority": "ALTA"
}
```


- **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.

Caso de Prueba 3: Actualizar estado de un ticket

- **Endpoint**: PUT /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_PROCESO"
```

}

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


```
{
  "id": 101,
  "title": "Error en inicio de sesión",
  "state": "EN_PROCESO",
  "priority": "ALTA",
  "employee": {
    "id": 1,
    "name": "Juan Pérez"
  },
  "category": {
    "id": 3,
    "name": "Autenticación"
  },
  "creationDate": "2025-09-14T14:30:00",
  "closingDate": null
}
```


- **Prueba Exitosa si**: El estado se actualiza correctamente y la API devuelve 200 OK con el ticket actualizado.

Caso de Prueba 4: Eliminar un ticket

- **Endpoint**: DELETE /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.

Caso de Prueba 5: Consultar un ticket por ID

- **Endpoint**: GET /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": 101,
  "title": "Error en inicio de sesión",
  "description": "El usuario no puede acceder con credenciales válidas",
  "priority": "ALTA",
  "state": "ABIERTO",
```

```
  "employee": {
    "id": 1,
    "name": "Juan Pérez"
  },
  "category": {
    "id": 3,
    "name": "Autenticación"
  },
  "creationDate": "2025-09-14T14:30:00",
  "closingDate": null
}
```


- **Prueba Exitosa si**: La API devuelve 200 OK y los datos del ticket corresponden al ID solicitado.

Caso de Prueba 6: Listar todos los tickets

- **Endpoint**: GET /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": 101,
```

```
  "title": "Error en inicio de sesión",
  "description": "El usuario no puede acceder con credenciales válidas",
  "priority": "ALTA",
  "state": "ABIERTO",
  "employee": {
    "id": 1,
    "name": "Juan Pérez"
  },
  "category": {
    "id": 3,
    "name": "Autenticación"
  },
  "creationDate": "2025-09-14T14:30:00",
  "closingDate": null
},
{
  "id": 102,
  "title": "Problema con impresión",
  "description": "La impresora no responde al enviar trabajos",
  "priority": "MEDIA",
  "state": "EN_PROCESO",
  "employee": {
    "id": 2,
    "name": "Laura Gómez"
  },
  "category": {
    "id": 4,
    "name": "Hardware"
```

```
  },
  "creationDate": "2025-09-13T10:15:00",
  "closingDate": null
 }
]
```


- **Prueba Exitosa si**: La API devuelve 200 OK y una lista con todos los tickets existentes en el sistema.

Caso de Prueba 7: Listar tickets por estado

- **Endpoint**: GET /api/v1/tickets/state/{state}
- **Método HTTP**: GET
- **Descripción**: Obtener una lista de tickets filtrados por estado.
- **Parámetros**:
  - state (path) → Estado del ticket (ej: ABIERTO, EN_PROCESO, CERRADO).
- **Headers**:
  - Authorization: Bearer <token>
- **Cuerpo de la Solicitud (Body)**: No aplica.
- **Código de Respuesta Esperado**: 200 OK
- **Cuerpo de la Respuesta Esperada**:


```
[
 {
  "id": 101,
  "title": "Error en inicio de sesión",
  "priority": "ALTA",
  "state": "ABIERTO",
  "employee": {
   "id": 1,
```

```
    "name": "Juan Pérez"
  },
  "category": {
    "id": 3,
    "name": "Autenticación"
  },
  "creationDate": "2025-09-14T14:30:00",
  "closingDate": null
 }
]
```


- **Prueba Exitosa si**: La API devuelve 200 OK con solo los tickets que tienen el estado solicitado.

Caso de Prueba 8: 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": 101,
    "employee": {
      "id": 1,
      "name": "Juan Pérez"
    },
    "text": "El ticket fue revisado y se asignó al equipo de soporte.",
    "createdAt": "2025-09-13T15:30:00"
  },
  {
    "id": 2,
    "ticketId": 101,
    "employee": {
      "id": 2,
      "name": "Laura Gómez"
    },
    "text": "Se identificó que el problema está en la base de datos.",
    "createdAt": "2025-09-13T16:00:00"
  }
]
```


- **Prueba Exitosa si**: La API devuelve 200 OK con todos los comentarios relacionados al ticket indicado.

Caso de Prueba 9: 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,
    "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.

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.
    - to (query) → Fecha de fin.
- **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 no encontrado"
}
```

- **Prueba Exitosa si**: La API devuelve código 404 y un mensaje indicando que el ticket no existe.

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.

- **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**:

```
[
```

```
 {
   "id": 1,
   "name": "Hardware",
   "description": "Problemas relacionados con equipos físicos"
 },
 {
   "id": 2,
   "name": "Software",
   "description": "Errores o incidencias en aplicaciones"
 }
]
```


- **Prueba Exitosa si**: La API devuelve 200 OK con la lista completa de categorías.

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.
    - 404 Not Found si no existe la categoría con ese ID.
- **Cuerpo de la Respuesta Esperada** (ejemplo éxito):

```
{
  "id": 1,
  "name": "Hardware",
  "description": "Problemas relacionados con equipos físicos"
}
```


- **Prueba Exitosa si**: La API devuelve 200 OK con la categoría correspondiente o 404 si no existe.

Caso de Prueba 3: 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**:
    - 201 Created si la categoría se creó correctamente.
    - 409 Conflict si ya existe una categoría con ese nombre.
- **Cuerpo de la Respuesta Esperada (ejemplo éxito)**:


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

"name": "Redes",

"description": "Problemas relacionados con conectividad"

}

- **Prueba Exitosa si**: La API devuelve 201 Created y los datos de la nueva categoría.

Caso de Prueba 4: 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.
    - 404 Not Found si no existe la categoría.
- **Prueba Exitosa si**: La API devuelve 204 No Content al eliminar la categoría indicada.

Caso de Prueba 5: 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.
    - 404 Not Found si no existe la categoría.
- **Cuerpo de la Respuesta Esperada (ejemplo éxito)**:

```
{
  "id": 1,
  "name": "Hardware y periféricos",
  "description": "Problemas relacionados con dispositivos físicos"
}
```

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

- **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** → Devuelve la lista de comentarios asociados al ticket.
- **204 No Content** → El ticket no tiene comentarios registrados.
- **401 Unauthorized** → Token inválido o ausente.

Cuerpo de la Respuesta Esperada (200 OK):


```
[
  {
    "id": 1,
    "ticketId": 101,
    "employee": {
  "id": 101,
  "name": "Carlos Pérez Actualizado",
  "email": "carlos.perez@empresa.com",
  "role": "ADMIN",
  "department": "TI"
},
    "text": "Este es un comentario de prueba",
    "createdAt": "2025-09-14T10:00:00"
  },
  {
    "id": 2,
    "ticketId": 101,
```

```
    "employee": {
  "id": 102,
  "name": "Carlos Pérez",
  "email": "carlos.perez@empresa.com",
  "role": "AGENT",
  "department": "TI"
}
    "text": "Segundo comentario del mismo ticket",
    "createdAt": "2025-09-14T11:30:00"
  }
]
```


Prueba Exitosa si:

- La API devuelve **200 OK** con una lista de comentarios pertenecientes al ticketId.
- Cada comentario contiene los campos esperados (id, ticketId, employee, text, createdAt).
- Si el ticket no tiene comentarios, responde con **204 No Content**.

Caso de Prueba 2: Obtener comentario por ID

- **Endpoint**: GET /comments/{id}
- **Método HTTP**: GET
- **Descripción**: Consultar un comentario específico mediante su ID.
- **Parámetros**:
    - id (path): ID del comentario.
- **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": 1,
  "ticketId": 101,
  "employee":{
  "id": 101,
  "name": "Carlos Pérez Actualizado",
  "email": "carlos.perez@empresa.com",
  "role": "ADMIN",
  "department": "TI"
},
  "text": "Este es un comentario de prueba",
  "createdAt": "2025-09-14T10:00:00"
}
```


- **Prueba Exitosa si**: La API devuelve 200 OK y retorna la información del comentario solicitado.

Caso de Prueba 3: Error al obtener comentario inexistente

- **Endpoint**: GET /comments/{id}
- **Método HTTP**: GET
- **Descripción**: Intentar consultar 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 con un mensaje indicando que el comentario no existe.

Caso de Prueba 4: 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": 101,
  "employeeId": 1,
  "text": "Nuevo comentario de prueba"
}
```

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


```
{
  "id": 3,
  "ticketId": 101,
  "employee": {
  "id": 1,
  "name": "Carlos Pérez Actualizado",
  "email": "carlos.perez@empresa.com",
  "role": "ADMIN",
  "department": "TI"
},
  "text": "Nuevo comentario de prueba",
  "createdAt": "2025-09-14T12:00:00"
}
```


- **Prueba Exitosa si**: La API devuelve 201 Created y retorna el comentario creado con su ID asignado.

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": 101,
  "employeeId": 1
}
```

- **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.

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.

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.

- **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": 123,
  "employeeId": 7
}
```

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

```
{
  "id": 10,
  "ticket": {
    "id": 123,
    "title": "Error en login"
  },
  "employee": {
    "id": 7,
    "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.

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 employeeId.
- **Parámetros**: Ninguno.
- **Headers**:
    - Authorization: Bearer <token>
    - Content-Type: application/json
- **Cuerpo de la Solicitud (Body)**:

```
{
  "ticketId": null,
  "employeeId": 7
}
```

- **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.

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.

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.

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.

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.

Caso de Prueba 7: Reasignar agente

- **Endpoint**: PATCH /api/v1/assignments/reassign/{id}?employeeId={nuevoId}
- **Método HTTP**: PATCH
- **Descripción**: Cambiar el empleado asignado a un ticket existente.
- **Parámetros**:
    - id (path): ID de la asignación.
    - employeeId (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.

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

- **Endpoint**: PATCH /api/v1/assignments/reassign/{id}?employeeId={nuevoId}
- **Método HTTP**: PATCH
- **Descripción**: Intentar reasignar a un empleado que no existe.
- **Parámetros**:
    - id (path): ID de la asignación.
    - employeeId (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.

- **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": 10,
  "ticketId": 1,
  "url": "https://example.com/evidence1.png"
}
```


- **Prueba Exitosa si**: La API devuelve 201 Created y la evidencia creada coincide con los datos enviados.

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.

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 vacia"
}
```

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

Caso de Prueba 4: Listar evidencias de un ticket

- **Endpoint**: GET /api/evidence/ticket/{ticketId}

- **Método HTTP**: GET
- **Descripción**: Devuelve todas las evidencias asociadas a 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": 10,
    "ticketId": 1,
    "url": "https://example.com/evidence1.png"
  },
  {
    "id": 11,
    "ticketId": 1,
    "url": "https://example.com/evidence2.png"
  }
]
```


- **Prueba Exitosa si**: La API devuelve 200 OK con la lista de evidencias asociadas al ticket.

Caso de Prueba 5: Error al listar evidencias de un ticket inexistente

- **Endpoint**: GET /api/evidence/ticket/{ticketId}

- **Método HTTP**: GET
- **Descripción**: Intentar obtener evidencias de un ticket inexistente.
- **Parámetros**:
    - ticketId (path): ID de ticket no registrado, ej. 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": "Ticket no encontrado"
}
```


- **Prueba Exitosa si**: La API devuelve 404 Not Found con un mensaje indicando que el ticket no existe.

- **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.

Caso de Prueba 2: 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).

Caso de Prueba 6: 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.

---

8. Conclusión

El resultado de las pruebas permitirá determinar si la API de TicketSystem cumple con los criterios de aceptación definidos. En caso afirmativo, la funcionalidad se considera lista para despliegue; de lo contrario, se documentarán las incidencias y se solicitarán correcciones antes de avanzar al siguiente entorno.

En desarrollo el resultado de pruebas para poner todas las conclusiones