Universidad ORT Uruguay

Facultad de Ingeniería

https://github.com/IngSoft-DA2-2023-2/276280-243218

# 1er Obligatorio Diseño de Aplicaciones

*Evidencia del diseño y especificación de la API.*

Gonzalo Loureiro – 243218

Santiago Alfonso – 276280

*02 de Mayo de 2024*

# Índice

## Descripción general

En la especificación del proyecto, se describió la necesidad de desarrollar una API que funcione como sistema para gestionar solicitudes de mantenimiento en un apartamento. Hemos diseñado un sistema en el que un Administrador (Admin) es responsable de crear perfiles de Encargados (Managers). El Admin genera una invitación que los Managers deben aceptar, y también se encarga de definir las categorías de las solicitudes de mantenimiento.

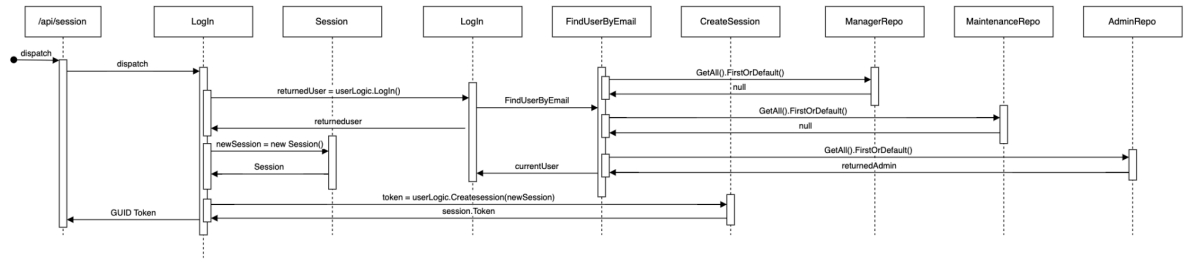
Los Managers tienen la tarea de crear perfiles para los Empleados de Mantenimiento. Además, los Managers están encargados de crear y asignar solicitudes de mantenimiento a estos empleados, quienes a su vez pueden aceptarlas y marcarlas como completadas. Los Managers también tienen la capacidad de acceder a reportes detallados, ya sea por edificio o por empleado.

Adicionalmente, es posible crear registros de edificios y apartamentos, los cuales son necesarios para generar las solicitudes de mantenimiento. Considerando la seguridad y la integridad del sistema, se ha implementado un endpoint específico para autenticación. Al realizar el login, el sistema retorna un token que es utilizado para autenticar las interacciones con la API dependiendo del rol del usuario (Admin, Manager, Mantenimiento). Es crucial destacar que los roles son mutuamente excluyentes; es decir, un usuario no puede desempeñar más de un rol al mismo tiempo.

# Autenticación

Para adecuarnos a los requerimientos del sistema, decidimos diferenciar los recursos que requieren autenticación de aquellos que no. Los recursos accesibles sin autenticación incluyen los datos del propietario del apartamento y los detalles del apartamento, así como el endpoint de sesión, que se usa para generar los tokens necesarios para la autenticación en otros recursos. Además, el proceso de aceptación de una invitación también se puede realizar sin necesidad de autenticarse, aunque para crear, eliminar o consultar invitaciones existentes sí se requiere autenticación.

Para la autenticación de solicitudes, hemos implementado filtros de autenticación basados en roles. El flujo de Autenticacion funciona de la siguiente manera:




1. Inicio de Sesión: El usuario envía una solicitud al endpoint /api/session, que es accesible sin autenticación. Este endpoint procesa las credenciales y, si son válidas, devuelve un token en formato GUID.
2. Uso del Token: Este token debe incluirse en el encabezado Authentication de cualquier solicitud subsiguiente a recursos que requieran autenticación. El sistema valida este token y permite el acceso a la funcionalidad basada en el rol del usuario autenticado.

Esta estructura asegura que solo los usuarios autorizados puedan acceder a funciones específicas de la API, manteniendo la seguridad y la integridad del sistema.

## Códigos de Estado

En este proyecto, hemos decidido utilizar un conjunto específico de códigos de estado HTTP para ofrecer respuestas más precisas y facilitar el desarrollo al reducir la cantidad de casos límite a probar. Los códigos seleccionados y sus usos son los siguientes:

- **200 OK**: Utilizado cuando una solicitud se completa con éxito. Esto incluye la creación de nuevos recursos (como Apartamentos, Invitaciones, Edificios, etc.) o la realización exitosa de acciones, como obtener una lista de todos los Administradores. Este es el código de estado más común y confirma que la solicitud ha sido recibida, entendida y aceptada correctamente.
- **204 No Content**: Empleado en situaciones donde la solicitud ha sido exitosa pero no es necesario enviar contenido de vuelta, como en las operaciones de borrado (DELETE). Esto ayuda a mantener la comunicación eficiente y clara.
- **400 Bad Request**: Este código se utiliza cuando hay errores en los datos proporcionados por el usuario, como un formato de correo electrónico inválido. Sirve para informar al usuario que debe corregir los datos enviados para que la solicitud pueda ser procesada correctamente.
- **401 Unauthorized**: Indica que para acceder al recurso solicitado se requieren credenciales válidas, que no han sido proporcionadas o son incorrectas. Se usa específicamente cuando se intenta acceder a un recurso sin un token o con un token de formato incorrecto. Solo se aceptan tokens de tipo GUID.
- **403 Forbidden**: Similar al 401, pero se utiliza cuando el usuario tiene autenticación pero no tiene los permisos necesarios para realizar la acción. Por ejemplo, si un administrador intenta realizar una acción que sólo los Manager pueden ejecutar, se utilizará este código para indicar la falta de permisos.
- **404 Not Found**: Se emplea cuando el recurso solicitado no se encuentra disponible, ya sea porque la URL no existe o el recurso específico (como un usuario con un email específico) no existe. Ayuda a clarificar que el elemento solicitado no se puede encontrar en el servidor.

- **500 Internal Server Error**: Utilizado como respuesta genérica cuando el servidor encuentra una condición inesperada que impide cumplir con la solicitud, como errores con la base de datos. Este código señala problemas internos del servidor que necesitan ser resueltos.

Estos códigos nos permiten manejar las respuestas del servidor de manera más específica y adecuada, lo que contribuye a un desarrollo más ágil y una mejor experiencia para el usuario al identificar claramente el tipo de error o éxito de la solicitud.

# Recursos

## Admins (Administradores)

- Descripción: Este recurso maneja la información relacionada con los administradores del sistema, quienes tienen acceso y control sobre diversas funciones de la plataforma.
- Endpoints:
    - GET /api/admins
        - Parámetros: Ninguno
        - Código de error: 200, 404, 401
    - POST /api/admins
        - Parámetros: Ninguno
        - Ejemplo de cuerpo de solicitud:

```
{
  "name": "John",
  "lastName": "Doe",
  "password": "pass123",
  "email": "john.doe@example.com"
}
```

        - Código de error: 200, 404, 401
    - GET /api/admins/{id}
        - Parámetros: id: integer (path, requerido)
        - Código de error: 200, 404, 401
    - PUT /api/admins/{id}
        - Parámetros: id: integer (path, requerido)
        - Ejemplo de cuerpo de solicitud:

```
{
  "name": "John",
  "lastName": "Doe",
  "password": "pass123"
}
```

        - Código de error: 200, 404, 401

- DELETE /api/admins/{id}
    - Parámetros: id: integer (path, requerido)
    - Código de error: 200, 204, 404, 401

## Apartments (Departamentos)

- Descripción: Este recurso se encarga de gestionar la información relacionada con los departamentos dentro de los edificios, incluyendo detalles como ubicación, dueños, etc.
- Endpoints:
    - GET /api/apartments
        - Parámetros: Ninguno
        - Código de error: 200, 404, 401
    - POST /api/apartments
        - Parámetros: Ninguno
        - Ejemplo de cuerpo de solicitud:

```
{
  "floor": 2,
  "number": 101,
  "numberOfBedrooms": 3,
  "numberOfBathrooms": 2,
  "terrace": true,
  "ownerId": 1,
  "buildingId": 1
}
```

        - Código de error: 200, 404, 401
    - GET /api/apartments/{id}
        - Parámetros: id: integer (path, requerido)
        - Código de error: 200, 404, 401
    - DELETE /api/apartments/{id}
        - Parámetros: id: integer (path, requerido)
        - Código de error: 200, 204, 404, 401

## Apartment Owners (Dueños de Departamento)

- Descripción: Este recurso gestiona la información de los propietarios de los departamentos.
- Endpoints:
  - GET /api/ApartmentOwners/{id}
    - Parámetros: id: integer (path, requerido)
    - Código de error: 200, 404, 401
  - PUT /api/ApartmentOwners/{id}
    - Parámetros: id: integer (path, requerido)
    - Ejemplo de cuerpo de solicitud:

      ```
      {
        "name": "Jane",
        "lastName": "Smith",
        "email": "jane.smith@example.com"
      }
      ```
    - Código de error: 200, 404, 401
  - DELETE /api/ApartmentOwners/{id}
    - Parámetros: id: integer (path, requerido)
    - Código de error: 200, 204, 404, 401
  - POST /api/ApartmentOwners
    - Parámetros: Ninguno
    - Ejemplo de cuerpo de solicitud:

      ```
      {
        "name": "Jane",
        "lastName": "Smith",
        "email": "jane.smith@example.com"
      }
      ```
    - Código de error: 200, 404, 401

## Buildings (Edificios)

- Descripción: Este recurso se encarga de administrar la información relacionada con los edificios, como su ubicación, constructores, gastos comunes, etc.
- Endpoints:
  - GET /api/building
    - Parámetros: Ninguno
    - Código de error: 200, 404, 401
  - POST /api/building
    - Parámetros: Ninguno
    - Ejemplo de cuerpo de solicitud:

```
{
  "name": "Malvin Towers",
  "address": "123 Main St",
  "latitude": 40.7128,
  "longitude": -74.0060,
  "constructionCompany": "BuildIt Co.",
  "commonExpenses": 200,
  "managerAssociatedId": 2
}
```

    - Código de error: 200, 404, 401
  - GET /api/building/{id}
    - Parámetros: id: integer (path, requerido)
    - Código de error: 200, 404, 401
  - PUT /api/building/{id}
    - Parámetros: id: integer (path, requerido)
    - Ejemplo de cuerpo de solicitud:

```
{
  "constructionCompany": "BuildIt Co.",
  "commonExpenses": 200
}
```

    - Código de error: 200, 404, 401
  - DELETE /api/building/{id}
    - Parámetros: id: integer (path, requerido)
    - Código de error: 200, 204, 404, 401

## Categories (Categorías)

- Descripción: Este recurso gestiona las categorías disponibles para las solicitudes de mantenimiento de los edificios.
- Endpoints:
    - GET /api/categories
        - Parámetros: Ninguno
        - Código de error: 200, 404, 401
    - POST /api/categories
        - Parámetros: Ninguno
        - Ejemplo de cuerpo de solicitud:

          ```
          {
            "name": "Utilities"
          }
          ```
        - Código de error: 200, 404, 401
    - GET /api/categories/{id}
        - Parámetros: id: integer (path, requerido)
        - Código de error: 200, 404, 401
    - DELETE /api/categories/{id}
        - Parámetros: id: integer (path, requerido)
        - Código de error: 200, 204, 404, 401

## Invitations (Invitaciones)

- Descripción: Este recurso maneja las invitaciones enviadas a usuarios para unirse a la plataforma con roles específicos.
- Endpoints:
    - GET /api/invitations
        - Parámetros: Ninguno
        - Código de error: 200, 404, 401

- POST /api/invitations
    - Parámetros: Ninguno
    - Ejemplo de cuerpo de solicitud:

      ```
      {
        "email": "invitee@example.com",
        "name": "Invitee",
        "deadLine": "2023-12-31T23:59:59",
        "creatorId": 1
      }
      ```
    - Código de error: 200, 404, 401
- GET /api/invitations/{id}
    - Parámetros: id: integer (path, requerido)
    - Código de error: 200, 404, 401
- PUT /api/invitations/{id}
    - Parámetros: id: integer (path, requerido)
    - Ejemplo de cuerpo de solicitud:

      ```
      {
        "email": "invitee@example.com",
        "password": "secure123",
        "acceptInvitation": true
      }
      ```
    - Código de error: 200, 404, 401
- DELETE /api/invitations/{id}
    - Parámetros: id: integer (path, requerido)
    - Código de error: 200, 204, 404, 401

## Maintenance Staff (Personal de Mantenimiento)

- Descripción: Este recurso gestiona la información relacionada con el personal encargado del mantenimiento de los edificios.
- Endpoints:
    - GET /api/MaintenanceStaff
        - Parámetros: Ninguno
        - Código de error: 200, 404, 401
    - POST /api/MaintenanceStaff
        - Parámetros: Ninguno
        - Ejemplo de cuerpo de solicitud:

```
{
  "name": "Mike",
  "lastName": "Johnson",
  "password": "maint123",
  "email": "mike.johnson@example.com",
  "associatedBuildingId": 3
}
```

        - Código de error: 200, 404, 401
    - GET /api/MaintenanceStaff/{id}
        - Parámetros: id: integer (path, requerido)
        - Código de error: 200, 404, 401
    - DELETE /api/MaintenanceStaff/{id}
        - Parámetros: id: integer (path, requerido)
        - Código de error: 200, 204, 404, 401

## Manager Requests (Solicitudes del Encargado)

- Descripción: Este recurso maneja las solicitudes creadas por los encargados para solicitar mantenimiento en los departamentos.
- Endpoints:
  - GET /api/manager/requests
    - Parámetros: category: string (query, opcional)
    - Código de error: 200, 404, 401
  - POST /api/manager/requests
    - Parámetros: Ninguno
    - Ejemplo de cuerpo de solicitud:

```
{
  "description": "Luz Rota",
  "departmentId": 1,
  "categoryId": 2
}
```

    - Código de error: 200, 404, 401
  - POST /api/manager/assign
    - Parámetros: Ninguno
    - Ejemplo de cuerpo de solicitud:

```
{
  "requestId": 1,
  "maintenanceId": 401
}
```

    - Código de error: 200, 404, 401
  - PUT /api/manager/requests/accept
    - Parámetros: Ninguno
    - Ejemplo de cuerpo de solicitud:

```
{
  "requestId": 1
}
```

    - Código de error: 200, 404, 401

- PUT /api/manager/requests/complete
    - Parámetros: Ninguno
    - Ejemplo de cuerpo de solicitud:

```
{
  "requestId": 1,
  "finalPrice": 500
}
```

    - Código de error: 200, 404, 401

## Reports (Reportes)

- Descripción: Este recurso proporciona funcionalidades para generar informes sobre las solicitudes y el mantenimiento de los edificios.
- Endpoints:
    - GET /api/reports/building
    - Parámetros: Ninguno
        - Ejemplo de cuerpo de solicitud:

```
{
  "buildingId": 3
}
```

        - Código de error: 200, 404, 401
    - GET /api/reports/maintenance
        - Parámetros: Ninguno
        - Ejemplo de cuerpo de solicitud:

```
{
  "workerName": "Mike",
  "buildingId": 3
}
```

        - Código de error: 200, 404, 401

## Session Management (Gestión de Sesiones)

- Descripción: Este recurso se encarga de gestionar las sesiones de usuario en la plataforma.
- Endpoints:
    - POST /api/session
        - Parámetros: Ninguno
        - Ejemplo de cuerpo de solicitud:

```
{
  "email": "user@example.com",
  "password": "securepass"
}
```

        - Código de error: 200, 404, 401