Interfaz de programación de aplicaciones (API) - Sistema Mantenimiento CALP.

Versión 2.0.0

—

Cooperativa Agropecuaria La Paz Limitada (CALP)

Cabana, Iahn, Zandomeni.

26.07.2024

Índice

Loguearse

Este endpoint permite mediante un correo electrónico válido registrado en el sistema y su contraseña, poder obtener los datos del usuario que se loguea, y además un token que almacenará el id del usuario, esto esencial para validar en otros endpoints, que dicho usuario sea válido.

Tipo de petición: POST

URL: http://localhost:8080/api/auth/login

Requisitos: se debe de llamar a la api mediante una petición POST, pasándole un JSON que tenga las siguientes keys: email y password.

FetchAPI ejemplo de uso:

```
const body = {
      email: "dev-calp@calp.com",
      password: "testadmin123"
}

fetch('http://localhost:8080/api/auth/login',{
     method: 'POST',
     headers: {
          'Content-Type': 'application/json'
     },
     body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json));
.catch(err => console.log(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

usuario: objeto que contiene los datos del usuario creado, incluyendo su ID, nombre, apellido y correo electrónico.

token: el token que almacena el id del usuario.

---

1) Creación de usuarios

Este endpoint permite crear un nuevo usuario en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud (Debe de validar que el usuario tenga el rol de Gerente Operativo).

Tipo de petición: POST

URL: http://localhost:8080/api/usuarios

Requisitos: Se debe de llamar a la api mediante una petición POST, enviando en los headers el TOKEN correspondiente al usuario, el cual es necesario para identificar que un usuario valido solo pueda insertar usuarios en la bd.

El cuerpo de la solicitud debe ser un objeto JSON con las siguientes claves:

- nombre: Nombre del nuevo usuario.
- apellido: Apellido del nuevo usuario.
- email: Correo electrónico del nuevo usuario.
- password: Contraseña del nuevo usuario.
- rol: Rol del nuevo usuario (GO para Gerente Operativo, JM para Jefe de Mantenimiento, EA para Encargado de Área).
- id_area (obligatorio si el rol es EA): ID del área a la que se asignará el usuario, el **ID del área debe existir en la BD**.

- id_deposito (opcional): ID del depósito al que se asignará el usuario (solo para Jefes de Mantenimiento).

FetchAPI ejemplo de uso:

```
const body = {
      nombre: "NombreUsuario",
      apellido: "ApellidoUsuario",
      email: "usuario@example.com",
      password: "contraseña123",
      rol: "JM", // Reemplaza con el rol deseado: GO, JM
o EA
      id_area: 123 // Opcional: ID del área (solo para
EA)
};

fetch('http://localhost:8080/api/usuarios', {
      method: 'POST',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

usuario: objeto que contiene los datos del usuario creado, incluyendo su ID, nombre, apellido, correo electrónico y rol.

---

2) Actualizar usuarios

Este endpoint permite actualizar los datos de un usuario existente en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido (Debe de validar que el usuario tenga el rol de Gerente Operativo) en el encabezado de la solicitud y especificar el ID del usuario que se desea actualizar en la URL.

Tipo de petición: PUT

URL: http://localhost:8080/api/usuarios/{id} (reemplaza {id} con el ID del usuario que deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al usuario válido.
- Se debe especificar el ID del usuario que se desea actualizar en la URL.
- En el cuerpo de la solicitud, se pueden incluir **uno** o **más** de los siguientes campos para actualizar:

    nombre: Nuevo nombre del usuario.

    apellido: Nuevo apellido del usuario.

    email: Nuevo correo electrónico del usuario.

    password: Nueva contraseña del usuario.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: "NuevoNombre",
      apellido: "NuevoApellido",
```

```
      email: "nuevo@email.com",
      password: "nuevacontraseña123"
};
fetch('http://localhost:8080/api/usuarios/{id}', {
      method: 'PUT',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

usuario: objeto que contiene los datos actualizados del usuario, incluyendo su ID, nombre, apellido, correo electrónico y rol.

---

3) Baja (lógica) de usuario

Este endpoint permite desactivar (eliminar lógicamente) un usuario existente en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido (GO) en el encabezado de la solicitud y especificar el ID del usuario que se desea eliminar en la URL.

Tipo de petición: DELETE

URL: http://localhost:8080/api/usuarios/{id} (reemplaza {id} con el ID del usuario que deseas eliminar)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al usuario válido.
- Se debe especificar el ID del usuario que se desea eliminar en la URL.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/usuarios/{id}', {
      method: 'DELETE',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que el usuario ha sido desactivado con éxito.

---

4) Demitir Área de Encargado (Eliminar el área al encargado)

Este endpoint permite demitir (eliminar lógicamente) el área asignada a un encargado de área en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID del usuario encargado de área cuyo área se desea eliminar en la URL.

Tipo de petición: DELETE

URL: http://localhost:8080/api/usuarios/areas/{id} (reemplaza {id} con el ID del encargado de área cuyo área deseas eliminar)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al usuario válido.
- Se debe especificar el ID del usuario encargado de área cuyo área se desea eliminar en la URL.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/usuarios/areas/{id}', {
      method: 'DELETE',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que el área del encargado ha sido demitida con éxito.

---

5) Actualizar Área de Encargado

Este endpoint permite actualizar el área asignada a un encargado de área en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido **(Debe ser GO)** en el encabezado de la solicitud y especificar el ID del usuario encargado de área cuyo área se desea actualizar, en la URL. Además, en el cuerpo de la solicitud, se debe proporcionar el nuevo ID del área que se asignará al encargado.

Tipo de petición: PUT

URL: http://localhost:8080/api/usuarios/areas/{id} (reemplaza {id} con el ID del encargado de área cuyo área deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al usuario válido.
- Se debe especificar en la URL el ID del usuario encargado de área cuyo área se desea actualizar.

- En el cuerpo de la solicitud, se debe incluir el campo id_area con el nuevo ID del área **(debe ser un área existente y que no esté asignada a otro EA)**.

FetchAPI ejemplo de uso:

```
const body = {
      id_area:      NUEVO_ID_DE_AREA      //      Reemplaza
NUEVO_ID_DE_AREA con el ID del nuevo área
};
fetch('http://localhost:8080/api/usuarios/areas/{id}', {
      method: 'PUT',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

  msg: mensaje que indica que el área del encargado ha sido actualizada con éxito.

---

6) Actualizar Deposito de Jefe

Este endpoint permite actualizar el depósito asignado a un jefe de mantenimiento en la base de datos. Para realizar esta acción, es

necesario proporcionar un token de autenticación válido **(Debe ser GO)** en el encabezado de la solicitud y especificar el ID del jefe de mantenimiento cuyo depósito se desea actualizar, en la URL. Además, en el cuerpo de la solicitud, se debe proporcionar el nuevo ID del depósito que se asignará al jefe.

Tipo de petición: PUT

URL: http://localhost:8080/api/usuarios/depositos/{id} (reemplaza {id} con el ID del jefe de mantenimiento cuyo depósito deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al usuario válido.
- Se debe especificar en la URL el ID del usuario jefe de mantenimiento cuyo depósito se desea actualizar.
- En el cuerpo de la solicitud, se debe incluir el campo id_deposito con el nuevo ID del depósito **(debe ser un depósito existente y que no esté asignado a otro JM)**.

FetchAPI ejemplo de uso:

```
const body = {
      id_deposito:   NUEVO_ID_DE_DEPOSITO   //   Reemplaza
NUEVO_ID_DE_DEPOSITO con el ID del nuevo deposito
};
fetch('http://localhost:8080/api/usuarios/depositos/{id}'
, {
      method: 'PUT',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
```

```
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK

- JSON de respuesta:

  msg: mensaje que indica que el depósito del jefe de mantenimiento ha sido actualizado con éxito.

---

7) Creación de Área

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo**, agregar una nueva área en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el nombre del área que se desea crear en el cuerpo de la solicitud.

Tipo de petición: POST

URL: http://localhost:8080/api/areas

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- En el cuerpo de la solicitud, se debe incluir el campo nombre con el nombre del área que se desea crear **(El nombre del área no debe repetirse con algún otra área existente)**.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: 'Nombre del Área' // Reemplaza 'Nombre del
Área' con el nombre del área que deseas crear
};
fetch('http://localhost:8080/api/areas', {
      method: 'POST',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

area: objeto que contiene información sobre el área recién creada, incluyendo su nombre.

---

8) Actualización de Área

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo**, actualizar el nombre de un área específica en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud, especificar el ID del área que se desea actualizar en la URL y proporcionar el nuevo nombre del área en el cuerpo de la solicitud.

Tipo de petición: PUT

URL: http://localhost:8080/api/areas/{id} (reemplaza {id} con el ID del área que deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID del área que se desea actualizar en la URL.
- En el cuerpo de la solicitud, se debe incluir el campo nombre con el nombre del área que se desea crear **(El nombre del área no debe repetirse con algún otra área existente)**.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: 'Nuevo Nombre del Área' // Reemplaza 'Nuevo Nombre del Área' con el nuevo nombre del área que deseas asignar
```

```
};

fetch('http://localhost:8080/api/areas/{id}',{
      method: 'PUT',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que el nombre del área se ha actualizado con éxito.

---

9) Baja (lógica) de Área

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo**, eliminar un área específica de la base de datos de manera lógica. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID del área que se desea eliminar en la URL.

Tipo de petición: DELETE

URL: http://localhost:8080/api/areas/{id} (reemplaza {id} con el ID del área que deseas eliminar)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento válido.
- Se debe especificar el ID del área que se desea eliminar en la URL.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/areas/{id}', {
      method: 'DELETE',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      }
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que el área se ha eliminado con éxito.

---

10) Creación de Sector en un Área

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** agregar un nuevo sector en un área específica en la base de

datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud, especificar el ID del área en la URL donde se desea crear el sector y proporcionar el nombre del sector en el cuerpo de la solicitud.

Tipo de petición: POST

URL: http://localhost:8080/api/areas/{id} (reemplaza {id} con el ID del área en la que deseas crear el sector)

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID del área **(Debe ser un ID válido de un área existente que no tenga el estado en false)** en la URL donde se desea crear el sector.
- En el cuerpo de la solicitud, se debe incluir el campo nombre con el nombre del sector que se desea crear.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: 'Nombre del Sector' // Reemplaza 'Nombre
del Sector' con el nombre del sector que deseas crear
};
fetch('http://localhost:8080/api/areas/{id}', {
      method: 'POST',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
```

```
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

sector: objeto que contiene información sobre el sector recién creado, incluyendo su nombre.

---

11) Actualización de Sector

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** actualizar un sector específico en la base de datos, incluyendo su nombre o el área al que pertenece. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID del sector que se desea actualizar en la URL.

Tipo de petición: PUT

URL: http://localhost:8080/api/sectores/{id} (reemplaza {id} con el ID del sector que deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID del sector **(Debe ser un ID válido de un sector existente que no tenga el estado en false)** que se desea actualizar en la URL.
- Se puede proporcionar en el cuerpo de la solicitud:

  nombre (opcional): El nuevo nombre del sector.

  id_area (opcional): El nuevo ID del área **(Debe ser un ID válido)** al que pertenece el sector.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: "Nuevo Nombre del Sector",
      id_area: 2 // Reemplaza con el ID del área
correspondiente (opcional)
}
fetch('http://localhost:8080/api/sectores/{id}', {
      method: 'PUT',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que el sector se ha actualizado con éxito.

sector: datos actualizados del sector, incluyendo su nombre y el ID del área a la que pertenece (si se proporcionaron en la solicitud).

---

12) Baja (lógica) de Sector

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** dar de baja (eliminar lógicamente) un sector específico en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID del sector que se desea dar de baja en la URL.

Tipo de petición: DELETE

URL: http://localhost:8080/api/sectores/{id} (reemplaza {id} con el ID del sector que deseas dar de baja)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID del sector que se desea dar de baja en la URL.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/sectores/{id}', {
      method: 'DELETE',
```

```
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      }
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que el sector se ha eliminado con éxito.

---

13) Creación de equipamiento/maquinaria en Sector

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** agregar equipamiento o maquinaria a un sector específico en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID del sector al que se desea agregar el equipamiento en la URL. Además, se debe proporcionar el nombre del equipamiento en el cuerpo de la solicitud.

Tipo de petición: POST

URL: http://localhost:8080/api/sectores/{id} (reemplaza {id} con el ID del sector al que deseas agregar el equipamiento)

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID del sector al que se desea agregar el equipamiento en la URL.
- Se debe proporcionar el **nombre** del equipamiento en el cuerpo de la solicitud en formato JSON.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: "Nombre del Equipamiento",
}
fetch('http://localhost:8080/api/sectores/{id}', {
      method: 'POST',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

equipamiento: información sobre el equipamiento o maquinaria creado, incluyendo su nombre y otros detalles.

14) Actualización de Equipamiento/Maquinaria

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** actualizar el nombre o el sector al que pertenece un equipamiento o maquinaria en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID del equipamiento que se desea actualizar en la URL. Además, se debe proporcionar el nombre y/o el ID del nuevo sector en el cuerpo de la solicitud.

Tipo de petición: PUT

URL: http://localhost:8080/api/maquinaria/{id} (reemplaza {id} con el ID del equipamiento que deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID del equipamiento que se desea actualizar en la URL.
- Se pueden proporcionar los siguientes campos en el cuerpo de la solicitud en formato JSON:

  nombre (opcional): El nuevo nombre del equipamiento.

  id_sector (opcional): El ID del nuevo sector al que pertenecerá el equipamiento.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: "Nuevo Nombre del Equipamiento",
      id_sector: 123
}
fetch('http://localhost:8080/api/maquinaria/{id}',{
      method: 'PUT',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que el equipamiento se ha actualizado con éxito.

equipamiento: Información actualizada del equipamiento, incluyendo su nombre y el nuevo sector al que pertenece (si se proporcionó).

---

15) Baja (Lógica) de Equipamiento/Maquinaria

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** dar de baja (lógica) un equipamiento o maquinaria en la base de datos. La baja lógica implica cambiar el estado del equipamiento a "inactivo" (false) sin eliminarlo físicamente de la base de datos. Para

realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID del equipamiento que se desea dar de baja en la URL.

Tipo de petición: DELETE

URL: http://localhost:8080/api/maquinaria/{id} (reemplaza {id} con el ID del equipamiento que deseas dar de baja)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID del equipamiento que se desea dar de baja en la URL.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/maquinaria/{id}',{
      method: 'DELETE',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que el equipamiento se ha dado de baja con éxito.

---

16) Crear Alarma de Mantenimiento para un Equipamiento

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** crear una alarma de mantenimiento para un equipamiento específico en la base de datos. Una alarma de mantenimiento se utiliza para programar una tarea de mantenimiento en un equipamiento en una fecha específica. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID del equipamiento al que se asociará la alarma de mantenimiento en la URL. Además, se debe proporcionar la fecha de la alarma en el cuerpo de la solicitud de manera obligatoria. Por otro lado, la descripción de solicitud, descripción de tarea y prioridad pueden ser añadidas en el cuerpo pero no son datos obligatorios.

Tipo de petición: POST

URL: http://localhost:8080/api/maquinaria/{id} (reemplaza {id} con el ID del equipamiento al que deseas asociar la alarma)

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID del equipamiento al que se asociará la alarma de mantenimiento en la URL.
- Se debe proporcionar la fecha en el cuerpo de la solicitud. También es posible asignar la descripción de la solicitud, descripción de la tarea y prioridad. Atributos del cuerpo de la petición:

  fecha: La fecha de la alarma de mantenimiento en el formato 'AAAA-MM-DD' (por ejemplo, '2023-10-15').

  descripcion: la descripción de la tarea.

  desc_soli: descripción de la solicitud.

  prioridad: prioridad de la futura tarea.

  id_responsable: ID del responsable asignado para realizar la tarea.

FetchAPI ejemplo de uso:

```
const body = {
      fecha: '2023-10-15', // Reemplaza con la fecha
deseada
      desc_soli: 'Se solicitó mantenimiento a la máquina
4.',
      descripcion: 'Revisar correas y engranajes.',
      id_responsable: 1
};
fetch('http://localhost:8080/api/maquinaria/{id}', {
      method: 'POST',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
```

```
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created.
- JSON de respuesta:

msg: Alarma creada correctamente.

alarma: objeto JSON con detalles de la alarma creada.

---

17) Actualizar Alarma de Mantenimiento

Este endpoint permite al **Jefe de Mantenimiento** que creó la alarma o a un **Gerente Operativo** actualizar la fecha, la descripción, la prioridad y el responsable de la misma. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID de la alarma de mantenimiento que se desea actualizar en la URL. Además, se debe proporcionar por lo menos alguna de estas opciones a actualizar en el cuerpo de la petición: la nueva fecha, la nueva descripción, la prioridad o el nuevo responsable.

Tipo de petición: PUT

URL: http://localhost:8080/api/alarmas/{id} (reemplaza {id} con el ID de la alarma de mantenimiento que deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID de la alarma de mantenimiento que se desea actualizar en la URL.
- Se debe proporcionar la nueva fecha, la nueva descripción, la prioridad, el nuevo responsable, de la alarma de mantenimiento en el cuerpo de la solicitud. Atributos del cuerpo de la petición:

  fecha: La nueva fecha de la alarma de mantenimiento en el formato 'AAAA-MM-DD' (por ejemplo, '2023-11-01').

  descripcion: La descripción de la tarea que se debe realizar (dicha alarma actúa como tarea a futuro, así que actúa como descripción de una tarea).

  prioridad: la prioridad que va a tener esta alarma cuando se vuelva tarea.

  id_responsable: el ID del nuevo responsable asignado para realizar la tarea.

FetchAPI ejemplo de uso:

```
const body = {
      fecha: '2023-11-01', // Nueva fecha (reemplaza con
la nueva fecha)
      descripcion: 'Mantenimiento preventivo a maquina
3',
      id_responsable: 2
};
fetch('http://localhost:8080/api/alarmas/{id}', {
      method: 'PUT',
      headers: {
```

```
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que la alarma se actualizó con éxito.

alarma: información actualizada de la alarma.

---

18) Baja (Física) de Alarma

Este endpoint permite al **Jefe de Mantenimiento** que creó la alarma o a un **Gerente Operativo** quitarla de la base de datos (el eliminar una alarma borra todo registro tanto de tarea como solicitud creada físicamente). Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y especificar el ID de la alarma de mantenimiento que se desea eliminar en la URL.

Tipo de petición: DELETE

URL: http://localhost:8080/api/alarmas/{id} (reemplaza {id} con el ID de la alarma de mantenimiento que deseas eliminar)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID de la alarma de mantenimiento que se desea eliminar en la URL.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/alarmas/{id}',{
      method: 'DELETE',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que la alarma se ha dado de baja con éxito.

---

19) Creación de Responsable de tarea/Empleado

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** crear un nuevo empleado o responsable de tarea en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y los datos del empleado, incluyendo su nombre, correo electrónico y teléfono.

Tipo de petición: POST

URL: http://localhost:8080/api/empleados

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se deben proporcionar los siguientes datos en el cuerpo de la solicitud en formato JSON:

  nombre: El nombre del empleado (cadena de texto, entre 2 y 46 caracteres).

  telefono: El teléfono del empleado (debe ser un número válido en formato string y único en el sistema).

  email: El correo electrónico del empleado (debe ser un correo electrónico válido y único en el sistema).

FetchAPI ejemplo de uso:

```
const body = {
      nombre: "Nombre del Empleado",
      email: "correo@ejemplo.com",
```

```
      telefono: "3434444344"
}
fetch('http://localhost:8080/api/empleados', {
      method: 'POST',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

empleado: los datos del empleado creado, excluyendo su estado.

---

20) Actualización del Responsable/Empleado

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** actualizar la información de un empleado o responsable de tarea en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud, el ID del empleado que se desea actualizar y los datos actualizados, como el nombre, el correo electrónico o el teléfono del empleado.

Tipo de petición: PUT

URL: http://localhost:8080/api/empleados/{id} (reemplaza {id} con el ID del empleado que deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se deben proporcionar los datos a actualizar en el cuerpo de la solicitud en formato JSON. Puedes actualizar el nombre, el correo electrónico, el teléfono o todos a la vez.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: "Nuevo nombre del Empleado",
      email: "otroCorreo@ejemplo.com"
}
fetch('http://localhost:8080/api/empleados/3', {
      method: 'PUT',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que el empleado se ha actualizado con éxito.

empleado: objeto JSON con los datos del empleado actualizado, excluyendo su estado.

---

21) Baja (Lógica) de Responsable/Empleado

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** realizar la baja lógica de un empleado o responsable de tarea en la base de datos. La baja lógica implica cambiar el estado del empleado de "activo" a "inactivo" (true o false) sin eliminar físicamente su registro. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y el ID del empleado que se desea dar de baja lógicamente.

Tipo de petición: DELETE

URL: http://localhost:8080/api/empleados/{id} (reemplaza {id} con el ID del empleado que deseas dar de baja)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- Se debe especificar el ID del responsable/empleado de que se desea eliminar en la URL.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/empleados/{id}',{
      method: 'DELETE',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que el empleado se ha dado de baja con éxito.

---

22) Creación de un Depósito

Este endpoint permite a un **Gerente** crear un nuevo depósito en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud y los datos del depósito que se desea crear.

Tipo de petición: POST

URL: http://localhost:8080/api/depositos

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Gerente Operativo válido.
- Se deben proporcionar los siguientes datos en el cuerpo de la solicitud en formato JSON:

  nombre: El nombre del depósito (cadena de texto, entre 2 y 46 caracteres).

FetchAPI ejemplo de uso:

```
const body = {
      nombre: "Nombre del Nuevo Depósito"
}

fetch('http://localhost:8080/api/depositos', {
      method: 'POST',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

deposito: JSON que incluye los detalles del depósito recién creado, incluido su nombre.

23) Actualización de un Depósito

Este endpoint permite a un **Gerente** actualizar la información de un depósito existente en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud, el ID del depósito que se desea actualizar y los nuevos datos para el depósito.

Tipo de petición: PUT

URL: http://localhost:8080/api/depositos/{id} (reemplaza {id} con el ID del depósito que deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Gerente Operativo válido.
- Se deben proporcionar los datos a actualizar en el cuerpo de la solicitud en formato JSON. Puedes actualizar el nombre.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: "Nuevo nombre del Depósito"
}
fetch('http://localhost:8080/api/depositos/3', {
      method: 'PUT',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
```

```
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que el depósito se actualizó correctamente.

deposito: objeto con los datos del depósito actualizado.

---

24) Baja (Lógica) de un depósito

Este endpoint permite a un **Gerente Operativo** eliminar (lógicamente) un depósito. La eliminación se realiza mediante una solicitud DELETE que incluye el ID del depósito.

Tipo de petición: DELETE

URL: http://localhost:8080/api/depositos/{id} (reemplaza {id} con el ID del depósito que deseas dar de baja)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada, donde {id} es el ID del depósito que se desea eliminar.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Gerente Operativo válido.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/depositos/{id}',{
      method: 'DELETE',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que el depósito se ha eliminado con éxito.

Depósito: información sobre el depósito eliminado.

---

25) Agregar Inventario a un Depósito

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** agregar un nuevo artículo al inventario de un depósito existente en la base de datos. Para realizar esta acción, es necesario proporcionar un token de autenticación válido en el encabezado de la solicitud, el ID del depósito al que se desea agregar el inventario y los datos del nuevo artículo.

Tipo de petición: POST

URL: http://localhost:8080/api/depositos/{id}

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Gerente Operativo válido.
- Se deben proporcionar los siguientes datos en el cuerpo de la solicitud en formato JSON:

  nombre: el nombre del artículo (cadena de texto, entre 2 y 46 caracteres).

  stock: la cantidad en stock del artículo.

FetchAPI ejemplo de uso:

```
const body = {
      nombre: 'Nuevo Artículo',
      stock: 50, // Reemplaza con la cantidad deseada en
stock
};

fetch('http://localhost:8080/api/depositos/1', {
      method: 'POST',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

inventario: objeto que incluye los detalles del artículo recién creado en el inventario.

26) Actualización de un artículo en inventario

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** actualizar un artículo específico en el inventario de un depósito. La actualización se realiza mediante una solicitud PUT que incluye el ID del artículo a actualizar y los datos a modificar (nombre o cantidad en stock).

Tipo de petición: PUT

URL: http://localhost:8080/api/inventario/{id} (reemplaza {id} con el ID del artículo que deseas actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.
- En el cuerpo de la solicitud en formato JSON, se pueden incluir los datos del artículo a actualizar, que pueden incluir el nombre y la cantidad en stock.
- Es posible actualizar solo el nombre, solo el stock o ambos.

FetchAPI ejemplo de uso:

```
const body = {
```

```
      nombre: "Nuevo nombre del Empleado", // Reemplaza
con el nuevo nombre o deja fuera esta propiedad si no
deseas cambiar el nombre
      stock: 50, // Reemplaza con la nueva cantidad en
stock o deja fuera esta propiedad si no deseas cambiar el
stock
}
fetch('http://localhost:8080/api/inventario/3', {
      method: 'PUT',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
      body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que el artículo se ha actualizado con éxito.

inventario: objeto que contiene los detalles del artículo actualizado en el inventario.

---

27) Baja (Lógica) de Artículo del Inventario

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** eliminar (lógicamente) un artículo específico del inventario de un depósito. La eliminación se realiza mediante una solicitud DELETE que incluye el ID del artículo a eliminar.

Tipo de petición: DELETE

URL: http://localhost:8080/api/inventario/{id} (reemplaza {id} con el ID del artículo de inventario que deseas dar de baja)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada, donde {id} es el ID del artículo en el inventario que se desea eliminar.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/inventario/{id}',{
      method: 'DELETE',
      headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que el artículo del inventario se ha eliminado con éxito.

28) Crear Solicitud de Tarea

Este endpoint permite a un **Encargado de Área** o **Gerente Operativo** crear una solicitud de tarea. La solicitud incluye una descripción y el ID del equipamiento relacionado (son obligatorios).

Tipo de petición: POST

URL: http://localhost:8080/api/solicitudes

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Encargado de Área o Gerente Operativo válido.
- El cuerpo de la solicitud debe incluir los siguientes campos:

  descripcion: Descripción detallada de la tarea (cadena de texto).

  id_equipamiento: ID del equipamiento relacionado con la tarea (número entero).

FetchAPI ejemplo de uso:

```
const body = {
      descripcion: 'Realizar mantenimiento preventivo en la maquinaria principal',
      id_equipamiento: 123 // Reemplaza con el ID del equipamiento relacionado
};

fetch('http://localhost:8080/api/solicitudes', {
  method: 'POST',
```

```
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
  body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

solicitud: objeto JSON que contiene la información de la solicitud recién creada.

---

29) Rechazar Solicitud

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** rechazar una solicitud de tarea. Se debe proporcionar el ID de la solicitud que se va a rechazar.

Tipo de petición: DELETE

URL: http://localhost:8080/api/solicitudes/:id (reemplaza {id} con el ID de la solicitud que se va a rechazar)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada con el ID de la solicitud en la ruta.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Jefe de Mantenimiento o Gerente Operativo válido.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/solicitudes/4', {
  method: 'DELETE',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que la solicitud se rechazó correctamente.

---

30) Cancelar Solicitud

Este endpoint permite a un **Encargado de Área** o **Gerente Operativo** cancelar una solicitud de tarea realizada que aún no ha sido aprobada. Se debe proporcionar el ID de la solicitud que se va a cancelar, por otro lado, solo puede cancelar la solicitud aquel EA que ha realizado esa solicitud o también puede hacerlo un GO.

Tipo de petición: PATCH

URL: http://localhost:8080/api/solicitudes/:id/cancel (reemplaza {id} con el ID de la solicitud que se desea cancelar)

Requisitos:

- Se debe realizar una solicitud PATCH a la URL especificada con el ID de la solicitud en la ruta.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar al Encargado de Área o Gerente Operativo válido.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/solicitudes/4/cancel', {
  method: 'PATCH',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que la solicitud se canceló correctamente.

---

31) Crear Tarea (aceptar solicitud)

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** crear una tarea aceptando una solicitud. Se debe proporcionar el ID de la solicitud que se va a aceptar en la URL. Se pueden proporcionar además datos de la tarea en el cuerpo de la solicitud, ejemplo, el responsable de realizarla, la prioridad y la descripción, pero estos datos no son obligatorios.

Tipo de petición: POST

URL: http://localhost:8080/api/solicitudes/{id} (reemplaza {id} con el ID de la solicitud que se va a aceptar en la URL y los datos de la tarea en el cuerpo de la solicitud)

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada con el ID de la tarea en la ruta.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Jefe de Mantenimiento válido o Gerente Operativo válido.
- El cuerpo de la solicitud puede incluir los siguientes campos:

  descripcion: Descripción detallada de la tarea.

  prioridad: prioridad de la tarea: 1 (alta), 2 (media), 3 (baja)

  id_responsable: ID del responsable asignado para realizar la tarea.

FetchAPI ejemplo de uso:

```
const body = {
```

```
      "descripcion": "Descripción de la tarea",
      "prioridad": 1,  // 1 (alta), 2 (media), 3 (baja)
      "id_responsable": 123   // Reemplaza con el ID del
empleado responsable asignado a la tarea
}

fetch('http://localhost:8080/api/solicitudes/4', {
  method: 'POST',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
  body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

tarea: objeto JSON que contiene la información de la tarea creada.

---

32) Cargar Tarea

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** cargar directamente una tarea. Se puede proporcionar los datos de la tarea y de la solicitud en el cuerpo de la petición.

Tipo de petición: POST

URL: http://localhost:8080/api/tareas/

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada con el ID de la tarea en la ruta.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Jefe de Mantenimiento válido o Gerente Operativo válido.
- El cuerpo de la solicitud debe incluir los siguientes campos:

  descripcion: observación de la tarea, no es obligatorio la descripción de la tarea, ya que, se puede proporcionar una descripción de solicitud propia (en este endpoint la solicitud se crea igual, siendo el usuario que realiza la petición el solicitante y el juez al mismo tiempo).

  desc_soli: descripción de la solicitud. Ej: Arreglar silo 1.

  prioridad: prioridad de la tarea: 1 (alta), 2 (media), 3 (baja)

  id_responsable: ID del responsable asignado para realizar la tarea.

FetchAPI ejemplo de uso:

```
const body = {
      "desc_soli": "Descripción de la solicitud no es obligatoria pero se puede proporcionar una",
      "descripcion": "Descripción de la tarea no es obligatoria",
      "prioridad": 1,  // 1 (alta), 2 (media), 3 (baja), no es obligatoria.
      "estado": "en curso", //en curso o finalizada (no es necesario mandarlo cuando es una tarea en curso)
      "id_responsable": 123  // Reemplaza con el ID del empleado responsable asignado a la tarea
}

fetch('http://localhost:8080/api/tareas', {
  method: 'POST',
```

```
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
  body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 Created
- JSON de respuesta:

msg: un mensaje que la tarea se creó correctamente

tarea: objeto JSON que contiene la información de la tarea creada.

---

33) Actualizar Tarea

Este endpoint permite al **Jefe de Mantenimiento** que creó la tarea o a un **Gerente Operativo** actualizarla. Se debe proporcionar el ID de la tarea que se va a actualizar en la URL y los datos actualizados en el cuerpo de la solicitud.

Tipo de petición: PUT

URL: http://localhost:8080/api/tareas/{id} (reemplaza {id} con el ID de la tarea que se va a actualizar)

Requisitos:

- Se debe realizar una solicitud PUT a la URL especificada con el ID de la tarea en la ruta.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Jefe de Mantenimiento válido o Gerente Operativo válido.
- El cuerpo de la solicitud debe incluir cualquiera de los siguientes campos, puede ser uno, dos o todos a la vez:

  descripcion: Descripción detallada de la tarea.

  prioridad: prioridad de la tarea: 1 (alta), 2 (media), 3 (baja)

  id_responsable: ID del responsable a la tarea con la tabla.

FetchAPI ejemplo de uso:

```
const body = {
      "descripcion": "Descripción de la tarea",
      "prioridad": 1,  // 1 (alta), 2 (media), 3 (baja)
      "id_responsable": 123   // Reemplaza con el ID del
empleado responsable asignado a la tarea
}

fetch('http://localhost:8080/api/tareas/4', {
  method: 'PUT',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
  body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje de éxito en la actualización de la tarea.

tarea: objeto JSON que contiene la información actualizada de la tarea.

---

34) Finalizar Tarea

Este endpoint permite al **Jefe de Mantenimiento** que creó la tarea o a un **Gerente Operativo** finalizarla. Se debe proporcionar el ID de la tarea que se va a finalizar en la URL.

Tipo de petición: PATCH

URL: http://localhost:8080/api/tareas/{id} (reemplaza {id} con el ID de la tarea que se va a finalizar)

Requisitos:

- Se debe realizar una solicitud PATCH a la URL especificada con el ID de la tarea en la ruta.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Jefe de Mantenimiento válido o Gerente Operativo válido.
- **IMPORTANTE:** para finalizar la tarea es necesario que la misma tenga asignado un responsable, caso contrario no se podrá modificar el estado a finalizado.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/tareas/4', {
  method: 'PATCH',
```

```
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje indicando que la tarea se finalizó correctamente.

---

35) Agregar Inventario a Tarea

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** agregar inventario a una tarea existente. Se debe proporcionar el ID de la tarea en la URL y la información del inventario a agregar en el cuerpo de la solicitud.

Tipo de petición: POST

URL: http://localhost:8080/api/tareas/{id} (reemplaza {id} con el ID de la tarea a la que se va a agregar inventario)

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada con el ID de la tarea en la ruta.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Jefe de Mantenimiento válido o Gerente Operativo válido.
- Se deben proporcionar los detalles del inventario:

  id_inventario: el id del inventario

  cantidad: la cantidad utilizada de ese inventario en dicha tarea

FetchAPI ejemplo de uso:

```
const body = {
      "id_inventario": 123,
      "cantidad": 123
}
fetch('http://localhost:8080/api/tareas/4', {
  method: 'POST',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
  body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 OK
- JSON de respuesta:

msg: mensaje que indica que el Inventario se agregó correctamente

inventarioTarea: objeto que contiene la información del inventario en tarea.

---

36) Eliminar Inventario a Tarea

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** quitarle inventario a una tarea existente. Se debe proporcionar el ID de la tarea en la URL y la información del inventario a quitar en el cuerpo de la solicitud.

Tipo de petición: DELETE

URL: http://localhost:8080/api/tareas/{id} (reemplaza {id} con el ID de la tarea a la que se desea quitar el inventario)

Requisitos:

- Se debe realizar una solicitud DELETE a la URL especificada con el ID de la tarea en la ruta.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Jefe de Mantenimiento válido o Gerente Operativo válido.
- Se deben proporcionar los detalles del inventario a eliminar:

  id_inventario: el id del inventario.

FetchAPI ejemplo de uso:

```
const body = {
      "id_inventario": 123,
}
fetch('http://localhost:8080/api/tareas/4', {
  method: 'DELETE',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
  body: JSON.stringify(body)
})
```

```
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que el Inventario se quitó correctamente de la tarea

---

37) Crear BackUp de manera manual

Este endpoint permite a un **Gerente Operativo** realizar un backup de la base de datos en el momento que se llama a este endpoint. Tenga en cuenta que para realizarlo debe de mandar la contraseña de su usuario en el cuerpo de la petición, con el objetivo de validar su identidad.

Tipo de petición: POST

URL: http://localhost:8080/api/backup/

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Gerente Operativo válido.
- Se deben proporcionar la contraseña del usuario que quiere realizar el backup:

  password: contraseña del usuario

FetchAPI ejemplo de uso:

```
const body = {
      "password": "testadmin123"
}
fetch('http://localhost:8080/api/backup', {
  method: 'POST',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
  body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 201 OK
- JSON de respuesta:

msg: mensaje que indica que el backup se ha creado con éxito

---

38) Configurar BackUp automático

Este endpoint permite a un **Gerente Operativo** activar y setear la frecuencia de un backup automático de la base de datos. En la URL de la petición se debe especificar la frecuencia que tendrá el backup (d es diario, w es semanal, m es mensual y n es ninguna frecuencia, es decir, se detiene el backup automático). Tenga en cuenta que debe de mandar

la contraseña de su usuario en el cuerpo de la petición, con el objetivo de validar su identidad.

Tipo de petición: POST

URL: http://localhost:8080/api/backup/{frecuencia} (reemplaza {frecuencia} con la inicial de la frecuencia del backup **d** es diario, **w** semanal, **m** mensual y **n** ninguna frecuencia, es decir, se detiene el backup automático)

Requisitos:

- Se debe realizar una solicitud POST a la URL especificada e incluir la frecuencia.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Gerente Operativo válido.

FetchAPI ejemplo de uso:

```
const body = {
      "password": "testadmin123"
}
fetch('http://localhost:8080/api/backup/d', {
  method: 'POST',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
  body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que el backup automático se ha configurado correctamente.

---

39) Realizar una restauración de datos (restore)

Este endpoint permite a un **Gerente Operativo** realizar una restauración de datos a la base de datos, volviendo a un punto de restauración en base a los backups disponibles. Se debe proporcionar el ID del backup (punto de restauración) al que se va a recuperar los datos en el cuerpo de la petición, además tenga en cuenta que debe de mandar la contraseña de su usuario, con el objetivo de validar su identidad.

Tipo de petición: POST

URL: http://localhost:8080/api/restore

Requisitos:

- Se debe realizar una solicitud POST a la URL.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Gerente Operativo válido.
- Se deben proporcionar los siguientes datos en el cuerpo de la petición:

id: el ID del backup al cual se utilizará como punto de restauración para recuperar los datos en ese punto.

password: contraseña del usuario.

FetchAPI ejemplo de uso:

```
const body = {
      "id": "BACKUP_2024-07-16_17-56-28.271692",
      "password": "testadmin123"
}
fetch('http://localhost:8080/api/restore', {
  method: 'POST',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
  body: JSON.stringify(body)
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que se ha restaurado completamente la base de datos.

---

40) Generar Informe y mandarlo por correo.

Este endpoint permite a un **Gerente Operativo** generar un informe de todas las tareas y enviarlo directamente al correo del usuario. Para utilizarlo, se debe proporcionar un token de autenticación válido en el encabezado de la solicitud.

Este endpoint es una petición POST que no requiere un cuerpo con datos, solo el token de autenticación. Además, se pueden usar parámetros de consulta para filtrar las tareas por estado, intervalo, responsable, entre otras opciones, a fin de generar informes más específicos. La siguiente tabla muestra los parámetros de consulta que se puede utilizar:

Parámetro	Valores admitidos			Descripción
estado	**C**	**F**		Filtra las tareas por su estado (en curso o finalizada)
intervalo	**T**	**W**	**M**	Filtra las tareas por un rango específico de fechas. T: Genera un informe con las tareas del día actual. W: Incluye las tareas de la última semana. M: Muestra las tareas de los últimos treinta días.
responsable	ID			Filtra las tareas por el ID de un responsable.
área	ID			Filtra las tareas por ID de un área.
equipamiento	ID			Filtra las tareas por el ID de un equipamiento específico. **Si se ha añadido el área en los parámetros, este ID de equipamiento debe pertenecer a esa área.**
gráfico	**Y**			El parámetro permite generar una página adicional en el informe que incluye un gráfico de barras. Este muestra la cantidad de tareas por cada área.

Tipo de petición: POST

URL: http://localhost:8080/api/reports

Requisitos:

- Se debe realizar una solicitud POST a la URL.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Gerente Operativo válido.

FetchAPI ejemplo de uso:

```
fetch('http://localhost:8080/api/reports', {
  method: 'POST',
  headers: {
      'Content-Type': 'application/json',
      'calp-token': 'TOKEN'
      },
})
.then(response => response.json())
.then(json => console.log(json))
.catch(err => console.error(err));
```

Respuesta exitosa:

- Código de estado HTTP 200 OK
- JSON de respuesta:

msg: mensaje que indica que el reporte fue generado y enviado con éxito.

Obtención de datos (lectura)

1) Obtener todos usuarios

Este endpoint permite a un **Gerente** obtener información sobre todos los usuarios registrados en el sistema.

Tipo de petición: GET

URL: http://localhost:8080/api/usuarios/

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Gerente válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la lista de usuarios, cada uno con su id, nombre, apellido, email y rol (EA = encargado de área, GO = Gerente Operativo, JM = Jefe de Mantenimiento).

---

2) Obtener usuario por correo

Este endpoint permite a un **Gerente** obtener un usuario registrado en el sistema filtrado por su correo electrónico.

Tipo de petición: GET

URL: http://localhost:8080/api/usuarios?correo=abc@correo.com

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Gerente válido.
- Se debe incluir en la query string o cadena de consulta, el parámetro correo con el valor correspondiente al correo que se desea consultar.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el usuario filtrado por correo, con su id, nombre, apellido, email y rol (EA = encargado de área, GO = Gerente Operativo, JM = Jefe de Mantenimiento).
- En caso de no encontrar ningún usuario el resultado será un objeto vacío

---

3) Obtener usuario por ID

Este endpoint permite a un **Gerente** obtener un usuario registrado en el sistema filtrado por su ID.

Tipo de petición: GET

URL: http://localhost:8080/api/usuarios/{id} (reemplaza {id} con el ID de un usuario válido en el sistema)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Gerente válido.

- Se debe incluir el ID en la URL.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el usuario filtrado por su ID, con su respectivo id, nombre, apellido, email y rol (EA = encargado de área, GO = Gerente Operativo, JM = Jefe de Mantenimiento).
- En caso de no encontrar ningún usuario el resultado será un objeto vacío

---

4) Obtener por rol

Este endpoint permite a un **Gerente** obtener todos los usuarios con el rol especificado en los parámetros, que se encuentren registrados en el sistema.

Tipo de petición: GET

URL: http://localhost:8080/api/usuarios?rol={JM, GO, EA}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un Gerente válido.
- Se debe incluir en la query string o cadena de consulta, el parámetro rol con el valor correspondiente al rol de los usuarios que se desea extraer.

Resultado:

- Código de estado HTTP 200 OK

- Objeto JSON que contiene la lista de usuarios pertenecientes al rol especificado, cada uno con su id, nombre, apellido, email y rol (EA = encargado de área, GO = Gerente Operativo, JM = Jefe de Mantenimiento).
- En caso de colocar un rol no válido, el endpoint traerá todos los usuarios del sistema.

---

5) Obtener Empleados

Este endpoint permite a un **Jefe de Mantenimiento o Gerente Operativo** obtener una lista con los usuarios responsables de las tareas o empleados correspondientes al taller de mantenimiento registrados en el sistema.

Tipo de petición: GET

URL: http://localhost:8080/api/empleados/

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la lista de empleados, cada uno con su id, nombre, email y teléfono.

---

6) Obtener Empleados por correo

Este endpoint permite a un **Jefe de Mantenimiento o Gerente Operativo** obtener un usuario responsable de las tareas o empleado correspondiente al taller de mantenimiento registrado en el sistema filtrado por su correo electrónico.

Tipo de petición: GET

URL: http://localhost:8080/api/empleados?correo=abc@correo.com

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir en la query string o cadena de consulta, el parámetro “correo” con el valor correspondiente al correo que se desea consultar.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el usuario filtrado por su correo, con su respectivo id, nombre, email y teléfono.
- En caso de no encontrar ningún usuario el resultado será un objeto vacío

---

7) Obtener Empleados por teléfono

Este endpoint permite a un **Jefe de Mantenimiento o Gerente Operativo** obtener un usuario responsable de las tareas o empleado

correspondiente al taller de mantenimiento registrado en el sistema filtrado por su teléfono.

Tipo de petición: GET

URL: http://localhost:8080/api/empleados?telefono=3436455555

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir en la query string o cadena de consulta, el parámetro “teléfono” con el valor correspondiente al teléfono que se desea consultar.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el usuario filtrado por su teléfono, con su respectivo id, nombre, email y teléfono.
- En caso de no encontrar ningún usuario el resultado será un objeto vacío

---

8) Obtener Empleados por ID

Este endpoint permite a un **Jefe de Mantenimiento o Gerente Operativo** obtener un usuario responsable de las tareas o empleado correspondiente al taller de mantenimiento registrado en el sistema filtrado por su ID.

Tipo de petición: GET

URL: http://localhost:8080/api/empleados/{id} (reemplaza {id} por el id de un empleado válido)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir en la URL, el ID del empleado o responsable.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el usuario filtrado por su ID, con su respectivo id, nombre, email y teléfono.

---

9) Obtener Area de un Encargado

Este endpoint permite a cualquier usuario registrado en el sistema independientemente de su rol, obtener el área de un Encargado de Área.

Tipo de petición: GET

URL: http://localhost:8080/api/areas/usr/{id} (reemplaza {id} con el ID del encargado de área cuyo área deseas saber)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar que es un usuario válido logueado en el sistema.
- Se debe incluir el ID del Encargado de Área en la URL.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el área del Encargado de Área con su id y nombre.
- En caso de no encontrar ningún usuario el resultado será un objeto vacío

---

10) Obtener Áreas

Este endpoint permite a un **Jefe de Mantenimiento o Gerente Operativo** obtener un listado con todas las áreas cargadas en el sistema.

Tipo de petición: GET

URL: http://localhost:8080/api/areas

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado de áreas.

---

11) Obtener Áreas con sus encargados

Este endpoint permite a un **Jefe de Mantenimiento o Gerente Operativo** obtener un listado con todas las áreas cargadas en el sistema, mostrando a su vez el nombre y el apellido del encargado de cada área.

Tipo de petición: GET

URL: http://localhost:8080/api/areas?encargados=true

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado de áreas con datos del área y el nombre y apellido del encargado.

---

12) Obtener Área por ID

Este endpoint permite a un **Jefe de Mantenimiento o Gerente Operativo** obtener un área filtrada por su ID.

Tipo de petición: GET

URL: http://localhost:8080/api/areas/{id} (reemplaza {id} con el ID del área)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID del área en la URL.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el área filtrada por su ID.
- En caso de ingresar un ID no válido, se devolverá un mensaje de error.

---

13) Obtener Sectores de un Área

Este endpoint permite a cualquier usuario registrado en el sistema, obtener un listado de todos los sectores de un área específica.

Tipo de petición: GET

URL: http://localhost:8080/api/areas/{id}/sectores

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar que es un usuario válido logueado en el sistema.
- Se debe incluir el ID del área.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado de sectores de un área.
- En caso de proporcionar un ID de área no válido, se devolverá un mensaje de error.
- En caso de no poseer sectores el resultado será un arreglo vacío.

---

14) Obtener todos los Sectores

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener un listado de todos los sectores.

Tipo de petición: GET

URL: http://localhost:8080/api/sectores

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado de sectores.
- En caso de no poseer sectores el resultado será un arreglo vacío.

---

15) Obtener Sector por ID

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener un sector filtrado por su ID.

Tipo de petición: GET

URL: http://localhost:8080/api/sectores/{id}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID del sector.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el sector filtrado.
- En caso de proporcionar un ID de sector no válido, se devolverá un mensaje de error.
- En caso de no poseer sectores el resultado será un objeto vacío.

---

16) Obtener todo el Equipamiento de un Sector

Este endpoint permite a cualquier usuario registrado en el sistema, obtener un listado de todas las maquinarias pertenecientes a un sector específico.

Tipo de petición: GET

URL: http://localhost:8080/api/sectores/{id}/maquinarias

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar que es un usuario válido logueado en el sistema.
- Se debe incluir el ID del sector.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado de todas las maquinarias de un sector.
- En caso de proporcionar un ID de sector no válido, se devolverá un mensaje de error.
- En caso de no haber maquinarias en dicho sector el resultado será un arreglo vacío.

---

17) Obtener todos los Equipamientos

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener un listado de todos los equipamientos.

Tipo de petición: GET

URL: http://localhost:8080/api/maquinaria

- Se debe realizar una solicitud GET a la URL especificada.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado de los equipamientos.
- En caso de no poseer sectores el resultado será un arreglo vacío.

---

18) Obtener Equipamiento por ID

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener un equipamiento por ID.

Tipo de petición: GET

URL: http://localhost:8080/api/maquinaria/{id}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de la maquinaria/equipamiento en la URL.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el equipamiento filtrado por ID.
- En caso de ingresar un ID no válido, se devolverá un mensaje de error.

19) Obtener Alarma de mant. de un Equipamiento

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo**, obtener la alarma de mantenimiento de una maquinaria en específico.

Tipo de petición: GET

URL: http://localhost:8080/api/maquinaria/{id}/alarmas

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID del equipamiento.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la alarma de mantenimiento de un equipamiento/maquinaria.
- En caso de proporcionar un ID de equipamiento no válido, se devolverá un mensaje de error.
- En caso de que el equipamiento no posea alarma el resultado será un objeto vacío.

20) Obtener todas las Alarmas de mantenimiento

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo**, obtener un listado con todas las alarmas.

Tipo de petición: GET

URL: http://localhost:8080/api/alarmas

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado de alarmas de mantenimiento.

---

21) Obtener todas las alarmas de mantenimiento ordenadas.

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo**, obtener un listado con todas las alarmas de manera ordenada.

Tipo de petición: GET

URL: http://localhost:8080/api/alarmas/order

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado de alarmas de mantenimiento.

---

22) Obtener Alarma por ID

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener la alarma filtrada por su ID.

Tipo de petición: GET

URL: http://localhost:8080/api/alarmas/{id}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de la alarma.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la alarma filtrada.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.

23) Obtener Alarmas por ID de juez

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener las alarmas que ha realizado un juez.

Tipo de petición: GET

URL: http://localhost:8080/api/alarmas/juez/{id} (reemplaza {id} por el id de juez)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID del juez.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene las alarmas que ha realizado el juez.

---

24) Obtener las alarmas de una maquinaria

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener las alarmas que contiene la maquinaria en cuestión.

Tipo de petición: GET

URL: http://localhost:8080/api/maquinaria/{id}/alarmas (reemplaza {id} por el id de la maquinaria)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de la maquinaria

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene las alarmas que tiene dicho id de maquinaria.

---

25) Obtener las alarmas de un juez en una maquinaria

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener las alarmas que ha hecho un juez en una determinada maquinaria.

Tipo de petición: GET

URL: http://localhost:8080/api/maquinaria/{id}/alarmas/juez/{id_juez} (reemplaza {id} por el id de la maquinaria y {id_juez} por el id del juez)

- Se debe realizar una solicitud GET a la URL especificada.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de la maquinaria y el ID del juez en la URL de la petición.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene las alarmas que ha hecho el juez en dicha maquinaria.

---

26) Obtener Depósito de Jefe Mantenimiento

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener un depósito filtrado por un ID correspondiente a un Jefe de Mantenimiento.

Tipo de petición: GET

URL: http://localhost:8080/api/depositos/usr/{id} (reemplaza {id} por el ID del Jefe de Mantenimiento)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID del Jefe de Mantenimiento .

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el depósito del Jefe de Mantenimiento.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.
- En caso de un ID no perteneciente a un Jefe de Mantenimiento el resultado será un objeto vacío.

---

27) Obtener todos los Depósitos

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener una lista completa de todos los depósitos.

Tipo de petición: GET

URL: http://localhost:8080/api/depositos/

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene una lista de todos los depósitos con la información del jefe de mantenimiento a cargo del mismo, si es que existiese, caso contrario, nombre y apellido vendrán en **null**.

28) Obtener Depósito por ID

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener un depósito filtrado por su ID.

Tipo de petición: GET

URL: http://localhost:8080/api/depositos/{id}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID del depósito.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene información del depósito.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.

29) Obtener inventario de un Depósito por ID

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener el inventario completo de un depósito filtrado por su ID.

Tipo de petición: GET

URL: http://localhost:8080/api/depositos/{id}/inventario

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID del depósito.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la lista de inventario de un depósito filtrado por su ID.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.
- En caso de que el depósito no posea inventario (vacío) entonces el resultado será un arreglo vacío.

---

30) Obtener el Inventario

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener el inventario completo.

Tipo de petición: GET

URL: http://localhost:8080/api/inventario

- Se debe realizar una solicitud GET a la URL especificada.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el inventario completo del taller de mantenimiento.
- En caso de que no haya inventario entonces el resultado será un objeto vacío.

---

31) Obtener articulo de Inventario por ID

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener el artículo filtrado por su ID.

Tipo de petición: GET

URL: http://localhost:8080/api/inventario/{id}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID del artículo del inventario.

Resultado:

- Código de estado HTTP 200 OK

- Objeto JSON que contiene información del artículo contenido en el inventario.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.

---

32) Obtener todas las Solicitudes

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener el listado completo de las solicitudes.

Tipo de petición: GET

URL: http://localhost:8080/api/solicitudes

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado completo de las solicitudes.
- En caso de que no haya solicitudes cargadas entonces el resultado será un objeto vacío.

---

33) Obtener todas las Solicitudes ordenadas

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener el listado completo de las solicitudes ordenadas por estado (`pendiente` primero) y por fecha más reciente.

Tipo de petición: GET

URL: http://localhost:8080/api/solicitudes/order

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado completo de las solicitudes.
- En caso de que no haya solicitudes cargadas entonces el resultado será un objeto vacío.

---

34) Obtener las Solicitudes por estado

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener el listado completo de las solicitudes filtradas por un estado en específico.

Tipo de petición: GET

URL: http://localhost:8080/api/solicitudes?estado={‘pendiente’, ‘rechazada’, ‘aceptada’}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir en la query string o cadena de consulta, el parámetro estado con el valor correspondiente al estado que se desea consultar.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado completo de las solicitudes con el estado filtrado.
- En caso de que no haya solicitudes en ese estado en específico, entonces el resultado será un objeto vacío.

---

35) Obtener las Solicitudes de un usuario

Este endpoint permite a un **Encargado de Área** o **Gerente Operativo** registrado en el sistema (esto es más que nada para que cada EA solo visualice sus solicitudes), obtener las solicitudes filtradas por un ID de usuario válido.

Tipo de petición: GET

URL: http://localhost:8080/api/solicitudes/usr/{id} (reemplaza {id} por el ID de un usuario)

Filtrar solicitudes de un usuario por estado:

URL: http://localhost:8080/api/solicitudes/usr/{id}?estado={'pendiente', 'rechazada', 'aceptada'} (reemplaza {id} por el ID de un usuario)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un EA o GO válido.
- Se debe incluir el ID de un usuario válido.
- En caso de filtrar las solicitudes por un estado en específico se debe incluir en la query string o cadena de consulta luego del ID, el parámetro estado con el valor correspondiente al estado que se desea consultar.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene las solicitudes realizadas por ese usuario.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.
- En caso de que un ID no contenga solicitudes, el resultado será un objeto vacío.

---

36) Obtener Solicitud por ID

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** o **Encargado de Area** registrado en el sistema, obtener una solicitud filtrada por su ID.

Tipo de petición: GET

URL: http://localhost:8080/api/solicitudes/{id}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO o EA válido.
- Se debe incluir el ID de la solicitud.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene información de la solicitud consultada.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.

---

37) Obtener la tarea de una Solicitud

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener la tarea de una solicitud en específico.

Tipo de petición: GET

URL: http://localhost:8080/api/solicitudes/{id}/tarea

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de una solicitud válida.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la información de la tarea de mantenimiento correspondiente a la solicitud realizada por un encargado de área.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.
- En caso de que la solicitud aún esté en estado ‘pendiente’ y no haya sido tratada aún, entonces el resultado será un objeto vacío.

---

38) Obtener todas las Tareas

Este endpoint permite a un **Gerente Operativo** registrado en el sistema, obtener el listado completo de las tareas de mantenimiento que estén ‘en curso’ o ‘finalizadas’. Se puede filtrar mediante la cadena de consulta por prioridad o estado.

Tipo de petición: GET

URL: http://localhost:8080/api/tareas

Filtrar por estado:

URL: http://localhost:8080/api/tareas?estado={‘en curso’ o 'finalizada'}

Filtrar por prioridad:

URL: http://localhost:8080/api/tareas?prioridad={1,2,3}

- Se debe realizar una solicitud GET a la URL especificada.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado completo de las tareas en un arreglo de objetos que son cada una de las tareas.
- En caso de que no haya tareas cargadas entonces el resultado será un arreglo vacío.
- En caso de filtrar las tareas por un estado o prioridad en específico se debe incluir en la query string o cadena de consulta, el parámetro estado o prioridad con el valor correspondiente al estado o prioridad que se desea consultar.

---

39) Obtener las Tareas de forma ordenada

Este endpoint permite a un **Gerente Operativo** registrado en el sistema, obtener las tareas de mantenimiento que estén 'en curso' y luego las 'finalizadas' ordenadas por la fecha (más reciente primero) la prioridad (1 la más alta aparecerán primero).

Tipo de petición: GET

URL: http://localhost:8080/api/tareas/order

- Se debe realizar una solicitud GET a la URL especificada.

- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el listado completo de las tareas.
- En caso de que no haya tareas cargadas entonces el resultado será un arreglo vacío.

---

40) Obtener Tarea por ID

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener una tarea de mantenimiento por un ID en específico.

Tipo de petición: GET

URL: http://localhost:8080/api/tareas/{id}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de la tarea.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene información de la tarea consultada.

- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.

---

41) Obtener Tareas por ID de maquinaria

Este endpoint permite a un **Gerente Operativo** registrado en el sistema, obtener el listado de tareas de mantenimiento que tiene una maquinaria, por medio de un ID de maquinaria en específico.

Tipo de petición: GET

URL: http://localhost:8080/api/tareas/maquinaria/{id}

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un GO válido.
- Se debe incluir el ID del equipamiento.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene un arreglo de tareas o un arreglo vacío si la maquinaria no se ha involucrado en ninguna tarea.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.

---

42) Obtener las Tareas de un JUEZ

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener todas las tareas correspondiente a un juez (usuario que rechaza las solicitudes o las acepta y las transforma en tareas). Este endpoint se recomienda utilizar más que nada, en las vistas correspondientes a los jefes de mantenimiento para que solo puedan observar sus propias tareas y no puedan finalizar u observar las tareas de otro jefe de mantenimiento o gerente operativo.

Tipo de petición: GET

URL: http://localhost:8080/api/tareas/juez/{id} (reemplaza {id} por un ID de un usuario válido)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de un usuario registrado en el sistema valido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la siguiente información de las tareas de mantenimiento correspondiente al juez, cada tarea tiene:

  “**usr_solicitante**”: el ID del usuario que realizó la solicitud.

  "id_solicitud": el ID de la solicitud a la que corresponde la tarea.

  "estado_solicitud": el estado en el que se encuentra la solicitud (siempre es ‘aceptada’)

"**fecha_soli_realizada**": la fecha en que se realizó la solicitud formato: AAAA-MM-DD,

"**desc_solicitud**": la descripción de la solicitud,

"**id_equipamiento**": el ID correspondiente al equipamiento (si es NULL significa que la tarea no proviene de un EA sino que fue cargada mediante un JM o GO).

"**id_tarea**": el ID de la tarea.

"**estado_tarea**": el estado en el que se encuentra la tarea,

"**fecha_tar_tratada**": la fecha en la que fue tratada, formato: AAAA-MM-DD.

"fecha_cumplimiento": fecha en la que se finalizó la tarea.

"**desc_tar**": la descripción de la tarea,

"**priori_tar**": la prioridad de la tarea (1= alta, 2=media, 3= baja),

"**responsable_tar**": el responsable asignado de realizar dicha tarea.

- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.
- En caso de que un usuario no haya aceptado solicitudes traduciéndose en tareas, entonces el resultado será un arreglo vacío.

---

43) Obtener las Tareas de un JUEZ ordenadas

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener todas las tareas correspondiente a un juez (usuario que rechaza las solicitudes o las acepta y las transforma en tareas). Las tareas se encuentran ordenadas apareciendo primero las que estén en curso, más recientes y con prioridad alta al mismo tiempo.

Este endpoint se recomienda utilizar más que nada, en las vistas correspondientes a los jefes de mantenimiento para que solo puedan observar sus propias tareas y no puedan finalizar u observar las tareas de otro jefe de mantenimiento o gerente operativo.

Tipo de petición: GET

URL: http://localhost:8080/api/tareas/juez/{id}/order (reemplaza {id} por un ID de un usuario válido)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de un usuario registrado en el sistema valido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la siguiente información de las tareas de mantenimiento correspondiente al juez, cada tarea tiene:

  “**usr_solicitante**”: el ID del usuario que realizó la solicitud.

  "id_solicitud": el ID de la solicitud a la que corresponde la tarea.

  "**estado_solicitud**": el estado en el que se encuentra la solicitud (siempre es ‘aceptada’)

  "**fecha_soli_realizada**": la fecha en que se realizó la solicitud formato: AAAA-MM-DD,

  "**desc_solicitud**": la descripción de la solicitud,

  "**id_equipamiento**": el ID correspondiente al equipamiento (si es NULL significa que la tarea no proviene de un EA sino que fue cargada mediante un JM o GO).

"**id_tarea**": el ID de la tarea.

"**estado_tarea**": el estado en el que se encuentra la tarea,

"**fecha_tar_tratada**": la fecha en la que fue tratada, formato: AAAA-MM-DD.

"fecha_cumplimiento": fecha en la que se finalizó la tarea.

"**desc_tar**": la descripción de la tarea,

"**priori_tar**": la prioridad de la tarea (1= alta, 2=media, 3= baja),

"**responsable_tar**": el responsable asignado de realizar dicha tarea.

- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.
- En caso de que un usuario no haya aceptado solicitudes traduciéndose en tareas, entonces el resultado será un arreglo vacío.

---

44) Obtener las Tareas de maquinaria de un JUEZ ordenadas

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema obtener las tareas correspondientes a una maquinaria específica. Las tareas deben haber sido aceptadas por un Juez (usuario encargado de aceptar o rechazar solicitudes y convertirlas en tareas o directamente crearlas). Las tareas se presentan en el siguiente orden: las que están en curso primero, seguidas por las más recientes y con mayor prioridad.

Este endpoint es especialmente útil para vistas donde se visualizan las maquinarias y las tareas asociadas a ellas. Sin embargo, es importante destacar que cada Jefe de Mantenimiento solo podrá ver las tareas de la

maquinaria que ha gestionado personalmente. No podrán finalizar ni observar las tareas asignadas a otros **Jefes de Mantenimiento** o **Gerentes Operativos**.

Tipo de petición: GET

URL: http://localhost:8080/api/tareas/juez/{id}/order/maq/{id_equipo}

(reemplaza {id} por un ID de un usuario válido y {id_equipo} por un ID de equipamiento sectorial válido)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de un usuario registrado en el sistema valido.
- Se debe incluir el ID de un equipamiento válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la siguiente información de las tareas de mantenimiento correspondiente al juez y la maquinaria asociada, cada tarea tiene:

  “**usr_solicitante**”: el ID del usuario que realizó la solicitud.

  "id_solicitud": el ID de la solicitud a la que corresponde la tarea.

  "**estado_solicitud**": el estado en el que se encuentra la solicitud (siempre es ‘aceptada’)

  "**fecha_soli_realizada**": la fecha en que se realizó la solicitud formato: AAAA-MM-DD,

  "**desc_solicitud**": la descripción de la solicitud,

"**id_equipamiento**": el ID correspondiente al equipamiento (si es NULL significa que la tarea no proviene de un EA sino que fue cargada mediante un JM o GO).

"**id_tarea**": el ID de la tarea.

"**estado_tarea**": el estado en el que se encuentra la tarea,

"**fecha_tar_tratada**": la fecha en la que fue tratada, formato: AAAA-MM-DD.

"fecha_cumplimiento": fecha en la que se finalizó la tarea.

"**desc_tar**": la descripción de la tarea,

"**priori_tar**": la prioridad de la tarea (1= alta, 2=media, 3= baja),

"**responsable_tar**": el responsable asignado de realizar dicha tarea.

- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.
- En caso de que un usuario no haya realizado tareas en esa maquinaria, entonces el resultado será un arreglo vacío.

---

45) Obtener el inventario de una tarea

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** registrado en el sistema, obtener todo el inventario utilizado en una tarea.

Tipo de petición: GET

URL: http://localhost:8080/api/tareas/{id}/inventario (reemplaza {id} por el ID de una tarea válida)

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un JM o GO válido.
- Se debe incluir el ID de una tarea válida.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene el inventario utilizado en la tarea, con su respectiva cantidad utilizada.
- En caso de proporcionar un ID no válido, se devolverá un mensaje de error.
- En caso de que un ID de tarea no contenga un inventario, el resultado será un arreglo vacío.

---

46) Obtener los backups realizados

Este endpoint permite a un **Gerente Operativo** registrado en el sistema, obtener todos los backups disponibles que se han realizado.

Tipo de petición: GET

URL: http://localhost:8080/api/backup

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene los backups como un arreglo de objetos, el cual cada backup contiene su fecha, hora y el id.
- En caso de que no haya backups el arreglo será vacío.

---

47) Obtener la frecuencia del backup automático.

Este endpoint permite a un **Gerente Operativo** registrado en el sistema, obtener la frecuencia del backup automatico.

Tipo de petición: GET

URL: http://localhost:8080/api/backup?freq=true/

- Se debe realizar una solicitud GET a la URL especificada.
- Se debe incluir el token de autenticación en el encabezado de la solicitud para identificar a un GO válido.

Resultado:

- Código de estado HTTP 200 OK
- Objeto JSON que contiene la frecuencia (diary diario, weekly semanal, monthly mensual, none ninguna, es decir, no está activado el backup automático).

Códigos de Estado HTTP y Manejo de Errores

Códigos de estado HTTP

Los códigos de estado HTTP en resumen es un número de tres dígitos que genera un servidor en respuesta a una petición. Un ejemplo de estos es el famoso 404 Not Found. Son indicadores estándar que la API utiliza para comunicar el resultado de una solicitud. Aquí se proporciona una breve descripción de los códigos de estado comunes y cómo se aplican en esta API.

Código	Descripción
2xx	**Éxito**
4xx	**Error del cliente**
5xx	**Error del servidor**

Codigo de errores

A continuación se detallan los posibles códigos de estado HTTP y los mensajes de error que podrían surgir al interactuar con la API.

Código	Nombre	Descripción	Problemática
400	**Bad Request**	**Se devuelve cuando la solicitud del cliente es incorrecta o no se puede procesar. Por ejemplo, si los parámetros proporcionados en una solicitud POST no cumplen con los requisitos.**	**Solicitud incorrecta. Revise los parámetros proporcionados.**
401	**Unauthorized**	**Se devuelve cuando la autenticación es necesaria y ha fallado o no se ha proporcionado. Por ejemplo, al intentar acceder a recursos protegidos sin un token de autenticación válido.**	**Usuario no autorizado. El token de autenticación es inválido o ha expirado.**
404	**Not Found**	**Indica que el recurso solicitado no pudo ser encontrado en el servidor. Esto se devuelve cuando se accede a una URL que no existe.**	**No se encontró el recurso solicitado.**
500	**Internal Server Error**	**Este código se utiliza para indicar que ha ocurrido un error interno en el servidor. Es un mensaje genérico de error cuando algo inesperado sucede.**	**Error interno del servidor. Conexión errónea con la base de datos. Se debe comunicar y lo debe revisar el administrador.**