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

Versión 1.0.0

—

Cooperativa Agropecuaria La Paz Limitada (CALP)

Cabana, Iahn, Zandomeni.

02.10.2023

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 el ID del usuario encargado de área cuyo área se desea actualizar en la URL.

- 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 por 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) 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.

---

7) 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.

---

8) 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.

---

9) 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.

---

10) 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).

---

11) 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.

---

12) 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.

---

13) 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).

---

14) 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.

---

15) 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 actividades de mantenimiento en un equipamiento en una fecha y hora específicas. 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 deben proporcionar la fecha y la hora de la alarma de mantenimiento en el cuerpo de la solicitud.

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 deben proporcionar la fecha y la hora de la alarma de mantenimiento en el cuerpo de la solicitud.

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

  hora: La hora de la alarma de mantenimiento en el formato 'HH:MM' (por ejemplo, '14:30').

FetchAPI ejemplo de uso:

```
const body = {
      fecha: '2023-10-15', // Reemplaza con la fecha deseada
      hora: '14:30', // Reemplaza con la hora deseada
};
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:

alarma: objeto JSON con detalles de la alarma creada.

---

16) Actualizar Alarma de Mantenimiento

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** actualizar la fecha y hora de una alarma de mantenimiento existente asociada a un equipamiento 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 de la alarma de mantenimiento que se desea actualizar en la URL. Además, se debe proporcionar la nueva fecha y hora en el cuerpo de la solicitud.

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 o la nueva hora de la alarma de mantenimiento en el cuerpo de la solicitud.

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

  hora: La nueva hora de la alarma de mantenimiento en el formato 'HH:MM' (por ejemplo, '15:00').

FetchAPI ejemplo de uso:

```
const body = {
      fecha: '2023-11-01', // Nueva fecha (reemplaza con
la nueva fecha)
      hora: '15:00', // Nueva hora (reemplaza con la
nueva hora)
};
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.

---

17) Baja (Física) de Alarma

Este endpoint permite a un **Jefe de Mantenimiento** o a un **Gerente Operativo** eliminar una alarma de mantenimiento existente asociada a un equipamiento 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 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.

---

18) 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 y correo electrónico.

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).

  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"
}
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.

---

19) 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 o el correo electrónico 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 o ambos.

- Es posible actualizar solo el nombre, solo el correo electrónico o ambos.

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.

---

20) 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.

---

21) 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.

---

22) 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.

23) 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.

---

24) 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.

---

25) 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.

---

26) 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.

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.

27) 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.

---

28) Crear Tarea

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 y los datos de la tarea en el cuerpo de la solicitud.

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 debe incluir los siguientes campos:

  descripcion: Descripción detallada de la tarea (cadena de texto, no es obligatoria).

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 (no es obligatoria)",
      "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.

---

29) Cargar Tarea

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

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: Descripción de la tarea, en este caso es obligatorio la descripción de la tarea, ya que, como no existe descripción de solicitud propia de un EA o in ID de equipamiento a la cual atender, debe quedar plasmado lo que se realizó en la tarea (en este endpoint la solicitud se crea igual, siendo el usuario que realiza la petición el solicitante y el juez al mismo tiempo).

  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 en este
caso es obligatoria",
      "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', {
  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.

---

30) Actualizar Tarea

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** actualizar una tarea existente. 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 (cadena de texto, no es obligatoria).

  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.

---

31) Finalizar Tarea

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** finalizar una tarea existente. 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.

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.

---

32) 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:

  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.

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 y email.

---

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
- En caso de no encontrar ningún usuario el resultado será un objeto vacío

---

7) 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

---

8) 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.

---

9) 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.

10) 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.

11) 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.

---

12) 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.

---

13) 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.

---

14) 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.

---

15) 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.

---

16) 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}/alarma

- 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.

---

17) 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.

18) 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.

19) Obtener Depósito de Jefe Mantenimiento

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/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.

---

20) 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/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 una lista de todos los depósitos.

---

21) 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/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 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.

22) 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/alarmas/{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.

23) 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.

---

24) 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.

---

25) 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.

---

26) 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.

---

27) 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.

---

28) Obtener Solicitud por ID

Este endpoint permite a un **Jefe de Mantenimiento** o **Gerente Operativo** 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 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.

29) 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.

30) 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 los parámetros query 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 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.

31) 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.

32) Obtener las Tareas de un JUEZ

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

"**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 traduciendolas en tareas, entonces el resultado será un arreglo vacío.

---

33) 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 un ID de tarea no contenga un inventario, el resultado será un arreglo vacío.

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.**