Módulo Centro de Ayuda

- Denuncias(Consultar Sucursal (1,2)) - [getBranchsDesignations]
- Denuncias(Crear demanda (1)) - [store]
- Denuncias(Url Demanda PDF (1)) - [pdf]
- Denuncias(Consultar demanda (1)) - [obtener]

Denuncias(Consultar Sucursal (1,2)) - [getBranchsDesignations]

◫ Descripción

Servicio que obtiene dinámicamente:

- **La lista de sucursales activas con más de 1 empleado asignado**, o
- **La lista de puestos (Designations)** disponibles en el ERP,

dependiendo del parámetro `type` recibido en la solicitud.

Este servicio consulta directamente al ERP mediante los métodos internos `dbErp()` y `ServiceErp()`.

? **Endpoint**

POST `/get-branchs-designations`

? **Request Body**

```
{
  "type": "Sucursal" | "Puesto"
}
```

Valores permitidos:

type	Acción
`"Sucursal"`	Retorna sucursales con empleados asociados.
`"Puesto"`	Retorna todos los puestos (Designation).

? Seguridad

- Requiere autenticación interna del ERP.
- Las consultas se realizan vía `dbErp()` (SQL ERPNext) o `ServiceErp()` (REST del ERP).

? Flujo del Servicio

?? Si type = `"Sucursal"`

1. Construye la consulta SQL:
   - Selecciona sucursales (`tabBranch`).
   - Hace JOIN con empleados (`tabEmployee`).
   - Agrupa y retorna solo sucursales con **más de 1 empleado registrado**.
2. Ejecuta la consulta vía:

   `dbErp("POST", body, /method/send-query-database)`
3. Si encuentra resultados:
   - Devuelve solo el nombre de la sucursal (`br.name`).
4. Si ocurre error:
   - Retorna mensaje de error controlado.

?? Si type = `"Puesto"`

1. Solicita al ERP la lista completa de Designations:

   `GET resource/Designation?limit=None`
2. Extrae solo el nombre del puesto (`name`).
3. Devuelve la lista al cliente.

? Si type ? “Sucursal” ni “Puesto”

Retorna error indicando que la opción no es válida.

? **Response 200 – Ejemplos**

?? Respuesta para type = `"Sucursal"`

```
{
  "valor": true,
  "msg": "Se encontraron sucursales",
  "data": [
    "Sucursal Lima",
    "Sucursal Arequipa",
    "Sucursal Trujillo"
  ]
}
```

?? Respuesta para type = `"Puesto"`

```
{
  "valor": true,
  "msg": "Se encontraron puestos",
  "data": [
    "Asistente Administrativo",
    "Operario de Almacén",
    "Jefe de Sucursal"
  ]
}
```

? Opción no válida

```
{
  "valor": false,
  "msg": "Opcion no valida",
  "data": []
}
```

? Error del servidor

```
{
  "valor": false,
  "msn": "Error de servidor",
  "data": "<detalle_del_error>"
}
```

? Schemas que utiliza

Branch (SQL ERP)

Campos relevantes:

```
br.name
```

Employee

Relacionada solo para contar empleados por sucursal:

```
emp.branch
emp.name
```

Designation

Campos relevantes:

```
{
  "name": "string"
}
```

? Lógica en Pseudocódigo

```
tipo = request["type"]

if tipo == "Sucursal":
    response = SQL:
        SELECT br.name
        FROM Branch br
        LEFT JOIN Employee emp ON emp.branch = br.name
        GROUP BY br.name
        HAVING COUNT(emp.name) > 1

    return nombres de sucursales

else if tipo == "Puesto":
    response = GET Designation
    return lista de name

else:
    return error "Opción no válida"
```

Denuncias(Crear demanda (1)) - [store]

? Descripción

*Este servicio crea un registro oficial de una **denuncia interna** realizada por un colaborador o usuario del sistema.*
Permite registrar denuncias anónimas o identificadas, incluir detalles del hecho, denunciados, fechas, archivos adjuntos y otros campos relevantes.

La información se almacena directamente en el ERP, creando un nuevo documento del tipo:

`DocType: Denuncias`
Incluye reglas de validación para asegurar que los datos enviados cumplan los requisitos mínimos según la normativa interna del área de cumplimiento.

? Endpoint

POST `/denuncias/store`

? Seguridad

Requiere autenticación interna:

- La creación se realiza vía `ServiceErp()`, que requiere un token válido del ERP.
- No se permiten denuncias sin validar los campos obligatorios.

? Flujo del Servicio (Resumen Real)

1. **Lee parámetros del request:**
   - Identificación del denunciante
   - Datos de contacto (si aplica)

- Motivo de denuncia
- Fechas del hecho
- Sucursal
- Denunciados (en JSON)
- Archivo adjunto
- Otros detalles relacionados

2. **Valida que el JSON de denunciados sea correcto.**
3. **Ejecuta múltiples validaciones obligatorias:**
   - Si se identifica → celular y correo son obligatorios.
   - Campos `motivo`, `sucursal`, `desde`, `hasta`, `detalle` deben existir.
   - Si conoce involucrados → mínimo 1 y máximo 3 denunciados.
   - Cada denunciado debe tener nombre, sucursal y área.
4. **Valida y arma estructura del archivo adjunto** si se envía.
5. **Genera un código aleatorio único** que identificará la denuncia.
6. **Crea el documento Denuncias en el ERP:**

   `POST /resource/Denuncias`
7. **Guarda los denunciados como tabla hija (table_16).**
8. **Devuelve confirmación y el código de seguimiento.**

? Request Body (Ejemplo)

```
{
  "identificar": 1,
  "celular": "987654321",
  "correo": "user@example.com",
  "motivo": "Acoso laboral",
  "continua_ocurriendo": 0,
  "sucursal": "SUC-001",
  "desde": "2025-01-01",
  "hasta": "2025-01-15",
  "conoce_involucrados": 1,
  "denunciados": "[{\"nombre\":\"Juan Perez\",\"sucursal\":\"Lima\",\"area\":\"Operaciones\"}]",
  "detalle": "Descripción detallada del hecho...",
  "compromiso": "Ninguno",
  "archivo": "codigo-archivo.pdf"
}
```

? Validaciones Importantes

Campo	Requerido	Condición
identificar	Sí	Debe ser 0 o 1
celular, correo	Obligatorios si identificar = 1	—

Campo	Requerido	Condición
motivo	Sí	—
sucursal	Sí	—
continua_ocurriendo	Sí	Debe ser 0 o 1
desde / hasta	Sí	Fechas del hecho
conoce_involucrados	Sí	0 o 1
denunciados	Condicional	1 a 3 registros si conoce_involucrados = 1
detalle	Sí	—

Si el archivo existe, se formatea a: `https://fileserver.shalomcontrol.com/file-view/<archivo>`

? Response 200 – Ejemplo exitoso

```
{
  "success": true,
  "message": "Se registró correctamente",
  "data": {
    "codigo": "ABC123"
  }
}
```

? Posibles Errores

1. JSON malformado en denunciados

```
{
  "success": false,
  "message": "El campo denunciados debe ser un json codificado"
}
```

2. Campos obligatorios faltantes

```
{
  "success": false,
  "message": "Debe ingresar un celular"
}
```

3. Número incorrecto de denunciados

```
{
  "success": false,
  "message": "Solo puede ingresar hasta 3 denunciados como máximo"
}
```

4. Error al registrar en el ERP

```
{
  "success": false,
  "message": "Ocurrió un error al registrar",
  "response": { ... }
}
```

5. Error inesperado del servidor

```
{
  "success": false,
  "message": "Ocurrió un error al registrar",
  "exception": "<mensaje>"
}
```

? Esquema del DocType Denuncias (Campos usados)

```
{
  "identificar": "0|1",
  "celular": "string|null",
  "correo": "string|null",
  "motivo": "string",
  "continua_ocurriendo": "0|1",
  "sucursal": "string",
  "desde": "date",
  "hasta": "date",
  "conoce_involucrados": 0,
  "detalle": "string",
  "compromiso": "string|null",
  "archivo": "string|null",
  "estado_denuncias": "PENDIENTE",
  "doctype": "Denuncias",
  "table_16": [],
  "codigo_aleatorio": "string"
}
```

? Lógica en Pseudocódigo

```
leer todos los campos del request

validar denunciados (JSON)
validar campos uno por uno

si identificar == 1:
    validar celular & correo

si conoce_involucrados == 1:
    validar entre 1 a 3 denunciados

si hay archivo:
    generar url pública

generar codigo aleatorio

armar payload del ERP

insert = POST /resource/Denuncias

si insert falla:
    retornar error

retornar success + codigo
```

Denuncias(Url Demanda PDF (1)) - [pdf]

? Descripción

*Este servicio obtiene la información completa de una denuncia registrada en el ERP (Frappe/ERPNext), validando primero que exista una denuncia cuyo **código aleatorio** coincida con el parámetro enviado.*

Una vez encontrada:

1. Obtiene la información principal de la denuncia.
2. Consulta nuevamente la denuncia para traer sus tablas hijas (por ejemplo table_16).
3. Construye un array final con los datos completos.
4. Genera un **PDF oficial de la denuncia** usando una plantilla Blade.
5. Retorna el archivo descargable **Demanda.pdf**.

Este servicio se utiliza para permitir a los usuarios descargar un reporte o constancia oficial de la denuncia registrada.

? Endpoint

GET /pdf/{codigo}

□□ *El parámetro `{codigo}` corresponde al campo `codigo_aleatorio` de la denuncia.*

? Parámetros

URL Params

Parámetro	Tipo	Obligatorio	Descripción
`codigo`	string	✔	Código aleatorio que identifica la denuncia

□□ *No recibe parámetros en el body.*

? Seguridad

El servicio realiza llamadas internas al ERP a través de **ServiceErp()**, por lo que requiere autenticación válida configurada internamente.
No necesita token desde el cliente.

? Flujo del Servicio (Resumen real)

1?? Buscar denuncia por código aleatorio

Realiza un GET al recurso: `GET Denuncias?filters=[["codigo_aleatorio","=",<codigo>]]`

Si no encuentra coincidencias → retorna error.

2?? Para cada denuncia encontrada

Hace un GET para traer la información completa del documento con sus tablas hijas: `GET Denuncias/{name}`

Guarda el contenido de la tabla `table_16` y arma un arreglo final.

3?? Generar el PDF

Usa: `PDF::loadView("pdf/demanda", ["data" => $demandas_final[0]])`

Lo genera en tamaño **A5**, orientación **portrait**, y lo retorna como descarga: `Demanda.pdf`

? Response – Ejemplo exitoso (PDF descargable)

El navegador descargará directamente: `Demanda.pdf`

Con el contenido renderizado desde la vista `pdf/demanda.blade.php`.

? Posibles Errores

1. No existe la denuncia

```
{
  "success": false,
  "message": "No se encontró la denuncia"
}
```

2. Error consultando el ERP

```
{
  "success": false,
  "message": "<mensaje de la excepción>"
}
```

3. La denuncia existe, pero falla la obtención completa

(No rompe el flujo; simplemente ignora la denuncia afectada.)

? Schemas utilizados

Denuncias (GET)

Campos utilizados:

```
{
  "name": "string",
  "codigo_aleatorio": "string",
  "table_16": [ ... ],
  ...otros campos
}
```

PDF Output

Basado en la vista: `resources/views/pdf/demanda.blade.php`

? Lógica en Pseudo-código

```
body = {
    filters: [["codigo_aleatorio", "=", codigo]],
    fields: ["*"]
}

demandas = GET Denuncias WHERE codigo_aleatorio=codigo

if demandas is empty:
    return error

foreach denuncia in demandas:
    detalle = GET Denuncias/{name}
    agregar table_16 al resultado final

generar PDF usando vista "pdf/demanda"
retornar archivo: Demanda.pdf
```

Denuncias(Consultar demanda (1)) - [obtener]

? Descripción

Permite **consultar una denuncia registrada** en el ERP usando un **código aleatorio** como identificador de búsqueda.

El servicio:

- Recibe un código único (`codigo`) enviado al usuario.
- Busca en el ERP la denuncia asociada.
- Retorna sus datos principales (estado, creación, fechas de proceso, archivo adjunto, etc.)
- Ajusta la URL del archivo si existe.

Es un servicio utilizado para que un usuario pueda revisar:

- El estado de su denuncia,
- La respuesta del área responsable,
- Y descargar el archivo asociado (si lo tiene).

? Endpoint

POST /obtener

? Request Body

```
{
  "codigo": "string"
}
```

Parámetros

Campo	Tipo	Obligatorio	Descripción
codigo	string	✔ Sí	Código aleatorio generado al registrar la denuncia

---

? Seguridad

Requiere autenticación interna mediante
`$this->general->ServiceErp()`
(usa cookies o token ya configurado en el backend).

? Flujo del Servicio (resumen real)

1. **Obtiene el código** enviado en el request.
2. Construye el body para consultar la denuncia en el ERP:
   - fields: name, creation, estado_denuncias, fechas, archivo, respuesta.
   - filters: `[["codigo_aleatorio","=", $codigo]]`
3. Llama al ERP:
   `GET resource/Denuncias`
4. Si **no existe denuncia**, retorna error.
5. Si existe:
   - Verifica si `archivo_denuncia` no está vacío.
   - Si tiene archivo, le agrega la URL base del servidor.
6. Retorna toda la información encontrada.

? Response 200 – Ejemplo exitoso

```
{
  "success": true,
  "message": "Encontrado",
  "data": {
    "name": "DEN-00012",
    "creation": "2025-01-10 12:40:15",
    "estado_denuncias": "Atendido",
    "fecha_atendido": "2025-01-18",
    "fecha_proceso": "2025-01-15",
    "archivo_denuncia": "https://dominio.com/files/denuncia_12.pdf",
    "respuesta_de_denuncia_atendido": "Se realizó la evaluación correspondiente."
  }
}
```

? Posibles Errores

1?? Denuncia no encontrada


```
{
  "success": false,
  "message": "No se encontró la denuncia"
}
```


2?? Error en la comunicación con el ERP


```
{
  "success": false,
  "message": "Error del servidor: <detalle>"
}
```


3?? Respuesta vacía del ERP


```
{
  "success": false,
  "message": "No se encontró la denuncia"
}
```

? Esquema utilizado (Denuncias – GET)

Campos usados:

```
{
  "name": "string",
  "creation": "datetime",
  "estado_denuncias": "string",
  "fecha_atendido": "date",
  "fecha_proceso": "date",
  "archivo_denuncia": "string",
  "respuesta_de_denuncia_atendido": "string"
}
```

? Lógica en pseudo-código

```
codigo = request.codigo

body = {
    limit: None,
    fields: [name, creation, estado_denuncias, fecha_atendido, fecha_proceso, archivo_denuncia,
respuesta_de_denuncia_atendido],
    filters: [["codigo_aleatorio", "=", codigo]]
}

response = GET Denuncias(body)

if response is error or empty:
    return error "No se encontró la denuncia"

denuncia = response[0]

if denuncia.archivo_denuncia != "":
    denuncia.archivo_denuncia = BASE_CAPACITACION + denuncia.archivo_denuncia

return success data = denuncia
```