Módulo Supervision

- Lista Sucursales (1) - [listSucursalSupervition]
- Informacion de sucursal (1) - [infosupervitionName]
- Puntajes de las categorias (1) - [listCategory]
- Respuestas formulario (1) - [missing_questions_form]
- Respuestas (1) - [missing_questions]
- Adjuntar imagen (1) - [uploadFileErp]
- Actualizar terminos de supervision (1) - [updatequestion]

Lista Sucursales (1) - [listSucursalSupervition]

? Descripción

Este servicio obtiene todas las supervisiones realizadas por un **Supervisor Nacional**, junto con información adicional relacionada a:

- Las sucursales supervisadas
- La fecha de registro
- El documento generado
- La programación de supervisión vinculada
- La fecha real en la que se ejecutó dicha supervisión

El servicio consolida datos procedentes de:

- `Check List del Supervisor Nacional 2`
- `Tabla supervisores`
- `Programación de Supervisores`

y devuelve una lista enriquecida de supervisiones.

? Endpoint

POST `/list-sucursal-supervition`

? Parámetros de Entrada (Request)

Body / FormData

Parámetro	Tipo	Obligatorio	Descripción

`Employe`	string	✔	Código del empleado supervisor cuyo historial de supervisiones será consultado.

Ejemplo:

```
{
  "Employe": "EMP-001245"
}
```

? Seguridad

- Requiere autenticación interna del ERP (token manejado por `ServiceErp()` y `dbErp()`).
- Solo accesible para usuarios válidos dentro del entorno de capacitación.

? Flujo del Servicio (Resumen Detallado)

1?? Validar parámetro recibido

Si `Employe` viene vacío, se retorna un mensaje de validación.

2?? Listar supervisiones creadas por el supervisor

Consulta en el ERP:

```
GET resource/Check List del Supervisor Nacional 2
Filters: supervisor = Employe
Fields: sucursal, fecha, name
```

- Si no hay resultados → Retorna mensaje indicando ausencia de registros.

3?? Obtener la relación con Tabla Supervisores

Con los `name` obtenidos, se consulta:

```
POST send-query-database
FROM tabCheck List del Supervisor Nacional 2 spr
LEFT JOIN tabTabla supervisores tab ON spr.name = tab.id_doctype
WHERE spr.name IN (...)
SELECT spr.name, tab.parent
```

Construye:

- `$datanew[parent] = name`
- `$data_alrevez[name] = parent`

4?? Obtener programación de supervisores

Si existen supervisiones relacionadas, consulta:

```
POST send-query-database
FROM tabProgramacion de Supervisores
WHERE name IN parents
SELECT name, prog_supervisores, fecha_real_de_la_supervicion
```

Se agrega esta información a cada supervisión detectada.

5?? Unir los datos

Cada supervisión final contendrá:

- sucursal
- fecha del checklist
- nombre del documento checklist
- programación de supervisión (`prog_supervisores`)
- fecha real de supervisión (`fecha_real_de_la_supervicion`)

? Response (200 – Ejemplo)

```
{
  "valor": true,
  "msn": "Lista de Sucursales",
  "data": [
    {
      "sucursal": "SUC-014",
      "fecha": "2024-10-03",
      "name": "CHK-SUP-00045",
      "supervision": "SUP-NACIONAL-05",
      "fechaSupervision": "2024-10-04"
    }
  ]
}
```

? Posibles Errores

1. Supervisor no enviado


```
{
  "valor": false,
  "msn": "El campo empleado es obligatorio",
  "data": []
}
```


2. Error del servicio ERP


```
{
  "valor": false,
  "msn": "Ocurrio un error al listar",
  "data": []
}
```


3. Supervisor sin supervisiones registradas


```
{
  "valor": false,
  "msn": "No hay ningun registro creado",
  "data": []
}
```

? Tablas / Recursos involucrados

Check List del Supervisor Nacional 2

Campos usados:

```
{
  "sucursal": "string",
  "fecha": "date",
  "name": "string",
  "supervisor": "string"
}
```

Tabla supervisores

Relación entre registros checklist y programaciones.

Programación de Supervisores

Campos utilizados:

```
{
  "name": "string",
  "prog_supervisores": "string",
  "fecha_real_de_la_supervicion": "date"
}
```

? Pseudocódigo del Servicio

```
if Employe is empty:
    return error

listName = GET Check List supervisor where supervisor=Employe

if listName empty:
    return sin registros

names = extract "name" from listName

relations = QUERY tabla supervisores WHERE spr.name IN names

parents = extract unique parent

programaciones = QUERY programacion supervisores WHERE name IN parents

merge programaciones into listName

return listName enriched
```

Informacion de sucursal (1) - [infosupervitionName]

? Descripción

Obtiene la información general de un checklist del **Supervisor Nacional**, incluyendo:

- Datos principales de la cabecera del checklist
- Promedio de progreso de todas las categorías evaluadas
- Puntaje total acumulado

El servicio combina información proveniente de:

- ERP (doc: *Check List del Supervisor Nacional 2*)
- Servicio interno `listCategory()` (el cual obtiene las secciones/categorías del checklist)

? Endpoint

POST /infosupervition-name

□□ Recibe un parámetro obligatorio en el body:

```
{
  "name": "ID del checklist"
}
```

? Seguridad

Requiere autenticación interna mediante `ServiceErp()` y dependencias del módulo `general`.

? Flujo del Servicio (resumen real)

1. **Valida parámetro** `name`
   Si viene vacío, retorna error utilizando `responseValidate()`.
2. **Obtiene todas las categorías** asociadas al checklist
   Usa:

   `listCategory($request)`
3. **Valida respuesta de categorías**
   - Si ocurre error → retorna mensaje de fallo.
   - Si la data viene vacía → indica que no existen registros creados.
4. **Calcula progresos y puntajes totales**
   Recorre cada categoría para:
   - Sumar `section_puntaje`
   - Sumar `progreso`
   - Contar categorías

   Luego obtiene:
   - `all_progress = total_progreso / cantidad_categorias`
   - `all_point = round(total_puntaje)`
5. **Consulta información de cabecera del checklist en ERP**
   Realiza:

   `GET Check List del Supervisor Nacional 2?filters=[["name","=",<name>]]`
   Campos obtenidos:
   - sucursal
   - fecha
   - name
6. **Agrega métricas calculadas** al resultado:
   - `all_progress`
   - `all_point`
7. **Retorna respuesta final con la información consolidada**

? Request Body

```
{
  "name": "CHK-00045"
}
```

? Response 200 – Ejemplo


```
{
  "valor": true,
  "msn": "Informacion obtenida",
  "data": {
    "sucursal": "LIMA-01",
    "fecha": "2025-01-15",
    "name": "CHK-00045",
    "all_progress": 82.5,
    "all_point": 120
  }
}
```

? Posibles Errores

1. name vacío


```
{
  "valor": false,
  "msn": "Campo requerido vacío"
}
```


2. Error al obtener categorías


```
{
  "valor": false,
  "msn": "Error al obtener la informacion",
  "error": { ... }
}
```


3. No existen registros en categorías


```
{
  "valor": false,
  "msn": "No hay ningun registro creado",
  "data": []
}
```


4. Error consultando al ERP

```
{
  "valor": false,
  "msn": "Error al obtener informacion",
  "data": []
}
```

? Schemas usados

Categoría (respuesta de listCategory)


```
{
  "section_puntaje": "number",
  "progreso": "number"
}
```


Check List del Supervisor Nacional 2 (ERP)


```
{
  "sucursal": "string",
  "fecha": "date",
  "name": "string"
}
```

? Lógica en pseudo-código

```
if name == "":
    return error

categorias = listCategory(request)

if categorias.error:
    return error

if categorias.data empty:
    return no registros

for categoria in categorias:
    total_puntaje += categoria.section_puntaje
    total_progreso += categoria.progreso
    cantidad += 1

GET checklistHeader FROM ERP WHERE name = <name>

header.all_progress = total_progreso / cantidad
header.all_point = round(total_puntaje)

return header
```

Puntajes de las categorias (1) - [listCategory]

? Descripción

Obtiene y procesa dinámicamente todas las **categorías**, **preguntas** y **porcentajes de avance** del **Check List del Supervisor Nacional 2**, basándose en su estructura de campos del DocType y los valores registrados en un documento específico.

Este servicio:

1. Lee la estructura del DocType para identificar:
   - Secciones (categorías).
   - Preguntas tipo **Check**, **Data**, **Attach Image**.
2. Recupera los valores reales del documento filtrado por `name`.
3. Agrupa los campos por categoría.
4. Calcula el avance (%) y puntaje por sección.
5. Devuelve un resumen por categoría con:
   - Total de campos.
   - Campos completados.
   - Progreso (%).
   - Puntaje asignado.

? Endpoint

POST /list-category

? Request Body

```
{
  "name": "<id del documento Check List del Supervisor Nacional 2>"
}
```

Parámetros

Campo	Tipo	Requerido	Descripción
name	string	✔	ID del documento cuyo checklist se evaluará

? Seguridad

El servicio utiliza autenticación interna mediante: `$this->general->ServiceErp()`

Por lo tanto, requiere credenciales válidas para consumir el ERP.

? Flujo del Servicio (Explicación Técnica Real)

1?? Obtener estructura del DocType

Se consulta: `GET resource/DocType/Check List del Supervisor Nacional 2`

Esto trae **todos los campos** del formulario.
El servicio filtra únicamente:

- Section Break → marca inicio de categoría
- Check
- Data
- Attach Image

También **omite** los campos de puntaje global:

- porcentaje_de_imagen_y_presentación
- puntaje_de_imagen_y_presentación

2?? Construcción dinámica de categorías y preguntas

Ejemplo de agrupación generada:

```
{
  "Gestión Documentaria": [
    "documentos_autoridades",
    "imagen_14",
    "comentario_14",
    ...
  ],
  "Presentación del almacén": [
    "foto_area",
    "check_orden",
    "comentario_orden"
  ]
}
```

3?? Recuperar valores del documento

Se unen todos los fields encontrados y se hace:

```
GET resource/Check List del Supervisor Nacional 2
?fields=[...fields]
&filters=[["name","=",name]]
```

Esto trae los valores registrados del documento.

4?? Agrupar los valores por categoría

El servicio arma una estructura:

```
{
  "Gestión Documentaria": {
     "documentos_autoridades": 1,
     "imagen_14": "/files/img.png",
     "comentario_14": "OK"
  }
}
```

5?? Calcular puntajes y progreso

Para cada categoría:

- Se cuenta el total de campos (Check, Data, Imagen)
- Se cuentan los completados:
  - Check → valor 1
- Se asigna un puntaje:
  - Cada check activo suma **1.82**
- Se calcula:

`progreso = (complete_fields / total_fields_categoria) * 100`

⚠ Nota: El servicio divide el total de campos entre 3 (`$maxFields = $maxFields / 3`) para compensar que cada pregunta tiene 3 tipos de fields (check, data, image).

? Response 200 – Ejemplo


```
{
  "valor": true,
  "msn": "Lista De Categorias",
  "data": [
    {
      "category": "Gestión Documentaria",
      "progreso": 67,
      "total_fields": 6,
      "complete_fields": 4,
      "section_puntaje": 7.28
    },
    {
      "category": "Áreas Críticas",
      "progreso": 50,
      "total_fields": 4,
      "complete_fields": 2,
      "section_puntaje": 3.64
    }
  ]
}
```

? Posibles Errores

1. No se puede obtener el DocType


```
{
  "value": false,
  "msn": "Ocurrio un error al consular"
}
```


2. Documento no encontrado

(El servicio devolverá valores vacíos por categoría)

? Estructuras Utilizadas

DocType (GET)

Campos analizados:

```
{
  "fieldname": "string",
  "label": "string",
  "fieldtype": "Section Break | Check | Data | Attach Image"
}
```

? Lógica en pseudocódigo

```
doctype = GET DocType fields
categorias = {}

foreach field in doctype.fields:
    if field is Section Break:
        categoria_actual = field.label

    if field is Check/Data/Image:
        categorias[categoria_actual].append(field.fieldname)


# Obtener valores reales del documento
values = GET Check List del Supervisor Nacional 2 where name = request.name

# Agrupar valores por categoría
foreach categoria in categorias:
    newQuestions[categoria] = tomar valores según fieldname


# Calcular porcentaje y puntaje
foreach categoria in newQuestions:
    total = count(fields) / 3
    completos = fields con valor 1
    puntaje = completos * 1.82

    agregar resultado al response
```

Respuestas formulario (1) - [missing_questions_form]

? Descripción

Este servicio identifica **qué preguntas no han sido completadas** dentro de un formulario del DocType:

Check List del Supervisor Nacional 2

El servicio analiza:

- Una **categoría** específica del formulario (Section Break).
- Un documento del Doctype, identificado por **name**.
- Obtiene **todos los campos relevantes** (Check, Data, Attach Image).
- Compara estos campos con los valores del documento
- Devuelve **solo los campos incompletos**, agrupados por categoría.

Se usa principalmente para:

✔ Validar formularios incompletos
✔ Mostrar al usuario qué preguntas faltan llenar
✔ Controlar flujos de supervisión/inspecciones

? Endpoint

POST `/missing-questions-form`

? Parámetros del Request

Campo	Tipo	Obligatorio	Descripción
category	string	✔ Sí	Nombre de la sección del formulario (label del Section Break).

Campo	Tipo	Obligatorio	Descripción
name	string	✔ Sí	ID del documento del Doctype "Check List del Supervisor Nacional 2".

Ejemplo de request:

```
{
  "category": "Limpieza y Orden",
  "name": "CHK-SUP-00015"
}
```

? Seguridad

Utiliza autenticación interna a través de:

`$this->general->ServiceErp()`
No requiere autenticación externa del usuario final.
Toda la data se obtiene desde el ERP.

? Flujo del Servicio (Resumen Técnico)

1?? Validaciones iniciales

- Verifica que **category** no esté vacío.
- Verifica que **name** no esté vacío.

Si falta un campo → retorna error inmediato.

2?? Obtiene estructura del formulario

GET al Doctype: `GET /resource/DocType/Check List del Supervisor Nacional 2`

De ahí obtiene:

- Section Breaks
- Campos tipo Check
- Campos tipo Data
- Campos tipo Attach Image

Y construye un arreglo organizado por secciones:

```
{
  "Nombre de Sección": {
      "fields_name": { fieldname : label },
      "data_name":   { fieldname : label },
      "img_name":    { fieldname : label }
  }
}
```

Se excluyen automáticamente:

- porcentaje_de_imagen_y_presentación
- puntaje_de_imagen_y_presentación

3?? Filtra solo la categoría solicitada

Busca dentro de todas las secciones y selecciona solo: `questions[category]`

Si no existe → retorna vacío.

4?? Obtiene valores del documento filtrado por name

`GET /resource/Check List del Supervisor Nacional 2?filters=[["name","=",<name>]]`
Aquí obtiene los valores reales del formulario.

5?? Determina qué preguntas faltan completar

Compara para cada campo: `si campo == 0 → está incompleto → se agrega a la lista`

Guarda:

- fieldnames faltantes

- Texto de la pregunta
- Identificación por sección

6?? Retorna solo las preguntas faltantes

? Response 200 – Ejemplo Real

```
{
  "valor": true,
  "msn": "Missing Questions",
  "data": {
    "Limpieza y Orden": [
      "pregunta_1",
      "pregunta_3"
    ],
    "questions": [
      "¿Los ambientes se encuentran limpios?",
      "¿El personal cumple con el orden establecido?"
    ]
  }
}
```

? Posibles Errores

1?? category vacío

```
{
  "valor": false,
  "msn": "El campo category es requerido"
}
```

2?? name vacío

```
{
  "valor": false,
  "msn": "El campo name es requerido"
}
```

3?? El Doctype no responde

```
{
  "valor": false,
  "msn": "Ocurrio un error al consular"
}
```

4?? Error en consulta de valores del documento

```
{
  "valor": false,
  "msn": "Ocurrio un error durante el proceso, intente de nuevo."
}
```

? Estructuras Usadas

? DocType: Check List del Supervisor Nacional 2

Campos analizados:

- Section Break
- Check
- Data
- Attach Image

Se ignoran:

- porcentaje_de_imagen_y_presentación
- puntaje_de_imagen_y_presentación

? Algoritmo en Pseudocódigo

```
validar(category)
validar(name)

doctype = GET DocType structure

preguntas = extraer_secciones_y_campos(doctype)

selectedCategory = preguntas[category]

campos = obtener_fieldnames(selectedCategory)

valores = GET Documento ERP con esos campos

missing = []

para cada campo en selectedCategory:
    si valor == 0:
        agregar a missing

return missing
```

Respuestas (1) - [missing_questions]

? Descripción

Este servicio obtiene las **preguntas incompletas y completas** de una categoría específica del Doctype **“Check List del Supervisor Nacional 2”**, basándose en un registro identificado por su `name`.

El servicio:

1. Obtiene toda la estructura de campos del Doctype.
2. Identifica qué campos pertenecen a cada categoría (Section Break).
3. Filtra únicamente los campos tipo:
   - **Check**
   - **Data** *(comentado actualmente)*
   - **Attach Image** *(comentado actualmente)*
4. Consulta los valores reales del formulario (`0` o `1`).
5. Determina qué preguntas están completas o incompletas.
6. Devuelve solo la categoría solicitada.

Es un servicio orientado a auditorías y evaluación operacional.

? Endpoint

POST /missing-questions

? Parámetros (Request Body)

Campo	Tipo	Obligatorio	Descripción
`name`	string	✔	Identificador del documento del Doctype “Check List del Supervisor Nacional 2”.

Campo	Tipo	Obligatorio	Descripción
`category`	string	✔	Nombre de la categoría (Section Break) a analizar.

Ejemplo:

```
{
  "name": "CHK-SUP-00045",
  "category": "Gestión Documentaria"
}
```

? Seguridad

Requiere autenticación ERP válida, manejada internamente por **ServiceErp()**.

? Flujo del Servicio (Resumen)

1. **Validación de parámetros**
   Si `name` o `category` vienen vacíos → error inmediato.
2. **Obtener estructura del Doctype**
   GET `/resource/DocType/Check List del Supervisor Nacional 2`
   - Extrae los campos agrupados por sección (Section Break).
   - Identifica solo campos tipo **Check**.
3. **Agrupar preguntas por categoría**
   - Mapea `category` → lista de campos (fieldname → label).
4. **Construir lista total de fields para consultar sus valores**
   Ejemplo:

   `["documentos_autoridades", "imagen_14", "comentario_14", ...]`
5. **Consultar valores reales del registro**
   GET `/resource/Check List del Supervisor Nacional 2?fields=[...]&filters=[["name","=",name]]`
6. **Clasificar por categoría**:
   - Si campo = `0` → **Incompleta**
   - Si campo = `1` → **Completa**
7. **Retornar solo la categoría solicitada.**

? Response 200 – Ejemplo

```
{
  "valor": true,
  "msn": "Missing Questions",
  "data": {
    "Incompletas": [
      "Falta documento de autoridades",
      "Informe integrado no subido"
    ],
    "Completas": [
      "Imagen de fachada",
      "Checklist SST completado"
    ]
  }
}
```

? Posibles Errores

1. Parámetro vacío


```
{
  "valor": false,
  "msn": "El campo name es obligatorio"
}
```


2. Error al consultar el DocType


```
{
  "value": false,
  "msn": "Ocurrio un error al consultar"
}
```


3. Error consultando los valores del registro


```
{
  "valor": false,
  "msn": "Ocurrio un error durante el proceso",
  "error": { ... }
}
```

? Estructuras utilizadas

?? Doctype: **Check List del Supervisor Nacional 2**

Campos relevantes:

Tipo	Uso
Section Break	Agrupar preguntas por categoría
Check	Representa preguntas de Sí/No
Data	*(comentado, no procesado actualmente)*
Attach Image	*(comentado, no procesado actualmente)*

Ejemplo de estructura procesada:

```
[
  "Gestión Documentaria" => [
      "documentos_autoridades" => "1. Documentos de autoridades",
      "informativo_integrado"  => "2. Informativo integrado",
  ],
  "Seguridad Operacional" => [
      "extintores" => "1. Extintores en regla",
  ]
]
```

? Pseudocódigo Real

```
validate name, category

doctypeFields = GET DocType Check List

questions = group fields by Section Break (only Check fields)

allFields = extract all fieldnames used in questions

values = GET Check List record where name = request.name

For each category:
    For each field:
        if value == 0 → add to Incompletas
        if value == 1 → add to Completas
```

`return data of selected category`

Adjuntar imagen (1) - [uploadFileErp]

? Descripción

Este servicio permite **subir un archivo al ERP** utilizando el endpoint interno `method/upload_file`.

Funciona enviando el archivo recibido desde el request PHP (`$_FILES["file"]`) mediante **cURL** hacia el servicio de carga del ERP y devuelve los metadatos del archivo subido:

- URL pública
- Fecha de creación
- Nombre del archivo
- Tamaño del archivo

Es un servicio auxiliar utilizado por funcionalidades que requieren cargar documentos o imágenes hacia el servidor del ERP.

? Endpoint

POST `/upload-file-erp`

□□ Requiere enviar un archivo en el campo `file` del formulario.

? Seguridad

No utiliza el token del usuario.
El servicio realiza la subida como **Guest**, usando los headers de cookie:

```
Cookie: full_name=Guest; sid=Guest; system_user=no; user_id=Guest; user_image=
```

➡ Esto significa que la API de ERP ya está configurada para permitir carga de archivos desde clientes externos bajo permisos Guest.

? Flujo del Servicio (resumen)

1. Captura el archivo enviado en `$_FILES["file"]`.
2. Crea un objeto `CURLFile` con:
   - ruta temporal (`tmp_name`)
   - tipo MIME
   - nombre original
3. Envía el archivo al endpoint:

   `POST /method/upload_file`
4. Espera la respuesta del ERP.
5. Decodifica la respuesta JSON.
6. Retorna un objeto con:
   - url del archivo
   - fecha de creación
   - nombre
   - tamaño

? Request Body

Debe enviarse como `multipart/form-data`.

Ejemplo (form-data):

Campo	Tipo	Descripción
file	File	Archivo a subir

Ejemplo en HTML:

```
<form method="POST" enctype="multipart/form-data">
   <input type="file" name="file">
</form>
```

? Response 200 – Ejemplo

```
{
  "url": "/files/documento.pdf",
  "creation": "2025-01-14 09:43:55.12893",
  "file_name": "documento.pdf",
  "file_size": 120394
}
```

? Posibles Errores

1. Error al procesar el archivo


```
{
  "msn": "Failed to open stream..."
}
```


2. Error en cURL


```
{
  "msn": "cURL Error #:Timeout..."
}
```


3. El ERP no devuelve estructura válida


```
{
  "msn": "Error al procesar respuesta del ERP"
}
```

? Estructura de respuesta del ERP

El ERP retorna algo como:

```
{
  "message": {
    "file_url": "/files/image.png",
    "creation": "2025-01-15 10:45:00.12584",
    "file_name": "image.png",
    "file_size": 48204
  }
}
```

? Lógica en pseudo-código

```
try:
    file = new CURLFile(tmp, type, name)
catch error:
    return error

send file to ERP via cURL → POST /method/upload_file

if error in curl:
    return error

parse response

return {
    url: response.message.file_url,
    creation: response.message.creation,
    file_name: response.message.file_name,
    file_size: response.message.file_size
}
```

Actualizar terminos de supervision (1) - [updatequestion]

? Descripción

Actualiza los campos (preguntas) de un documento del Doctype **“Check List del Supervisor Nacional 2”** dentro del ERP, permitiendo modificar dinámicamente cualquier grupo de preguntas o valores enviados desde la aplicación.

El servicio **solo actualiza los campos enviados**, y valida previamente que realmente existan cambios antes de ejecutar la operación.

? Endpoint

POST /updatequestion

? Request Body

```
{
  "questions": "{...}",
  "name": "CHK-LIST-0001"
}
```

? Parámetros

Campo	Tipo	Obligatorio	Descripción
questions	string (JSON-stringified)	✔	Objeto JSON con los campos a modificar.
name	string	✔	ID del documento del check list que se actualizará.

Ejemplo del campo **questions**:

```
{
  "pregunta_1": "SI",
  "pregunta_2": "NO",
  "observacion": "Todo conforme"
}
```

? Seguridad

Requiere autenticación interna mediante: `$this->general->ServiceErp(...)`

➡ Se usa el token y contexto configurado internamente para comunicarse con el ERP.

? Flujo del Servicio (resumen real)

1. **Valida parámetros obligatorios**
   Si `questions` o `name` viene vacío, retorna error de validación.
2. **Valida que existan cambios reales**
   Si `questions == '{}'`, se considera que no hubo modificación.
3. **Convierte a array el JSON recibido**

   `$questions = json_decode($questionsForm, true);`
4. **Genera el payload a actualizar**
   Todos los campos recibidos se envían directamente al ERP.
5. **Realiza la actualización mediante API ERP**

   `PUT resource/Check List del Supervisor Nacional 2/{name}`
6. **Retorna el resultado de la operación**

? Response 200 – Ejemplos

? Actualización correcta

```
{
  "valor": true,
  "msn": "Actualizado Correctamente"
}
```

?? No hubo cambios enviados

```
{
  "valor": false,
  "msn": "Usted no realizo ningun cambio"
}
```

? Error de validación

```
{
  "valor": false,
  "msn": "El campo questions es obligatorio"
}
```

? Falla en el PUT

```
{
  "valor": false,
  "msn": "Error al actualizar"
}
```

? Posibles Errores

Error	Descripción
Parámetros vacíos	Si `questions` o `name` está vacío.
Sin cambios	Cuando el JSON está vacío (`{}`).
Error en ERP	Si el PUT falla por permisos, campos inexistentes o problemas del servidor.
Error inesperado	Respuesta falsa o nula desde ServiceErp.

? Schemas (datos usados)

? Check List del Supervisor Nacional 2 (PUT)

Campos dinámicos según las preguntas.
Ejemplo:

```
{
  "pregunta_1": "SI",
  "pregunta_2": "NO",
  "observacion": "Correcto"
}
```

? Lógica en pseudo-código

```
questionsForm = request.questions
name = request.name

if questionsForm empty → return error
if name empty → return error

if questionsForm == "{}" → return "no hizo cambios"

questions = decode(questionsForm)
fields = questions

update = PUT /resource/Check List del Supervisor Nacional 2/{name} with fields

if update fails → return error

return success
```