Módulo Contratación

- Documento de Ingresos (Subir Documentos de ingresos (1)) - [documentosIngreso]
- Documento de Ingresos (Formulario Ficha de Personal (1)) - [formFichaPersonal]
- Documento de Ingresos (Verificar Registro de Postulación (1)) - [validatePersonalRequirement]
- Documento de Ingresos (Registrar Proceso de Descarga (1)) - [registerProcessApp]
- Documento de Ingresos (Registrar Equipamiento (1)) - [store_equipament]
- Documento de Ingresos (Registrar Proceso de Descarga (1, 2,3,4,5)) - [registerProcessApp]
- Contrato de Trabajo(Url Contrato de trabajo PDF (1)) - [printContractPerDesignation]

Documento de Ingresos (Subir Documentos de ingresos (1)) - [documentosIngreso]

? Descripción

*Registra y actualiza los **documentos obligatorios de ingreso** de un empleado dentro del ERP, así como su información bancaria y de fondo de pensiones.*

Este servicio valida la información mínima necesaria antes de actualizar el documento Employee en el ERP.

□□ Es un servicio crítico para el proceso de onboarding, ya que sin este registro el empleado no continúa con otros procesos internos.

? Endpoint

POST `/documentos-ingreso`

? Parámetros de entrada (Request Body)

```
{
  "employee": "EMP-00123",
  "banco": "BCP",
  "dni": "12345678",
  "copia_dni_de_hijos": "url_file",
  "certificado_de_estudios": "url_file",
  "cv": "url_file",
  "copia_dni_de_conyuge": "url_file",
  "certificado_de_trabajo": "url_file",
  "certijoven": "url_file",
  "copia_de_recibo": "url_file",
  "fondo_pensiones": "AFP",
  "afiliado_fondo_pensiones": "SI"
}
```

? Seguridad

Requiere token válido del ERP (la autenticación se maneja internamente mediante `ServiceErp()`).

? Flujo del Servicio (Resumen Real)

1. **Valida que existan campos obligatorios**:
   - employee
   - banco
   - fondo_pensiones
   - afiliado_fondo_pensiones (“SI” o “NO”)
2. Construye un arreglo con los datos a actualizar:

```
$data = [
  "eleccion_banco" => banco,
  "elección_fondo_pensiones" => fondo_pensiones,
  "afiliado_fondo_pensiones" => afiliado_fondo_pensiones,
  ...
];
```

3. Si el fondo de pensiones es **ONP**, también actualiza:

```
"fondo_de_pensiones" => "ONP"
```

4. Si se envían documentos opcionales (DNI, copia de DNI de hijos, CV, certificado de trabajo, etc.), también se agregan al payload.
5. Realiza la actualización del empleado:

`PUT /resource/Employee/{employee}`

6. Devuelve la respuesta del ERP junto con los datos enviados.

? Response 200 – Ejemplo

```
{
  "valor": true,
  "msn": "Documentos registrados correctamente",
  "data": {
    "eleccion_banco": "BCP",
    "elección_fondo_pensiones": "AFP",
    "afiliado_fondo_pensiones": "SI",
    "dni": "12345678",
    "cv": "url_file",
    "certificado_de_trabajo": "url_file"
  }
}
```

? Posibles Errores

1. Falta el employee

```
{
  "valor": false,
  "msn": "Falta key employee"
}
```

2. Falta banco

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

3. Falta fondo de pensiones

```
{
  "valor": false,
  "msn": "El fondo de pensiones es obligatorio"
}
```

4. Afiliado fondo pensiones inválido

```
{
  "valor": false,
  "msn": "Valor no permitido en Afiliado a Fondo de Pensiones"
}
```

5. Error del ERP al actualizar

```
{
  "valor": false,
  "msn": "Error al actualizar documentos",
  "data": { ...respuesta del ERP... }
}
```

? Data Model Usado

Employee (PUT)

Campos actualizables:

```
{
  "eleccion_banco": "string",
  "elección_fondo_pensiones": "AFP|ONP",
  "afiliado_fondo_pensiones": "SI|NO",
  "dni": "string",
  "copia_dni_de_hijos": "string|fileurl",
  "certificado_de_estudios": "string|fileurl",
  "certijoven": "string|fileurl",
  "cv": "string|fileurl",
  "copia_dni_de_conyuge": "string|fileurl",
  "certificado_de_trabajo": "string|fileurl",
  "copia_de_recibo": "string|fileurl",
  "asignacion_familiar_2": 0|1,
  "fondo_de_pensiones": "AFP|ONP"
}
```

? Pseudocódigo de la lógica

```
if employee is empty → error
if banco is empty → error
if fondo_pensiones is empty → error
if afiliado_fondo_pensiones not in ("SI","NO") → error

data = {
    eleccion_banco: banco,
    elección_fondo_pensiones: fondo_pensiones,
    afiliado_fondo_pensiones: afiliado_fondo_pensiones
}

if fondo_pensiones == "ONP":
    data["fondo_de_pensiones"] = "ONP"

Agregar campos opcionales si existen

PUT Employee/{employee} with data

return respuesta del ERP + msn "Documentos registrados correctamente"
```

Documento de Ingresos (Formulario Ficha de Personal (1)) - [formFichaPersonal]

? Descripción

*Registra la **Ficha Personal** del trabajador, permitiendo actualizar:*

- Datos de contacto de emergencia
- Información del cónyuge/conviviente según estado civil
- Registro del proceso en el módulo *historial_procesos_app*

Es un servicio utilizado por la aplicación para completar los datos obligatorios que el trabajador debe registrar para continuar con otros procesos internos.

*Este servicio actualiza directamente el documento **Employee** en el ERP.*

? Endpoint

`POST /form-ficha-personal`

? Request Body (JSON)

```
{
  "empleado": "EMP-0001",
  "contactoEmergencia": {
    "nombreCompleto": "Juan Perez",
    "parentesco": "Hermano",
    "telefono": "987654321"
  },
  "estadoCivil": {
    "nombreCompleto": "Ana Torres",
    "fechaNacimiento": "1990-01-01",
    "ocupacion": "Administradora",
    "centroTrabajo": "Empresa X",
    "direccion": "Av. Los Olivos 123"
  },
  "datosFamiliares": [...]
}
```

Los campos **contactoEmergencia** y **estadoCivil** vienen como JSON string y se decodifican dentro del servicio.

? Seguridad

- Requiere autenticación interna ERP vía `ServiceErp()`.
- Solo accesible para usuarios autenticados en la app.
- Valida que el Employee exista antes de actualizar.

? Flujo del Servicio (Paso a Paso)

1?? Validaciones iniciales

- Verifica que **estadoCivil** exista.
- Verifica que **contactoEmergencia** exista.
- Si faltan datos, devuelve error.

2?? Obtiene información del empleado

`GET /resource/Employee/{empleado}`

Valida que el empleado exista y revisa si tiene estado civil registrado en el ERP:

- Si es **Casado/a** o **Conviviente**, revisa que toda la información del cónyuge esté completa.

3?? Determina nuevo estado civil

Si el ERP no tiene estado civil registrado:

- Si `estadoCivil["nombreCompleto"]` existe → **Casado/a**
- Si no existe → **Soltero/a**

4?? Construye el body para actualizar Employee

Campos incluidos:

```
{
  "person_to_be_contacted": "...",
  "relation": "...",
  "emergency_phone_number": "...",
  "nombre_completo_conyugue": "...",
  "fecha_de_nacimiento_conyugue": "...",
  "ocupación_conyugue": "...",
  "centro_de_trabajo_conyugue": "...",
  "dirección_actual_conyugue": "...",
  "estado_civil_personal": "Casado/a"
}
```

Todos los textos se envían en mayúsculas.

5?? Actualiza Employee

`PUT /resource/Employee/{empleado}`

6?? Registra proceso de Ficha Personal

Inserta un registro en MySQL (`historial_procesos_app`) con:

- proceso = "registerFichaPersonal"
- fecha actual
- empleado

Sirve para el tracking de documentos obligatorios.

7?? Respuesta exitosa

Devuelve:

```
{
  "valor": true,
  "msg": "Se registró correctamente"
}
```

? Response 200 – Ejemplo

?? Caso exitoso

```
{
  "valor": true,
  "msg": "Se registró correctamente",
  "data": []
}
```

? Error: falta información del estado civil

```
{
  "valor": false,
  "msg": "No tiene su información de estado civil, debe comunicarse con Soporte",
  "data": []
}
```

? Error: datos incompletos del cónyuge

```
{
  "valor": false,
  "msg": "Complete la información del cónyugue o conviviente. Información faltante: fechaNacimiento",
  "data": []
}
```

? Error al actualizar en el ERP

```
{
  "valor": false,
  "msg": "Ocurrió un error al actualizar la información del trabajador.",
  "data": { ...respuestaERP... }
}
```

? Posibles Errores Detallados

Error	Descripción
Missing estadoCivil	No se envió estado civil o JSON inválido
Missing contactoEmergencia	Falta información mínima para completar la ficha
Employee no encontrado	GET Employee/{empleado} no encontró registro
Información incompleta del cónyuge	Falta un campo requerido cuando el estado civil es casado/conviviente
Error de actualización	PUT Employee falló
Error registrando proceso	Inserción en historial_procesos_app falló

? Schemas (Estructuras Utilizadas)

Employee (GET y PUT)

Campos usados:

```
person_to_be_contacted
relation
emergency_phone_number
nombre_completo_conyugue
fecha_de_nacimiento_conyugue
ocupación_conyugue
centro_de_trabajo_conyugue
dirección_actual_conyugue
estado_civil_personal
```

historial_procesos_app (INSERT)

```
usuario
empleado
proceso
fecha
estado
```

? Lógica en pseudo-código

```
empleado = request.empleado
contacto = json_decode(request.contactoEmergencia)
civil = json_decode(request.estadoCivil)

if civil is null or empty:
    return error("No tiene información de estado civil")

if contacto is null:
    return error("Falta información de contacto de emergencia")

employee = GET Employee/{empleado}

if civil.previous == Casado or Conviviente:
    validar campos completos de cónyuge

newCivil = employee.estadoCivil != "" ? employee.estadoCivil :
           (civil.nombreCompleto ? "Casado/a" : "Soltero/a")

bodyPUT = construir campos del contacto + cónyuge

update = PUT Employee/{empleado}

if update.ok:
    insertar registro en historial_procesos_app
    return success
else:
    return error("Ocurrió un error al actualizar")
```

Documento de Ingresos (Verificar Registro de Postulación (1)) - [validatePersonalRequirement]

? Descripción

Este servicio valida si un postulante (que ya es empleado o está en proceso de contratación) ***cumple con las condiciones necesarias para continuar el proceso de Alta y Requerimiento de Personal****.*

La validación se basa en:

- Información del **Employee**
- Su **postulación (Job Applicant)**
- Su **Convocatoria (Job Opening)**
- Su **Terminal / Sucursal (Branch)**
- El **Requerimiento de Personal** asociado a la convocatoria

Este servicio permite determinar si se puede generar su trámite de contratación y qué parámetros corresponden a su modalidad de trabajo y tipo de jornada.

? Endpoint

POST /validate-personal-requirement

? Request Body

```
{
  "document": "12345678"
}
```

Parámetros:

- **document**: DNI o pasaporte del postulante.

? Seguridad

Requiere autenticación interna hacia el ERP mediante `dbErp()` y rutas definidas en `APICAPACITACION` .

? Flujo del Servicio (Resumen)

1?? Validar si el documento pertenece a un *Employee*

Consulta en `tabEmployee`:

```
SELECT employment_type, tipo_de_jornada
FROM tabEmployee
WHERE passport_number = :document
```

- Si no existe → □ devuelve error.

2?? Buscar su postulación más reciente (Job Applicant)

Consulta:

```
SELECT job_title
FROM tabJob Applicant
WHERE numero_de_documento = :document AND docstatus != 2
ORDER BY creation DESC
```

- Si no tiene postulación → □ retorna error.

3?? Obtener información de la Convocatoria (Job Opening)

Consulta:

```
SELECT modalidad_de_trabajo, modalidad, sucursal
FROM tabJob Opening
WHERE name = :job_title AND docstatus != 2
```

- Determina:
  - **Tipo de jornada / modalidad** de la convocatoria.
  - **Sucursal asignada** al postulante.

4?? Obtener información de la Sucursal (Branch)

Consulta:

```
SELECT division_nacional
FROM tabBranch
WHERE name = :sucursal AND docstatus != 2
```

- Determina la zona RRHH responsable.
- Según la zona, define qué tabla usar para el Requerimiento de Personal:
  - **Lima →** `tabRequerimiento de Personal Lima`
  - **Otros →** `tabRequerimiento de Personal`

5?? Obtener el Requerimiento de Personal asociado a la convocatoria

```
SELECT tipo_de_jornada
FROM {RequerimientoTabla}
WHERE documento_convocatoria = :job_title AND docstatus != 2
```

- Si no existe RP → el proceso de contratación debe repetirse.

6?? Respuesta exitosa

Devuelve:

- Tipo de jornada
- Modalidad del puesto
- Código de convocatoria

? Response 200 – Ejemplo


```
{
  "valor": true,
  "Message": "Exito.",
  "data": {
    "tipo_jornada": "Tiempo completo",
    "modalidad": "Presencial",
    "convocatoria": "JOB-2025-00123"
  }
}
```

? Posibles Errores

1. No existe empleado con el documento


```
{
  "valor": false,
  "Message": "Surgió un error, contactar con soporte.",
  "data": []
}
```


---

2. No existen postulaciones válidas


```
{
  "valor": false,
  "Message": "No se encontró un registró de su postulación a una de nuestras convocatorias, comuniquese con su administrador.",
  "data": []
}
```


---

3. Convocatoria no encontrada


```
{
  "valor": false,
  "Message": "No se encontró la convocatoria vinculada a su postulación, comuniquese con su administrador.",
  "data": []
}
```


---

4. Sucursal/Terminal inexistente


```
{
  "valor": false,
  "Message": "No existe la terminal del usuario.",
  "data": []
}
```


---

5. Requerimiento de Personal no creado


```
{
  "valor": false,
  "Message": "La convocatoria del postulante no tiene REQUERIMIENTO DE PERSONAL creada, realizar nuevamente el proceso de contratación.",
  "data": []
}
```


---

6. Error del servidor


```
{
  "valor": false,
  "Message": "Ocurrio un error en el servidor.",
  "data": []
}
```

? Esquemas utilizados

Employee

Campos consultados:


```
{
  "employment_type": "string",
  "tipo_de_jornada": "string"
}
```

Job Applicant

```
{
  "job_title": "string"
}
```

Job Opening

```
{
  "modalidad_de_trabajo": "string",
  "modalidad": "string",
  "sucursal": "string"
}
```

Branch

```
{
  "division_nacional": "string",
  "name": "string"
}
```

Requerimiento de Personal

```
{
  "tipo_de_jornada": "string",
  "name": "string"
}
```

? Lógica en Pseudocódigo

```
employee = GET Employee WHERE passport_number = document
if not employee:
    error("Empleado no encontrado")

postulacion = GET Job Applicant WHERE document = document ORDER BY creation DESC
if not postulacion:
    error("Postulación no encontrada")

convocatoria = GET Job Opening WHERE name = postulacion.job_title
if not convocatoria:
    error("Convocatoria no encontrada")

branch = GET Branch WHERE name = convocatoria.sucursal
select RP_table = Branch.zone == Lima ? RP_Lima : RP_Normal

rp = GET RP_table WHERE documento_convocatoria = job_title
if not rp:
    error("RP no creada")

return {
    tipo_jornada: rp.tipo_de_jornada,
    modalidad: convocatoria.modalidad || modalidad_de_trabajo,
    convocatoria: job_title
}
```

Documento de Ingresos (Registrar Proceso de Descarga (1)) - [registerProcessApp]

? Descripción

Registra en la tabla `historial_procesos_app` cada evento o proceso realizado por un usuario dentro de la aplicación móvil.
Además, dependiendo del tipo de proceso, realiza validaciones adicionales como:

- Validación de renovación de contrato
- Validación de cambio de modalidad
- Creación automática de usuario conductor en sistema externo
- Validación de existencia del empleado en ERP

Este servicio funciona como logger centralizado de acciones críticas del trabajador.

? Endpoint

POST `/register-process-app`

? Parámetros de Entrada (Body)

```
{
  "usuario": "string",
  "empleado": "string",
  "agencia": "string",
  "latitude": "string",
  "longitude": "string",
  "proceso": "string",
  "month": "int",
  "year": "int"
}
```

? Seguridad

- Acceso interno del sistema
- Requiere conexión válida a BD secundaria (`mysql2`)
- Valida datos contra API ERP mediante `dbErp` y `ServiceErp`

? Flujo del Servicio (Resumen Real)

1?? Obtiene parámetros principales del request:

- usuario
- empleado
- agencia
- lat/long
- proceso
- month / year

2?? Si el proceso es **descargaContratoReingreso**

- Ejecuta validación de renovación del contrato:

```
$this->validateProcessContractRenovation($request, $connection);
```

- Ejecuta validación de cambio de modalidad:

```
$this->validateProcessContractChangeModality($request,$connection);
```

3?? Si el proceso es **descargaContratoTrabajo**

a) Obtiene datos del empleado en el ERP

(Documento, sucursal, nombre y puesto)

```
SELECT passport_number, id_sucursal, nombre_completo, designation
FROM `tabEmployee`
```

b) Si el puesto contiene la palabra **CONDUCTOR**

El sistema crea un **usuario conductor** en un sistema externo:

```
POST EMPRESARIAL/api/crearUsuarioConductor
{
   documento: DNI,
   ter_id: id_sucursal,
   nombre: nombre_completo
}
```

Cualquier error en este paso retorna:

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

4?? Inserta el registro del proceso en `historial_procesos_app`

Campos insertados:

Campo	Descripción
usuario	Usuario que ejecuta el proceso
empleado	Código del empleado
agencia	ID de la agencia
latitude	Latitud
longitude	Longitud
proceso	Nombre del proceso ejecutado
fecha	Fecha y hora actual
estado	1 (registrado)
month	Mes
year	Año

5?? Devuelve respuesta exitosa

```
{ "valor": true, "msn": "Proceso registrado con éxito." }
```

? Respuestas

? **200 – Éxito**

```
{
  "valor": true,
  "msn": "Proceso registrado con éxito."
}
```

? Error al insertar registro

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

? Empleado no encontrado

```
{
  "valor": false,
  "msn": "No se encontró al empleado."
}
```

? Error de creación de usuario conductor

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

? Estructuras utilizadas

Tabla: historial_procesos_app


```
{
  "usuario": "string",
  "empleado": "string",
  "agencia": "string",
  "latitude": "string",
  "longitude": "string",
  "proceso": "string",
  "fecha": "datetime",
  "estado": 1,
  "month": "int",
  "year": "int"
}
```


ERP: Employee

Campos usados:


```
{
  "passport_number": "string",
  "id_sucursal": "string",
  "nombre_completo": "string",
  "designation": "string"
}
```

? Lógica en Pseudocódigo

```
leer usuario, empleado, proceso, month, year, lat, long

si proceso == "descargaContratoReingreso":
    validar renovación de contrato
    validar cambio de modalidad

si proceso == "descargaContratoTrabajo":
    traer datos del empleado
    si puesto incluye "CONDUCTOR":
        crear usuario conductor en sistema externo

insertar registro en historial_procesos_app

si inserción falla:
    retornar error

retornar éxito
```

Documento de Ingresos (Registrar Equipamiento (1)) - [store_equipament]

? Descripción

*Registra o actualiza las tallas de **equipamiento de EPP** (botas, polo, pantalón) para un postulante, validando previamente:*

- Que el documento exista en el sistema.
- Que las tallas enviadas sean válidas según los rangos permitidos.
- Que el postulante exista.
- Que ya tenga o no un registro previo en **Triaje de Postulante 2**, para decidir entre **crear (POST)** o **actualizar (PUT)**.

Es un servicio que integra:

- Validaciones locales.
- Consulta de postulantes.
- Inserción o actualización en el ERP vía **ServiceErp()**.

? Endpoint

POST `/store-equipament`

? Seguridad

Requiere autenticación interna contra el ERP mediante:

- `ServiceErp()`
- `searchPostulanteByDocument()`

No hace validaciones de token en el controlador; se maneja internamente por los métodos usados.

? Flujo del Servicio (explicación real)

1. **Lee los parámetros enviados en la petición**:
   - documento
   - botas
   - polo
   - pantalon
2. **Valida la estructura y tallas permitidas**:

Elemento	Tallas permitidas
botas	36–45
polo	S, M, L, XL
pantalón	S, M, L, XL

3. **Busca al postulante por documento**:
   Llama a:

   `searchPostulanteByDocument($documento)`
   Si no existe → retorna error.
4. **Busca si ya existe un registro de triaje activo en el ERP**:
   GET
   `/resource/Triaje de Postulante 2`
   con filtros:

   `documento = $documento`
   `docstatus != 2`
5. Dependiendo del resultado:
   - **No existe registro → crea uno (POST)**
   - **Sí existe → actualiza el existente (PUT)**
6. **Realiza el POST o PUT correspondiente** hacia el ERP.
7. **Si ERP devuelve error**, intenta decodificar `_server_messages` para obtener error real del ERP.
8. **Retorna respuesta final** indicando éxito o error.

? Request Body

Ejemplo:

```
{
  "documento": "12345678",
  "botas": 42,
  "polo": "M",
  "pantalon": "L"
}
```

? Response 200 – Ejemplos

?? Registro/Actualización exitosa


```
{
  "valor": true,
  "msn": "Equipamiento registrado con éxito"
}
```


? Documento no enviado


```
{
  "valor": false,
  "msn": "Falta enviar el documento"
}
```


? Talla inválida


```
{
  "valor": false,
  "msn": "Ingrese una talla de botas válida"
}
```


? Postulante no encontrado


```
{
  "valor": false,
  "msn": "No existe postulante con ese documento"
}
```


? Error del ERP

```
{
  "valor": false,
  "msn": "Ocurrió un error al registrar el triaje del postulante",
  "data": [
    "PUT",
    { ...body... },
    "URL del recurso usado"
  ]
}
```

? Posibles Errores del Servicio

Tipo	Ejemplo
Documento no enviado	`"Falta enviar el documento"`
Talla no válida	`"Ingrese una talla de botas válida"`
Postulante no encontrado	respuesta directa del servicio `searchPostulanteByDocument`
Error al crear/actualizar en ERP	Mensaje del `_server_messages` o mensaje genérico
Error inesperado	Devuelve el error capturado del ERP

? ERP: Documentos involucrados

Triaje de Postulante 2 (GET / POST / PUT)

Campos usados:

Campo	Tipo
documento	string
botas	int
polo	string
pantalon	string
name	string (clave para PUT)

? Lógica en Pseudocódigo

```
read document, botas, polo, pantalon

validate document
validate botas in [36..45]
validate polo in [S,M,L,XL]
validate pantalon in [S,M,L,XL]

postulante = searchPostulanteByDocument(document)

if postulante.invalid:
    return error

triaje = GET TriajeDePostulante2 where documento=document and docstatus != 2

if triaje exists:
    method = PUT
    url = recurso + "/" + triaje.name
else:
    method = POST
    url = recurso

body = {
    documento, botas, polo, pantalon
}

result = CALL ERP with method, body, url

if result.error:
    return readable ERP error

return success
```

Documento de Ingresos (Registrar Proceso de Descarga (1, 2,3,4,5)) - [registerProcessApp]

? Descripción

Registra en la tabla `historial_procesos_app` *todas las acciones (procesos) que el usuario realiza dentro del aplicativo móvil, como descargas de documentos, validación de contratos, marcaciones, entre otros.*

Antes de registrar el proceso:

1. **Valida si corresponde un proceso de Reingreso o Renovación de Contrato.**
2. **Procesa el registro especial para Conductores**, creando su usuario en el sistema empresarial cuando descarga su contrato.
3. **Guarda la información del evento** (geolocalización, mes, año, proceso, etc.) en la base de datos MySQL2.

Este servicio constituye una pieza clave para los módulos de:

- Marcación
- Documentos obligatorios
- Control de procesos del trabajador
- Contratos / renovaciones

? Endpoint

`POST /register-process-app`

? Request Body

```
{
  "usuario": "string",
  "empleado": "string",
  "agencia": "string",
  "latitude": "string|float",
  "longitude": "string|float",
  "proceso": "string",
  "month": "int",
  "year": "int"
}
```

? Seguridad

- Requiere **token válido** para consumir servicios internos del ERP.
- Conexión interna a MySQL2 para registrar procesos.
- Algunos procesos hacen integración con APIs externas (como Empresarial → creación de usuario conductor).

? Flujo del Servicio (resumen real)

1?? Obtiene los parámetros enviados desde la App

Usuario, empleado, agencia, geolocalización, proceso, mes y año.

2?? Si el proceso es **descargaContratoReingreso**

Ejecuta:

- **validateProcessContractRenovation()**
- **validateProcessContractChangeModality()**

Estas funciones validan:

- Si el empleado tiene renovación pendiente

- Si debe descargar documentos obligatorios
- Si existen cambios de modalidad pendientes

3?? Si el proceso es **descargaContratoTrabajo**

1. Obtiene datos del empleado desde ERP:
   - DNI (passport_number)
   - Sucursal
   - Nombre completo
   - Puesto (designation)
2. Si el puesto contiene `"CONDUCTOR"`, registra el usuario en el sistema empresarial:

`POST EMPRESARIAL/api/crearUsuarioConductor`

Enviando:

```
{
  "documento": "<passport_number>",
  "ter_id": "<id_sucursal>",
  "nombre": "<nombre_completo>"
}
```

4?? Inserta el proceso en la tabla MySQL:

Tabla: **historial_procesos_app**

Campos guardados:

- usuario
- empleado
- agencia
- latitude
- longitude
- proceso
- fecha
- estado
- month
- year

5?? Retorna la respuesta del proceso

? Response 200 – Ejemplo

```
{
  "valor": true,
  "msn": "Proceso registrado con éxito."
}
```

? Posibles Errores

1. No se puede registrar el proceso en MySQL

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

2. No se encuentra el empleado en el ERP

```
{
  "valor": false,
  "msn": "No se encontró al empleado."
}
```

3. Error creando usuario conductor

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

4. Error genérico

```
{
  "valor": false,
  "msn": "Error interno del servidor."
}
```

? Schemas – Objetos usados

? Employee (GET ERP)

Campos usados:


```
{
  "passport_number": "string",
  "id_sucursal": "string",
  "nombre_completo": "string",
  "designation": "string"
}
```


? Registro MySQL – historial_procesos_app


```
{
  "usuario": "string",
  "empleado": "string",
  "agencia": "string",
  "latitude": "string|float",
  "longitude": "string|float",
  "proceso": "string",
  "fecha": "datetime",
  "estado": 1,
  "month": "int",
  "year": "int"
}
```

? Lógica en pseudo-código

```
leer parametros del request

if proceso == "descargaContratoReingreso":
    validar renovación
    validar cambio modalidad

if proceso == "descargaContratoTrabajo":
    traer datos del empleado desde ERP

    if designation contiene "CONDUCTOR":
        crear usuario conductor via API Empresarial

insertar registro en historial_procesos_app

si insert falla:
    devolver error

return éxito
```

Contrato de Trabajo(Url Contrato de trabajo PDF (1)) - [printContractPerDesignation]

? Descripción

*Genera y descarga el **Contrato de Trabajo** correspondiente a un empleado, seleccionando de forma automática el **formato de contrato correcto** según:*

- Tipo de documento del empleado (DNI / PAS / CE)
- Modalidad laboral (Full time, Part time, Teletrabajo)
- Tipo de puesto (Ej. Conductor, Vigilante)
- Número de contratos previos
- Funciones asociadas al contrato (solo extranjeros)

El servicio determina internamente qué plantilla PDF debe usarse y devuelve el archivo final generado.

Es un servicio interno que consulta múltiples recursos del ERP:

- Employee
- Contrato de Trabajo
- Funciones del Contrato
- Otros contratos previos

? Endpoint

`GET /print-contract-per-designation/{employee}`

? Parámetros de ruta

Parámetro	Tipo	Obligatorio	Descripción
employee	string	Sí	Código único del empleado en ERP

? Seguridad

Requiere autenticación interna:

- Consulta ERP vía `dbErp()`
- Obtención de PDF vía `PDF::loadView()`

? Flujo del Servicio (Resumen Real)

1?? Validación del empleado

Consulta al ERP:

```
POST send-query-database
FROM tabEmployee
WHERE name = employee
SELECT fecha_de_ingreso_real
```

Si no existe → retorna error: "*No existe empleado*".

2?? Obtiene el contrato activo más reciente

Se buscan contratos con:

- `empleado = employee`
- `docstatus = 1`
- `fecha_de_ingreso_real = fecha_ingreso_empleado`
- Ordenado por creación descendente

```
POST send-query-database
FROM tabContrato de Trabajo
LEFT JOIN tabEmployee
```

Si no existe contrato → retorna "*No cuenta con Contrato*".

3?? Obtiene funciones asociadas (solo extranjeros)

Busca funciones del contrato:

```
FROM tabtabla_funcion_extranjero
WHERE parent = contrato.name
```

Estas funciones se agregan al contrato en el PDF.

4?? Determina el número de contratos previos

Solo aplica a empleados con documento **PAS** o **CE**.

Consulta cantidad de contratos registrados después del ingreso real.

Esto permite determinar si el contrato es:

- Primer contrato
- Segundo contrato
- Tercer contrato
- Etc. (usando arreglo ordinal: primero, segundo, tercero, ...)

5?? Selección automática de la plantilla PDF

En base a reglas:

Condición	Plantilla
modalidad_de_trabajo = Teletrabajo	contrato_teletrabajo
labor contiene "CONDUCTOR"	contrato_conductor_interprovincial
labor = "VIGILANTE"	contrato_full_time_vigilante
tipo_doc PAS o CE, **primer contrato**	contrato_extranjero
tipo_doc PAS o CE, **más de 1 contrato**	contrato_extranjero_mas_de_segundo_contrato
tipo_contrato = PART TIME, labor=CALL CENTER ATC	contrato_call_center_part_time
tipo_contrato = PART TIME	contrato_part_time
tipo_contrato = FULL TIME	contrato_full_time

El método retorna el PDF con:

```
PDF::loadView(...)->download();
```

? Response – Archivo PDF

Retorna directamente la descarga del contrato correspondiente.

Si ocurre algún error, retorna un JSON:

?? Empleado válido, pero sin contrato

```
{
  "valor": false,
  "msn": "No cuenta con Contrato"
}
```

? Empleado no encontrado

```
{
  "valor": false,
  "msn": "No existe empleado"
}
```

? Posibles Errores

Error	Descripción
Empleado no existe	No se encontró en tabEmployee
No tiene contratos vigentes	No existe Contrato de Trabajo activo
Error DB ERP	Respuesta inesperada en dbErp
Plantilla no encontrada	Error al cargar vista PDF (muy raro)

? Tablas y Campos Utilizados

Employee (GET)

Campos usados:

```
{
  "fecha_de_ingreso_real": "date"
}
```

Contrato de Trabajo

```
{
  "name": "string",
  "fecha_de_ingreso_real": "date",
  "empleado": "string",
  "labor": "string",
  "tipo_de_documento": "string",
  "modalidad_de_trabajo": "string",
  "tipo_de_contrato": "string"
}
```

Funciones del Contrato (Extranjeros)

```
{
  "funcion": "string"
}
```

? Lógica en Pseudocódigo

```
employeeData = GET Employee where name = employee
if not exists: return error

contract = GET Contrato where empleado=employee and activo
if not exists: return error

funciones = GET Funciones where parent=contract

if doc PAS/CE:
    countPrevious = GET Contratos after ingreso real
    determinar ordinal

seleccionar plantilla PDF según reglas

return descargar PDF
```