API Staff Attendance — Clientes v2

Versión: 2.9.0 · **Base URL:** `https://api.staffattendance.cl` · **Ambiente de prueba:** `https://test.api.staffattendance.cl` · **Swagger UI:** `/swagger-ui/index.html`

Esta API permite a los clientes de **Staff Attendance** consultar y mantener la información de empleados, locales, jornadas, marcas, ausentismos y horas extras de forma programática. Los endpoints están auto-documentados vía Swagger UI y esta guía funciona como referencia narrativa complementaria.

Tabla de contenidos

1. Introducción
    - 1.1 Autenticación
    - 1.1.1 Crear y administrar credenciales
    - 1.2 Scopes y restricciones
    - 1.3 Códigos HTTP
    - 1.4 Formato de respuesta común
    - 1.5 Límites y cuotas
    - 1.6 Ambientes de producción y de prueba
2. Resumen de endpoints
3. Endpoints de lectura (GET)
    - getEmpleados
    - getCargos
    - getLocales
    - getTipoAusentismo
    - getAusentismo
    - getHorasExtras
    - getHorasExtrasManuales
    - getJornadasPorDia

    - getMarcas
    - getReporteCumplimiento
    - getEstadoLectores
    - getReporteDomingosAdicionales
    - getCambiosMarcas
    - getMarcasPendientes y postConfirmarMarcas
4. Endpoints de escritura (POST y PUT)
    - postMarca
    - postAusentismo / putAusentismo
    - postCargo / putCargo
    - postEmpleado / putEmpleado
    - postLocal / putLocal
    - postJornadaPorDia / putJornadaPorDia
    - postTipoAusentismo / putTipoAusentismo
    - postHoraExtraManual / putHoraExtraManual
    - Corrección de marcas con aceptación del trabajador
5. Endpoints de eliminación (DELETE)
    - deleteLocal
    - deleteEmpleado
    - deleteJornadaPorDia
    - deleteCargo
    - deleteAusentismo
    - deleteTipoAusentismo
    - deleteHoraExtra
    - deleteHoraExtraManual

---

1. Introducción

1.1 Autenticación

Todos los endpoints de `/v2/client/**` usan **OAuth2 con flujo** `client_credentials`. Se necesita un `client_id` y un `client_secret` para obtener un `access_token` de tipo *Bearer*. El administrador del cliente crea esas credenciales directamente en Staff Attendance (ver 1.1.1).

Endpoint de token:

```
POST https://auth.staffattendance.cl/realms/staffattendance/protocol/openid-connect/token
```

Parámetros (`application/x-www-form-urlencoded`):

Campo	Valor
`client_id`	el de la credencial (por ejemplo `demo-erp`)
`client_secret`	el que se mostró al crear o rotar la credencial
`grant_type`	`client_credentials`

Ejemplo cURL:

```
curl -X POST 'https://auth.staffattendance.cl/realms/staffattendance/protocol/openid-
connect/token' \
  -H 'Content-Type: application/x-www-form-urlencoded' \
  --data-urlencode 'client_id=demo-erp' \
  --data-urlencode 'client_secret=<client-secret>' \
  --data-urlencode 'grant_type=client_credentials'
```

La respuesta incluye un `access_token` que debe enviarse en cada llamada a la API como header:

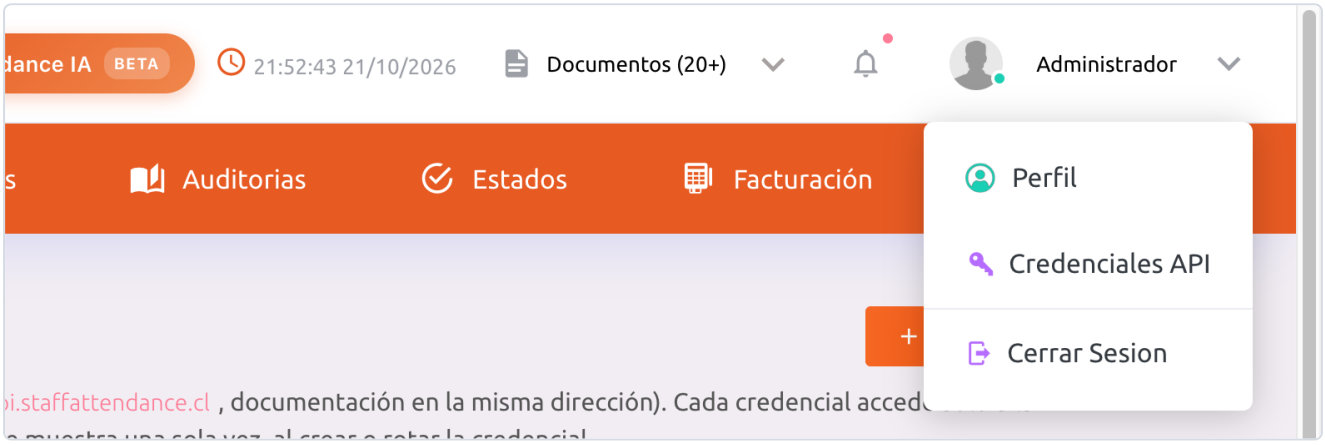
```
Authorization: Bearer <access_token>
```

> **Importante:** la credencial identifica al cliente. Un token solo puede acceder a los datos del cliente para el que se creó la credencial; no es necesario indicar el cliente en las llamadas.

1.1.1 Crear y administrar credenciales

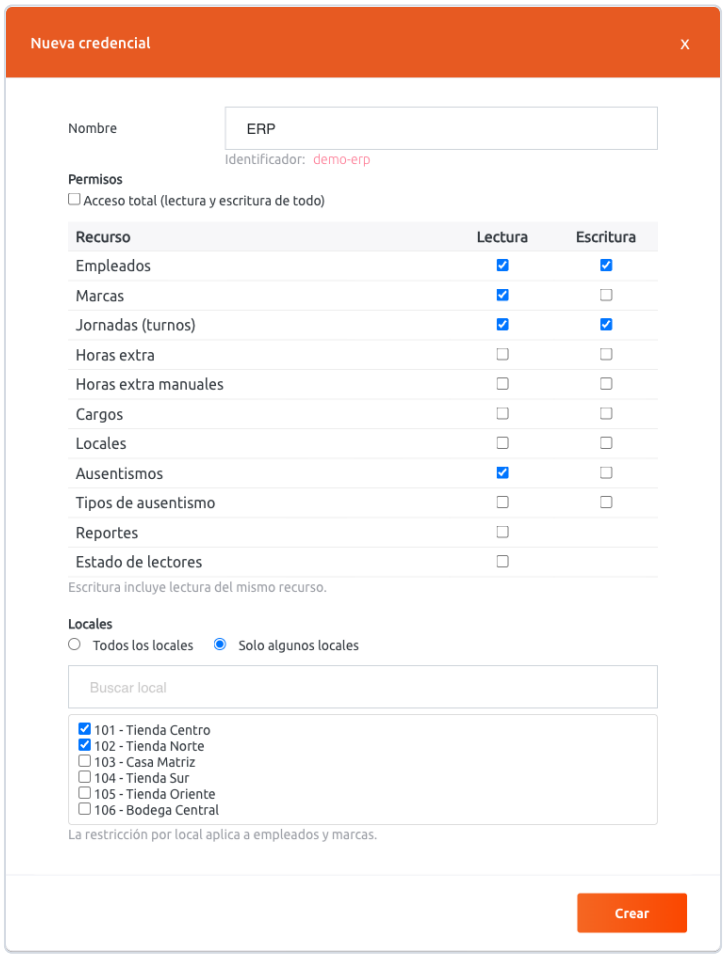
Las credenciales las crea un usuario **administrador** del cliente en el portal de Staff Attendance (`https://app.staffattendance.cl`). Cada cliente puede tener varias, una por sistema que se integre, cada una con sus propios permisos.

1. Abrir Credenciales API. Con el cliente seleccionado en el selector superior, abrir el menú de la foto de perfil (esquina superior derecha) y elegir **Credenciales API**.



2. Crear la credencial. Presionar **Nueva credencial** e indicar:

- **Nombre:** identifica el sistema que la usará (por ejemplo *ERP*). El `client_id` queda como `<cliente>-<nombre>`, por ejemplo `demo-erp`.
- **Permisos:** *Acceso total*, o lectura y escritura por recurso (ver 1.2). Escritura incluye lectura del mismo recurso.
- **Locales:** todos, o solo algunos. Con locales acotados aplica la restricción por tienda de la sección 1.2.



3. Copiar el secret. Al crear la credencial se muestran el `client_id` y el `client_secret`. **El secret se muestra una sola vez**: hay que copiarlo y guardarlo de forma segura en ese momento.



4. Administrar. La lista muestra cada credencial con sus permisos, locales, quién la creó o rotó y su estado. Desde la columna *Opciones*:

- **Rotar** genera un `client_secret` nuevo; **el anterior deja de funcionar de inmediato**. Para cambiar el secret sin interrumpir la integración, crear una credencial nueva, actualizar el sistema y luego revocar la anterior.
- **Revocar** deshabilita la credencial de forma definitiva: deja de entregar tokens y no se puede reactivar.



Las credenciales entregadas anteriormente por GPI Software siguen vigentes y aparecen en la lista con el nombre *Principal*.

1.2 Scopes y restricciones

Cada credencial OAuth recibe un conjunto de **scopes** que determinan a qué endpoints puede acceder. La jerarquía es:

Scope	Permite
`client`	Acceso completo a `/v2/client/**`.
`client.read`	Todos los endpoints de lectura (GET).
`client.write`	Todos los endpoints de escritura (POST, PUT y DELETE).
`client.<recurso>.read`	Sólo los GET del recurso indicado (ej. `client.empleados.read`).
`client.<recurso>.write`	POST, PUT y DELETE del recurso indicado. También permite GET del mismo recurso.

La escritura de marcas (`client.marcas.insertar`: `postMarca` y `deleteNarca`) es especial: la habilita GPI Software y no la incluye ningún otro scope; en `marcas` el cliente solo puede elegir lectura (ver postMarca).

`getCambiosMarcas`, `getMarcasPendientes` y `postConfirmarMarcas` usan la lectura de `marcas` (`client.marcas.read`, `client.read` o `client`): confirmar solo mueve el cursor de la propia credencial, no escribe marcas.

Recursos disponibles para los scopes granulares: `empleados`, `marcas`, `jornadas`, `horasExtras`, `horasExtrasManuales`, `cargos`, `locales`, `ausentismos`, `tipoAusentismos`, `correccionMarcas`, `reportes` y `lectores` (estos dos últimos sólo lectura).

Los scopes corresponden a los permisos elegidos al crear la credencial: *Acceso total* es `client` y cada casilla de lectura o escritura es un `client.<recurso>.read` o `.write`. **No es necesario** pasar el parámetro `scope` en la solicitud de token: los scopes asignados se incluyen automáticamente, y pedir uno no asignado hace que el servidor de identidad rechace la solicitud (`invalid_scope`). Si la credencial intenta acceder a un endpoint fuera de su scope, la API retorna `403 Forbidden`.

Restricción opcional por tienda

Una credencial puede estar acotada a un subconjunto de tiendas dentro del cliente (opción *Solo algunos locales* al crearla). Cuando aplica:

- `getEmpleados` retorna sólo empleados de tiendas autorizadas; un `rut` cuyo store no esté en la lista devuelve `401 Unauthorized`.
- `getMarcas` filtra a marcas de empleados de las tiendas autorizadas.
- `postEmpleado` / `putEmpleado` rechazan crear o modificar empleados en tiendas no autorizadas, e impiden mover un empleado existente fuera del scope autorizado.
- `deleteEmpleado` devuelve `403 Forbidden` si el empleado pertenece a una tienda fuera del scope.

Si la credencial no tiene esta restricción configurada, opera sobre todas las tiendas del cliente (comportamiento histórico).

1.3 Códigos HTTP

Código	Significado
`200 OK`	La petición se ejecutó correctamente.
`400 Bad Request`	Parámetros inválidos (fecha mal formateada, RUT fuera de rango, tipo de ausentismo inexistente, etc.).
`401 Unauthorized`	Token ausente, malformado o sin el `client_id` requerido.
`409 Conflict`	El recurso tiene dependencias que impiden borrarlo (por ejemplo, un local/cargo con empleados asociados).
`413 Payload Too Large`	Rango de fechas consultado mayor a 60 días.
`429 Too Many Requests`	Se superó el rate limit (máximo una consulta GET cada 2 segundos).
`403 Forbidden`	La credencial no tiene el scope necesario para el endpoint o pidió un local fuera de su restricción de tiendas.
`500 Internal Server Error`	Error no controlado en el servidor. Verificar los detalles en el body.

1.4 Formato de respuesta común

Los endpoints de **escritura y eliminación** devuelven un objeto `ErrorDetails`:

```
{
  "timestamp": "2024-10-15T14:36:52.884+00:00",
  "message": "success",
  "details": "ID=demo_13163137_2024-10-20 OK,"
}
```

- `message`: `success`, `success with warnings` o `error`.
- `details`: string con los IDs procesados o los mensajes de error, separados por coma.

Los endpoints de **lectura** devuelven directamente el arreglo JSON de resultados.

1.5 Límites y cuotas

- **Rango de fechas máximo:** 60 días por consulta (endpoints con `fecha_inicio` y `fecha_fin`). Si se excede, se retorna `413`.
- **Rate limit GET:** máximo 1 consulta cada 2 segundos por cliente. Si se excede, se retorna `429`.
- **Batch en POST/PUT:** los endpoints de escritura aceptan un arreglo JSON con N elementos. Se procesan secuencialmente; los errores individuales no detienen el procesamiento del resto del batch.

1.6 Ambientes de producción y de prueba

La API se ofrece en dos ambientes con los mismos endpoints, parámetros y respuestas:

Ambiente	URL base	Uso
Producción	`https://api.staffattendance.cl`	Operación real. Los cambios se reflejan de inmediato en Staff Attendance.
Prueba	`https://test.api.staffattendance.cl`	Desarrollo y validación de integraciones, sin afectar la operación real.

- **Mismas credenciales:** el `client_id` y el `client_secret` sirven en ambos ambientes, y el token se obtiene del mismo endpoint de la sección 1.1.
- **Datos:** el ambiente de prueba tiene una copia de los datos de producción a una fecha determinada. No se sincroniza en línea, así que los cambios hechos en un ambiente no aparecen en el otro.
- **Swagger UI:** disponible en ambos, en `/swagger-ui/index.html` de cada URL base.
- **Recomendación:** desarrollar y validar la integración en el ambiente de prueba, y al terminar cambiar solo la URL base a la de producción.

> **Importante:** el ambiente de prueba puede ser actualizado con una nueva copia de producción, lo que reemplaza los datos cargados en él. No se debe usar para guardar información que se necesite conservar.

---

2. Resumen de endpoints

Método	Path	Descripción
`GET`	`/v2/client/getEmpleados`	Listar empleados o uno específico por RUT.
`GET`	`/v2/client/getCargos`	Listar cargos.
`GET`	`/v2/client/getLocales`	Listar locales activos.
`GET`	`/v2/client/getTipoAusentismo`	Listar tipos de ausentismo.
`GET`	`/v2/client/getAusentismo`	Listar ausentismos en un rango de fechas.
`GET`	`/v2/client/getHorasExtras`	Listar horas extras calculadas en un rango.

Método	Path	Descripción
`GET`	`/v2/client/getHorasExtrasManuales`	Listar horas extras ingresadas por API.
`GET`	`/v2/client/getJornadasPorDia`	Listar jornadas (turnos) diarios.
`GET`	`/v2/client/getMarcas`	Listar marcas (entrada/salida) en un rango.
`GET`	`/v2/client/getCambiosMarcas`	**Nuevo v2.9.0.** Solo lo que cambió en las marcas desde la llamada anterior (cursor del cliente).
`GET`	`/v2/client/getMarcasPendientes`	**Nuevo v2.9.0.** Igual, con el cursor guardado en Staff Attendance por credencial.
`POST`	`/v2/client/postConfirmarMarcas`	**Nuevo v2.9.0.** Confirma lo recibido en `getMarcasPendientes` (avanza el cursor guardado).
`GET`	`/v2/client/getReporteCumplimiento`	Reporte de cumplimiento calculado por empleado y día.
`GET`	`/v2/client/getReporteDomingosAdicionales`	Domingos y sábados adicionales (Ley 20.823) por empleado, con año móvil.
`GET`	`/v2/client/getEstadoLectores`	Estado de conexión de los lectores (huelleros).
`POST`	`/v2/client/postMarca`	Insertar marcas válidas originadas por el trabajador (batch).
`POST`	`/v2/client/postSolicitudCorreccionMarca`	Corregir marcas con aceptación del trabajador (batch).
`GET`	`/v2/client/getSolicitudesCorreccionMarca`	Estado de las correcciones y de las solicitudes de los trabajadores.
`POST`	`/v2/client/postResolverSolicitudCorreccionMarca`	Aprobar o rechazar la solicitud de corrección de un trabajador.
`POST` / `PUT`	`/v2/client/{post,put}Ausentismo`	Upsert de ausentismos (batch).
`POST` / `PUT`	`/v2/client/{post,put}Cargo`	Upsert de cargos (batch).

Método	Path	Descripción
`POST` / `PUT`	`/v2/client/{post,put}Empleado`	Upsert de empleados (batch).
`POST` / `PUT`	`/v2/client/{post,put}Local`	Upsert de locales (batch).
`POST` / `PUT`	`/v2/client/{post,put}JornadaPorDia`	Upsert de jornadas diarias (batch).
`POST` / `PUT`	`/v2/client/{post,put}TipoAusentismo`	Upsert de tipos de ausentismo (batch).
`POST` / `PUT`	`/v2/client/{post,put}HoraExtraManual`	Upsert de horas extras manuales (batch).
`DELETE`	`/v2/client/deleteLocal`	Borrar local (valida dependencias).
`DELETE`	`/v2/client/deleteEmpleado`	Desactivar empleado (soft-delete).
`DELETE`	`/v2/client/deleteJornadaPorDia`	Borrar jornada de un RUT en un día.
`DELETE`	`/v2/client/deleteCargo`	Borrar cargo (valida dependencias).
`DELETE`	`/v2/client/deleteAusentismo`	Borrar ausentismo.
`DELETE`	`/v2/client/deleteTipoAusentismo`	Borrar tipo de ausentismo.
`DELETE`	`/v2/client/deleteHoraExtra`	Borrar hora extra (calculada).
`DELETE`	`/v2/client/deleteHoraExtraManual`	Borrar hora extra manual.

Nota sobre POST vs PUT: ambos verbos comparten el mismo body, pero la semántica difiere: **POST** es *strict-create* y rechaza el registro con `400` si la clave ya existe (mensaje "X ya existe; usar PUT para actualizar"); **PUT** es *upsert*, inserta si no existe y actualiza si existe. Se aplica por registro dentro del batch — un POST con varios items puede devolver `success with warnings` mezclando creados nuevos y errores de los que ya existían.

Nuevo en v2.9.0 — campos omitidos en PUT: al actualizar un registro existente, **un campo omitido (o `null`) conserva el valor que ya tiene** en Staff Attendance; solo se modifican los campos que vienen en el JSON. Excepciones: `finContrato` de `putEmpleado` (omitido = indefinido) y la colación de `putJornadaPorDia` (omitida = sin colación). Para quitar un valor hay que enviarlo explícitamente (por ejemplo `false`, `0` o `""`, según el campo). Al crear un registro nuevo, los campos omitidos toman su valor por defecto.

3. Endpoints de lectura (GET)

3.1 getEmpleados

Lista los empleados del cliente (activos e inactivos). Con `rut=0` devuelve todos; con un RUT específico, devuelve ese empleado.

`GET /v2/client/getEmpleados`

Parámetros:

Parámetro	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT del empleado (sin puntos ni DV). `0` para traer todos.
`codigoLocal`	string	No	Filtra el resultado a empleados de la tienda indicada (formato corto, sin prefijo). Si la credencial está acotada a un set de tiendas y se pide un código fuera de ese set, retorna `403`.

Respuestas:

- `200 OK`: arreglo de empleados (puede ser vacío si `rut` y `codigoLocal` no coinciden).
- `401 Unauthorized`: el RUT consultado no pertenece al cliente autenticado o está fuera del scope de tiendas autorizado.
- `403 Forbidden`: el `codigoLocal` solicitado está fuera del scope de tiendas autorizado para la credencial.

Ejemplo:

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getEmpleados?rut=0' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'accept: application/json'
```

```
[
  {
    "activo": true,
    "email": "prueba@prueba.cl",
    "horasContrato": 20,
    "nombre": "VENDEDOR PART TIME",
    "rut": 40404040,
    "codigoCargo": "1",
    "codigoLocal": "1",
    "codigoLocalDestino": "2",
    "inicioContrato": "01/01/2022",
    "finContrato": "01/01/3000",
    "articulo22": false,
    "marcaMovil": false,
    "marcaWeb": false,
    "marcaQR": false,
    "codigo": "",
    "tarjeta": "",
    "codigoTurno": ""
  }
]
```


> **Nota:** `finContrato = "01/01/3000"` significa contrato indefinido. El campo `password` **no se expone** en la respuesta. `codigoLocalDestino` es opcional: representa una tienda alternativa de asignación temporal y sólo aparece si está configurado para el empleado.

3.2 getCargos

Lista los cargos del cliente.

`GET /v2/client/getCargos`

Sin parámetros.

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getCargos' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```


```
[
  { "codigoCargo": "1", "nombreCargo": "JEFE DE TIENDA" },
  { "codigoCargo": "2", "nombreCargo": "VENDEDOR" }
]
```

3.3 getLocales

Lista los locales activos del cliente.

`GET /v2/client/getLocales`

Sin parámetros.

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getLocales' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
[
  {
    "codigoLocal": "2",
    "nombreLocal": "CASA MATRIZ",
    "direccionLocal": "DIRECCION CASA MATRIZ",
    "manangerRut": 0,
    "supervisorRut": 12343875,
    "rutAlternativo": 0,
    "razonSocialAlternativo": "",
    "idTimezone": "Santiago-Chile (GMT -04:00)",
    "managersAdicionales": [],
    "supervisoresAdicionales": [15123456, 16234567]
  }
]
```

> **Nuevo en v2.1:** se incluye `idTimezone` en la respuesta.
>
> **Nuevo en v2.6:** un local puede tener más de un manager y más de un supervisor. `manangerRut` y `supervisorRut` siguen siendo el manager y el supervisor **principal** (sin cambios). `managersAdicionales` y `supervisoresAdicionales` listan los RUTs (sin DV) de los responsables adicionales al principal; son un arreglo vacío cuando el local no tiene adicionales.

3.4 getTipoAusentismo

Lista los tipos de ausentismo definidos para el cliente.

`GET /v2/client/getTipoAusentismo`

Sin parámetros.

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getTipoAusentismo' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
[
  { "codigoAusentismo": "PA", "descripcionAusentismo": "PERMISO ADMINISTRATIVO" },
  { "codigoAusentismo": "VI", "descripcionAusentismo": "VIAJE INTERNACIONAL" },
  { "codigoAusentismo": "L",  "descripcionAusentismo": "LICENCIA" }
]
```

3.5 getAusentismo

Lista ausentismos (permisos, licencias, etc.) en un rango de fechas.

`GET /v2/client/getAusentismo`

Parámetro	Tipo	Requerido	Descripción
`fecha_inicio`	string `DD/MM/YYYY`	Sí	Inicio del rango.
`fecha_fin`	string `DD/MM/YYYY`	Sí	Fin del rango (incluye el día completo).

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getAusentismo?
fecha_inicio=01/10/2024&fecha_fin=31/10/2024' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
[
  {
    "rut": 13163137,
    "fecha": "20/10/2024",
    "tipo_permiso": "L",
    "observaciones": "licencia de prueba"
  }
]
```

3.6 getHorasExtras

Lista horas extras calculadas por el sistema en un rango de fechas.

`GET /v2/client/getHorasExtras`

Parámetro	Tipo	Requerido	Descripción
`fecha_inicio`	string `DD/MM/YYYY`	Sí	Inicio del rango.
`fecha_fin`	string `DD/MM/YYYY`	Sí	Fin del rango (incluye día completo).

Parámetro	Tipo	Requerido	Descripción
`filtro_aprobacion_entrada`	string	No	`si` , `no` , `si,no` .
`filtro_aprobacion_salida`	string	No	`si` , `no` , `si,no` .

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getHorasExtras?
fecha_inicio=01/04/2023&fecha_fin=30/04/2023&filtro_aprobacion_entrada=si,no&filtro_aprobac
ion_salida=si,no' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
[
  {
    "rut": 15789725,
    "dia": "27/04/2023",
    "minutos_extra_entrada": 34,
    "minutos_extra_salida": 124,
    "estado_aprobacion_entrada": "PENDIENTE",
    "estado_aprobacion_salida": "PENDIENTE"
  }
]
```

> **Estados posibles:** `PENDIENTE` , `AUTORIZADO` , `RECHAZADO` .

3.7 getHorasExtrasManuales

Lista horas extras ingresadas por API (no calculadas).

`GET /v2/client/getHorasExtrasManuales`

Parámetro	Tipo	Requerido	Descripción
`fecha_inicio`	string `DD/MM/YYYY`	Sí	Inicio del rango.
`fecha_fin`	string `DD/MM/YYYY`	Sí	Fin del rango (incluye día completo).

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getHorasExtrasManuales?
fecha_inicio=01/10/2024&fecha_fin=30/10/2024' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
[
  {
    "rut": 13163137,
    "fecha": "19/10/2024",
    "horaExtraMinutos": 30,
    "observaciones": "ingresada desde api"
  }
]
```

3.8 getJornadasPorDia

Lista las jornadas (turnos) asignadas a empleados en un rango de fechas.

`GET /v2/client/getJornadasPorDia`

Parámetro	Tipo	Requerido	Descripción
`fecha_inicio`	string `DD/MM/YYYY`	Sí	Inicio del rango.
`fecha_fin`	string `DD/MM/YYYY`	Sí	Fin del rango (incluye día completo).

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getJornadasPorDia?
fecha_inicio=01/10/2024&fecha_fin=30/10/2024' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
[
  {
    "diaJornada": "17/10/2024",
    "horaInicio": "08:00",
    "horaTermino": "18:00",
    "colacionInicio": "12:00",
    "colacionTermino": "13:00",
    "rut": 20202020,
    "solicitante": "T",
    "fechaAsignacion": "17/10/2024"
  }
]
```

> **solicitante:** `T` = Trabajador, `E` = Empleador.

3.9 getMarcas

Lista las marcas (entrada/salida) registradas en un rango.

`GET /v2/client/getMarcas`

Parámetro	Tipo	Requerido	Descripción
`fecha_inicio`	string `DD/MM/YYYY`	Sí	Inicio del rango.
`fecha_fin`	string `DD/MM/YYYY`	Sí	Fin del rango (incluye día completo hasta 23:59:59).
`codigoLocal`	string	No	Filtra a marcas de empleados de la tienda indicada (formato corto, sin prefijo). Si la credencial está acotada a tiendas y se pide un código fuera de ese set, retorna `403`.

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getMarcas?
fecha_inicio=14/10/2024&fecha_fin=14/10/2024' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```


```
[
  {
    "idAttendance": "13163137_2024_10_14_102000000",
    "rut": "13163137-5",
    "fecha": "14/10/2024",
    "hora": "17:20:00",
    "sentido": "Salida",
    "crc": "1119096083"
  }
]
```


> **Nota:** solo se devuelven marcas **activas y autorizadas**. El `rut` viene con dígito verificador.

3.10 getReporteCumplimiento

Reporte consolidado por empleado y día: jornada programada, marcas, atrasos, horas trabajadas, etc.

`GET /v2/client/getReporteCumplimiento`

Parámetro	Tipo	Requerido	Descripción
`fecha_inicio`	string `DD/MM/YYYY`	Sí	Inicio del rango.

Parámetro	Tipo	Requerido	Descripción
`fecha_fin`	string `DD/MM/YYYY`	Sí	Fin del rango.

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getReporteCumplimiento?
fecha_inicio=01/04/2023&fecha_fin=30/04/2023' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```


```
[
  {
    "fecha": "01/04/2023",
    "tienda": "TALCA",
    "origen_marca": "LECTOR TALCA",
    "rut": "18417359-0",
    "nombre": "NICOL CONSTANZA LOPEZ NUÑEZ",
    "dia": "Sabado",
    "jornada": "10:00 - 20:30 (14:15 - 15:15)",
    "marca_entrada": "09:59",
    "marca_salida": "20:38",
    "marca_inicio_colacion": "15:50",
    "marca_fin_colacion": "16:16",
    "asistencia": "Si",
    "horas_trabajadas": "09:30",
    "atraso": "00:00",
    "salida_anticipada": "00:00",
    "descuento": "--:--",
    "horas_fuera_de_turno": "00:09",
    "horas_extra_aprobadas": "00:00"
  }
]
```

3.11 getEstadoLectores

Estado de conexión de los lectores (huelleros) del cliente. Devuelve todos los lectores del cliente, activos e inactivos.

`GET /v2/client/getEstadoLectores`

Sin parámetros.

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getEstadoLectores' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

Campos de cada lector en la respuesta:

Campo	Tipo	Descripción
`idLector`	string	Identificador del lector (número de serie).
`nombre`	string	Nombre del lector.
`codigoLocal`	string	Código del local al que pertenece el lector.
`ubicacion`	string	Nombre del local (ubicación) del lector.
`ultimaConexion`	string	Última conexión en hora local, formato `HH:mm - dd/MM/yyyy`. Vacío si nunca se conectó.
`ultimaConexionMs`	number	Última conexión en epoch millis (GMT). `0` si nunca se conectó.
`estado`	string	`En línea`, `Desconectado` o `Inactivo`.
`version`	string	Versión de firmware del lector.
`huellas`	number	Huellas enroladas (sólo lectores de red; `null` en otros).
`usuarios`	number	Usuarios en el lector (sólo lectores de red; `null` en otros).
`transacciones`	number	Marcas registradas en el lector (sólo lectores de red; `null` en otros).

Estado: un lector aparece como `En línea` si reportó conexión dentro de los últimos minutos; `Desconectado` si no reporta hace tiempo; `Inactivo` si está dado de baja.


```
[
  {
    "idLector": "226381B31777",
    "nombre": "Lector Entrada Principal",
    "codigoLocal": "0035",
    "ubicacion": "TIENDA CENTRO",
    "ultimaConexion": "14:32 - 19/06/2026",
    "ultimaConexionMs": 1750430340000,
    "estado": "En línea",
    "version": "Ver 6.60 Jul 25 2018",
    "huellas": 487,
    "usuarios": 156,
    "transacciones": 12043
  }
]
```

3.12 getReporteDomingosAdicionales

Domingos y sábados adicionales de descanso (Ley 20.823 / Art. 38 bis del Código del Trabajo) por empleado. Entrega los conteos del rango consultado y los totales del **año móvil** vigente de cada trabajador (los 12 meses que parten en el aniversario de su contrato; para contratos anteriores al 07/04/2015, la anualidad parte ese día). Los valores son idénticos a los del reporte "Domingos adicionales" de la aplicación web.

Reglas de cálculo: cada domingo del año móvil se clasifica como trabajado (con marcación), libre por ley (los 2 primeros no trabajados de cada mes, Art. 38) o **adicional** (los siguientes no trabajados, que consumen el cupo anual de 7). Domingos en feriado legal o con licencia/permiso no se computan. Un **sábado adicional** es un sábado libre contiguo a un domingo libre (hasta 3 al año pueden reemplazar domingos adicionales). Se excluyen trabajadores con jornada de 20 horas semanales o menos y personal Art. 22. El día actual nunca se computa (el cálculo llega hasta ayer).

> **Baseline pre-implantación:** si la cuenta comenzó a usar Staff Attendance con el año móvil de sus trabajadores ya iniciado, los acumulados previos informados por la empresa se suman a los totales (`baselineDomingos` / `baselineSabados` , con la fecha de corte en `baselineCutoff`). Si no aplica, esos campos llegan en `0` y `""` .

`GET /v2/client/getReporteDomingosAdicionales`

Parámetro	Tipo	Requerido	Descripción
`fecha_inicio`	string `DD/MM/YYYY`	No	Inicio del rango. Si se omite, últimos 12 meses.
`fecha_fin`	string `DD/MM/YYYY`	No	Fin del rango. Si se omite, hoy. Rango máximo: 2 años.
`rut`	number	No	Rut sin dígito verificador. `0` o ausente = todos los empleados.
`codigoLocal`	string	No	Código corto del local. Si la credencial tiene restricción de locales y pide uno fuera de su alcance, responde `403` .

```
curl -X GET 'https://api.staffattendance.cl/v2/client/getReporteDomingosAdicionales?
fecha_inicio=01/01/2026&fecha_fin=30/06/2026' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

Campos de cada empleado en la respuesta:

Campo	Tipo	Descripción
`rut`	string	Rut con dígito verificador, ej. `12345678-9`.
`colaborador`	string	Nombre del empleado.
`local`	string	Nombre del local.
`cargo`	string	Nombre del cargo.
`domingosAdicionales`	number	Domingos adicionales dentro del rango consultado.
`sabadosAdicionales`	number	Sábados adicionales dentro del rango consultado.
`totalAdicionales`	number	Suma de los dos anteriores.
`totalLegal`	number	Domingos + hasta 3 sábados (tope legal de reemplazo).
`fechasDomingos`	string[]	Fechas `DD/MM/YYYY` de los domingos adicionales del rango.
`fechasSabados`	string[]	Fechas `DD/MM/YYYY` de los sábados adicionales del rango.
`inicioAnoMovil` / `finAnoMovil`	string	Año móvil vigente del trabajador (`Sin fecha` si no tiene fecha de contrato).
`totalDomingosAdicionalesAnioMovil`	number	Domingos adicionales acumulados en el año móvil (incluye baseline).
`totalDomingosTrabajadosAnioMovil`	number	Domingos trabajados en el año móvil.
`totalDomingosLibresLeyAnioMovil`	number	Domingos libres por ley (Art. 38) en el año móvil.
`totalDomingosEnPermisoAnioMovil`	number	Domingos con licencia/permiso (no computados).
`totalDomingosEnVacacionesAnioMovil`	number	Domingos en feriado legal (no computados).
`totalSabadosAnioMovil`	number	Sábados adicionales del año móvil (incluye baseline).
`baselineDomingos` / `baselineSabados`	number	Acumulados pre-implantación informados por la empresa (`0` si no aplica).
`baselineCutoff`	string	Fecha de corte del baseline (`""` si no aplica).

```
[
  {
    "rut": "12345678-9",
    "colaborador": "MARIA PEREZ SOTO",
    "local": "TIENDA CENTRO",
    "cargo": "VENDEDOR FULL TIME",
    "domingosAdicionales": 4,
    "sabadosAdicionales": 2,
    "totalAdicionales": 6,
    "totalLegal": 6,
    "fechasDomingos": ["22/03/2026", "26/04/2026", "24/05/2026"],
    "fechasSabados": ["21/03/2026", "16/05/2026"],
    "inicioAnoMovil": "02/10/2025",
    "finAnoMovil": "01/10/2026",
    "totalDomingosAdicionalesAnioMovil": 4,
    "totalDomingosTrabajadosAnioMovil": 12,
    "totalDomingosLibresLeyAnioMovil": 13,
    "totalDomingosEnPermisoAnioMovil": 0,
    "totalDomingosEnVacacionesAnioMovil": 1,
    "totalSabadosAnioMovil": 2,
    "baselineDomingos": 1,
    "baselineSabados": 0,
    "baselineCutoff": "31/01/2026"
  }
]
```

3.13 getCambiosMarcas

> **Nuevo en v2.9.0.** Para integraciones que consultan las marcas varias veces al día: en vez de pedir el mismo rango una y otra vez con `getMarcas`, se pide **solo lo que cambió** desde la llamada anterior.

Devuelve, en orden, las **altas, modificaciones y anulaciones** de marcas ocurridas después del `cursor`. Cada respuesta trae un `cursor` nuevo para la llamada siguiente.

`GET /v2/client/getCambiosMarcas`

Parámetro	Tipo	Requerido	Descripción
`cursor`	string	Sí, salvo la primera vez	El `cursor` que devolvió la llamada anterior (opaco: no interpretarlo ni modificarlo).
`desde`	string `DD/MM/YYYY`	Solo la primera vez	Sin `cursor`: cambios ocurridos desde ese día (hora de Chile). Máximo 30 días atrás.

Parámetro	Tipo	Requerido	Descripción
`limite`	entero	No	Máximo de cambios por respuesta, de 1 a 5000 (por defecto 1000).
`codigoLocal`	string	No	Igual que en getMarcas.

```
# primera vez
curl 'https://api.staffattendance.cl/v2/client/getCambiosMarcas?desde=11/11/2026' -H 'Autho
rization: Bearer AUTH_TOKEN'
# siguientes: con el cursor recibido
curl 'https://api.staffattendance.cl/v2/client/getCambiosMarcas?cursor=djF8Zm9ydXN8...' -H
'Authorization: Bearer AUTH_TOKEN'
```


```
{
  "cursor": "djF8Zm9ydXN8NTE4MzQxMjN8OTIyMzM3MjAzNjg1NDc3NTgwN3wxNzk0NDMwNDAwMDAw",
  "hayMas": false,
  "cantidad": 2,
  "cambios": [
    {
      "operacion": "alta",
      "idAttendance": "13163137_2026_11_11_092000000",
      "rut": "13163137-5",
      "fecha": "11/11/2026",
      "hora": "09:20:00",
      "sentido": "Entrada",
      "crc": "1119096083",
      "codigoLocal": "0035",
      "cambio": "2026-11-11T12:20:03.512Z"
    },
    {
      "operacion": "anulacion",
      "idAttendance": "13163137_2026_11_10_180500000",
      "rut": "13163137-5",
      "fecha": "10/11/2026",
      "hora": "18:05:00",
      "sentido": "Salida",
      "crc": "998877",
      "codigoLocal": "0035",
      "cambio": "2026-11-11T12:41:10.002Z"
    }
  ]
}
```


Campo	Descripción
`cursor`	Para la llamada siguiente. **Guárdelo solo después de procesar la respuesta**: si algo falla, repita la llamada con el cursor anterior y recibirá exactamente lo mismo.

Campo	Descripción
`hayMas`	`true` si quedaron cambios por entregar: llame de inmediato con el cursor nuevo. `false` : está al día.
`cambios[].operacion`	`alta` (marca nueva), `modificacion` (cambió la hora, el sentido o el RUT de una marca vigente) o `anulacion` (la marca dejó de ser válida: desactivada, rechazada o eliminada).
`cambios[].idAttendance`	Identificador estable de la marca: úselo para actualizar o borrar en su sistema. Una marca puede aparecer más de una vez si cambió varias veces; vale el **último** cambio.
`cambios[].cambio`	Hora (UTC) en que ocurrió el cambio en Staff Attendance.

> **Garantías:** los cambios salen en orden y **ninguno se salta ni se repite** entre llamadas consecutivas (solo se entregan cambios de transacciones ya terminadas). Las marcas pendientes de autorización no son vigentes: si después se autorizan llegan como `modificacion` . Un `cursor` de más de 30 días responde `410 Gone` : sincronice el período con `getMarcas` y pida un cursor nuevo con `desde` . El cursor solo sirve para el cliente de la credencial (otro responde `400`). Disponible para cambios desde el **11-11-2026**; lo anterior, con `getMarcas` .
>
> **Frecuencia sugerida:** cada 1 a 5 minutos. Una llamada sin cambios es liviana.

3.14 getMarcasPendientes y postConfirmarMarcas

> **Nuevo en v2.9.0.** Mismo contenido que getCambiosMarcas, pero el cursor lo guarda **Staff Attendance**, uno por credencial. Útil si su integración no puede guardar el cursor.

1. `GET /v2/client/getMarcasPendientes` devuelve los cambios después del cursor guardado. **No lo avanza.** La primera vez (sin cursor guardado) hay que indicar `desde` (`DD/MM/YYYY`); sin él responde `409` . Parámetros `limite` y `codigoLocal` como en `getCambiosMarcas` .
2. Procese la respuesta y llame a `POST /v2/client/postConfirmarMarcas?cursor=<cursor recibido>` : desde ahí, `getMarcasPendientes` entrega solo lo posterior.

```
curl 'https://api.staffattendance.cl/v2/client/getMarcasPendientes?desde=11/11/2026' -H
'Authorization: Bearer AUTH_TOKEN'
curl -X POST 'https://api.staffattendance.cl/v2/client/postConfirmarMarcas?
cursor=djF8Zm9ydXN8...' -H 'Authorization: Bearer AUTH_TOKEN'
```

```
{ "confirmado": true, "cursor": "djF8Zm9ydXN8..." }
```

A diferencia de otros sistemas, leer no avanza el cursor: si su proceso falla antes de confirmar, la siguiente llamada vuelve a entregar los mismos cambios. Si `hayMas` es `true`, confirme y vuelva a pedir. Cada credencial tiene su propio cursor (dos integraciones del mismo cliente no se interfieren).

---

4. Endpoints de escritura (POST y PUT)

Batch nativo: el body de todos estos endpoints es un **arreglo JSON** con N elementos. La validación y el procesamiento son por elemento: un error en un elemento no aborta el resto. El campo `details` de la respuesta describe el resultado agregado.

POST vs PUT: ambos aceptan el mismo body. **POST** solo crea: rechaza el registro si la clave ya existe. **PUT** crea o actualiza (*upsert*) y, al actualizar, conserva los campos omitidos (ver la nota de la sección 2).

4.1 postMarca

Inserta marcas (entrada/salida) de empleados.

Permiso especial: escribir marcas (`postMarca` y `deleteNarca`) requiere el permiso **Marcas (escritura)** (scope `client.marcas.insertar`), que habilita GPI Software a solicitud de la empresa. El acceso total (`client`) y la escritura general (`client.write`) **no** lo incluyen; sin él la API responde `403` . La lectura de marcas (`getMarcas` , scope `client.marcas.read`) sigue disponible para cualquier credencial.

Importante: `postMarca` registra marcas **válidas de inmediato**, sin aceptación del trabajador. Úselo para traspasar marcas que el trabajador hizo en otro sistema o dispositivo. Para **corregir** marcas (agregar una que faltó, cambiar una hora o eliminar una marca) use Corrección de marcas con aceptación del trabajador: la normativa de la Dirección del Trabajo exige que el trabajador acepte las correcciones que hace el empleador.

`POST /v2/client/postMarca`

Body: `Array<Marca>`

Campo	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT del empleado sin DV.

Campo	Tipo	Requerido	Descripción
`fecha`	string `DD/MM/YYYY`	Sí	Fecha de la marca.
`hora`	string `HH:mm:ss`	Sí	Hora de la marca (24h).
`sentido`	int	Sí	`0` = Entrada, `1` = Salida.
`email`	boolean	No	`true` para enviar comprobante por email (default `true`).
`idLector`	string	No	Número de serie del lector donde se hizo la marca, tal como lo entrega `idLector` en `getEstadoLectores`. Debe ser un lector del cliente; si no lo es, la marca se rechaza. Omitirlo o enviarlo vacío si la marca no viene de un lector registrado.

```
curl -X POST 'https://api.staffattendance.cl/v2/client/postMarca' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '[
    {
      "rut": 13427038,
      "fecha": "15/10/2024",
      "hora": "10:25:00",
      "sentido": 0,
      "email": true,
      "idLector": "226381B31777"
    }
  ]'
```

Respuesta exitosa:

```
{
  "timestamp": "2024-10-15T17:21:45.270+00:00",
  "message": "success",
  "details": "ID=13427038_2024_10_15_102500000 OK,"
}
```

Respuesta con error:

```
{
  "timestamp": "2024-10-15T14:34:56.791+00:00",
  "message": "error",
  "details": "ID=15795833_2024_10_15_100000 RUT no pertenece a cliente.,"
}
```

Cada marca se valida por separado y `details` informa el resultado de cada una. Errores posibles por marca:

Mensaje	Causa
`Fecha no valida. formato DD/MM/YYYY`	`fecha` no tiene el formato o no es una fecha real.
`Hora no valida. formato hh24:mm:ss`	`hora` no tiene el formato `HH:mm:ss`.
`RUT no es valido.`	`rut` fuera del rango de un RUT sin dígito verificador.
`RUT no pertenece a cliente.`	El trabajador no está registrado en el cliente.
`El sentido de la marca debe ser entre 0 y 3`	`sentido` fuera de rango.
`idLector no corresponde a un lector del cliente.`	Se envió `idLector` y no es el número de serie de un lector del cliente.
`Marca ya existe`	Ya hay una marca del trabajador con la misma fecha y hora.

4.2 postAusentismo / putAusentismo

Crea o actualiza ausentismos (permisos, vacaciones, licencias, etc.).

`POST /v2/client/postAusentismo` o `PUT /v2/client/putAusentismo`

Body: `Array<Ausentismo>`

Campo	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT del empleado.
`fecha`	string `DD/MM/YYYY`	Sí	Fecha del ausentismo.
`tipo_permiso`	string	Sí	Código del tipo definido en `getTipoAusentismo`.
`observaciones`	string	No	Texto libre. Si se omite y ya existe un ausentismo ese día, se conserva su observación.

> **Clave de upsert:** `(rut, fecha)`. Si ya existe un ausentismo ese día para ese RUT, se reemplaza por el nuevo registro. También se elimina el turno del día si existiese; esa eliminación queda registrada en el reporte de modificaciones de turnos.

```
curl -X POST 'https://api.staffattendance.cl/v2/client/postAusentismo' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '[
    {
      "rut": 13163137,
      "fecha": "20/10/2024",
      "tipo_permiso": "L",
      "observaciones": "licencia de prueba"
    }
  ]'
```

```
{
  "timestamp": "2024-10-15T14:36:52.884+00:00",
  "message": "success",
  "details": "ID=demo_13163137_2024-10-20 OK,"
}
```

4.3 postCargo / putCargo

Crea o actualiza cargos.

`POST /v2/client/postCargo` o `PUT /v2/client/putCargo`

Body: `Array<Cargo>`

Campo	Tipo	Requerido	Descripción
`codigoCargo`	int / string	Sí	Código numérico del cargo.
`nombreCargo`	string	Sí (al crear)	Nombre del cargo. Si se omite en un cargo existente, se conserva.

> **Clave de upsert:** `codigoCargo`. Un cargo existente conserva el orden (prioridad) configurado en Staff Attendance.

```
curl -X POST 'https://api.staffattendance.cl/v2/client/postCargo' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '[
    { "codigoCargo": "7070", "nombreCargo": "CARGO DE PRUEBA" }
  ]'
```

```
{
  "timestamp": "2024-10-15T14:42:45.800+00:00",
  "message": "success",
  "details": "ID=demo_7070 OK,"
}
```

4.4 postEmpleado / putEmpleado

Crea o actualiza empleados.

`POST /v2/client/postEmpleado` o `PUT /v2/client/putEmpleado`

Body: `Array<Empleado>`

Campo	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT sin puntos, sin DV, sin guion.
`nombre`	string	Sí (al crear)	Nombre completo. Si se omite en un empleado existente, se conserva.
`email`	string	No	Email válido (regex estándar). Se valida el formato si viene. Si se omite en un empleado existente, se conserva.
`horasContrato`	int	Sí	Horas semanales de contrato. Rango válido: **1 a 48**.
`inicioContrato`	string `DD/MM/YYYY`	Sí (al crear)	Fecha de inicio. Si se omite en un empleado existente, se conserva.
`finContrato`	string `DD/MM/YYYY`	No	Fecha de término. Si se omite, el contrato queda indefinido (`01/01/3000`), también en un empleado existente: es la excepción a la regla de campos omitidos.
`codigoCargo`	string	Sí	Debe existir en `getCargos` .
`codigoLocal`	string	Sí	Debe existir en `getLocales` .

Campo	Tipo	Requerido	Descripción
`codigoLocalDestino`	string	No	Tienda alternativa para asignación temporal del empleado. Debe existir en `getLocales`. **No** se valida contra el scope de tiendas de la credencial — por diseño el destino puede ser cualquier tienda del cliente (la idea es enviar al empleado a un local que la credencial no administra). Convención: campo **omitido o** `null` preserva el valor previo (no toca la asignación); **string vacío** `""` la limpia; valor explícito la actualiza.
`activo`	boolean	No	Empleado nuevo: `false` si se omite. Empleado existente: si se omite, se conserva el valor actual.
`articulo22`	boolean	No	Empleado nuevo: `false` si se omite. Empleado existente: si se omite, se conserva el valor actual.
`marcaMovil`	boolean	No	Empleado nuevo: `false` si se omite. Empleado existente: si se omite, se conserva el valor actual.
`marcaWeb`	boolean	No	Empleado nuevo: `false` si se omite. Empleado existente: si se omite, se conserva el valor actual.
`marcaQR`	boolean	No	Empleado nuevo: `false` si se omite. Empleado existente: si se omite, se conserva el valor actual.

Clave de upsert: `rut`.

Perfil: la API no recibe el perfil. Un empleado nuevo se crea con perfil trabajador; un empleado existente **conserva su perfil** (por ejemplo, manager), su foto, su última conexión y su estado de enrolamiento.

Password: no se acepta como input. Para empleados nuevos se genera una contraseña aleatoria de 16 caracteres con `newPassword = true` (el usuario debe cambiarla en el primer login). Para empleados existentes se preserva la contraseña anterior.

Validaciones: el RUT se valida contra el cliente del token (no puede existir en otro tenant). El `codigoCargo` y `codigoLocal` deben existir previamente en este cliente.

```
curl -X POST 'https://api.staffattendance.cl/v2/client/postEmpleado' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '[
    {
      "rut": 17636490,
      "nombre": "USUARIO DEMO",
      "email": "demo@demo.com",
      "horasContrato": 40,
      "inicioContrato": "01/01/2024",
      "finContrato": "30/12/2024",
      "codigoCargo": "1",
      "codigoLocal": "1",
      "codigoLocalDestino": "2",
      "activo": true,
      "articulo22": false,
      "marcaMovil": false,
      "marcaWeb": false,
      "marcaQR": false
    }
  ]'
```

```
{
  "timestamp": "2024-10-15T14:49:27.984+00:00",
  "message": "success",
  "details": "ID=17636490 OK,"
}
```

Errores típicos:

```
{
  "message": "error",
  "details": "ID=demo_17636490 codigo local no existe en cliente,"
}
```

```
{
  "message": "error",
  "details": "ID=demo_17636490 horasContrato debe estar entre 1 y 48,"
}
```

```
{
  "message": "error",
  "details": "ID=demo_17636490 email con formato invalido,"
}
```

```
{
  "message": "error",
  "details": "ID=demo_17636490 codigo local destino no existe en cliente,"
}
```

4.5 postLocal / putLocal

Crea o actualiza locales.

`POST /v2/client/postLocal` o `PUT /v2/client/putLocal`

Body: `Array<Local>`

Campo	Tipo	Requerido	Descripción
`codigoLocal`	string	Sí	Código único dentro del cliente.
`nombreLocal`	string	Sí (al crear)	Nombre del local. Si se omite en un local existente, se conserva.
`direccionLocal`	string	Sí (al crear)	Dirección. Si se omite en un local existente, se conserva.
`manangerRut`	int	No	RUT del manager (sin DV). Si se omite en un local existente, se conserva el manager actual; `0` lo quita.
`supervisorRut`	int	No	RUT del supervisor. Si se omite en un local existente, se conserva el supervisor actual; `0` lo quita.
`rutAlternativo`	int	No	RUT de la razón social alternativa. Si se omiten `rutAlternativo` y `razonSocialAlternativo` en un local existente, se conserva la razón social alternativa actual.
`razonSocialAlternativo`	string	No	Nombre de la razón social alternativa.
`idTimezone`	string	No	Timezone. Local nuevo: `Santiago-Chile (GMT -04:00)` si se omite. Local existente: se conserva.
`managersAdicionales`	int[]	No	RUTs (sin DV) de managers adicionales al principal (`manangerRut`). Ver reglas abajo.

Campo	Tipo	Requerido	Descripción
`supervisoresAdicionales`	int[]	No	RUTs (sin DV) de supervisores adicionales al principal (`supervisorRut`). Ver reglas abajo.

Nota sobre el campo `manangerRut` : el nombre del campo contiene un typo (`manangerRut` en lugar de `managerRut`) que se mantiene por compatibilidad.

Nuevo en v2.6 — responsables adicionales. `manangerRut` y `supervisorRut` siguen siendo el manager y el supervisor **principal** del local. Los campos `managersAdicionales` y `supervisoresAdicionales` son opcionales y se tratan por separado:

- **Campo omitido (o `null`):** los responsables adicionales actuales de ese rol **no se modifican**. Una integración que no envía estos campos funciona exactamente igual que antes.
- **Arreglo con valores:** reemplaza el conjunto completo de responsables adicionales de ese rol en el local.
- **Arreglo vacío `[]` :** elimina todos los responsables adicionales de ese rol.
- Cada RUT se valida con las mismas reglas que `manangerRut` / `supervisorRut` (entre 1.000.000 y 99.999.999 y perteneciente a un empleado del cliente). Si un RUT no es válido, el local no se guarda y se informa el error en `details` (por ejemplo `ID=demo_CX25 RUT Manager adicional no pertenece a cliente.`).
- Si un RUT adicional coincide con el principal del mismo rol o viene repetido, se ignora (no se duplica).

```
curl -X POST 'https://api.staffattendance.cl/v2/client/postLocal' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '[
    {
      "codigoLocal": "CX25",
      "nombreLocal": "LOCAL DE PRUEBA DEMO API",
      "direccionLocal": "DIRECCION DE PRUEBA 123, SANTIAGO, RM",
      "manangerRut": 0,
      "supervisorRut": 0,
      "rutAlternativo": 12345678,
      "razonSocialAlternativo": "RAZON SOCIAL ALTERNATIVA SA",
      "supervisoresAdicionales": [15123456, 16234567]
    }
  ]'
```

```
{
  "timestamp": "2024-10-15T14:58:32.299+00:00",
  "message": "success",
  "details": "ID=demo_CX25 OK,"
}
```

4.6 postJornadaPorDia / putJornadaPorDia

Crea o actualiza la jornada (turno) de un empleado para un día específico.

`POST /v2/client/postJornadaPorDia` o `PUT /v2/client/putJornadaPorDia`

Body: `Array<Jornada>`

Campo	Tipo	Requerido	Descripción
`diaJornada`	string `DD/MM/YYYY`	Sí	Día del turno.
`rut`	int	Sí	RUT del empleado.
`horaInicio`	string `HH:mm`	Sí	Hora inicio del turno.
`horaTermino`	string `HH:mm`	Sí	Hora fin del turno.
`colacionInicio`	string `HH:mm`	No	Hora inicio colación.
`colacionTermino`	string `HH:mm`	No	Hora fin colación. Si se omiten los dos campos de colación, el turno queda sin colación (el turno del día se reemplaza completo).
`solicitante`	string	No	`T` = Trabajador, `E` = Empleador. Default `E`.
`fechaAsignacion`	string `DD/MM/YYYY`	No	Fecha de asignación. Default: fecha actual.

Clave de upsert: `(rut, diaJornada)`. El turno existente es reemplazado; conserva quién solicitó el cambio de turno y su resolución. Cada creación o modificación queda registrada en el reporte de modificaciones de turnos de Staff Attendance, con el turno anterior, el nuevo y la fecha del cambio.

```
curl -X POST 'https://api.staffattendance.cl/v2/client/postJornadaPorDia' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '[
    {
      "diaJornada": "15/10/2024",
      "rut": 17636490,
      "horaInicio": "10:00",
      "horaTermino": "20:00",
      "colacionInicio": "14:00",
      "colacionTermino": "15:00",
      "solicitante": "E"
    }
  ]'
```

```
{
  "timestamp": "2024-10-15T15:03:32.301+00:00",
  "message": "success",
  "details": "ID=demo_17636490_2024-10-15 OK,"
}
```

4.7 postTipoAusentismo / putTipoAusentismo

Crea o actualiza tipos de ausentismo del cliente.

`POST /v2/client/postTipoAusentismo` o `PUT /v2/client/putTipoAusentismo`

Body: `Array<TipoAusentismo>`

Campo	Tipo	Requerido	Descripción
`codigoAusentismo`	string	Sí	Código corto (ej: `LIC`, `PA`).
`descripcionAusentismo`	string	Sí (al crear)	Descripción legible. Si se omite en un tipo existente, se conserva.

> **Clave de upsert:** `codigoAusentismo`.

```
curl -X POST 'https://api.staffattendance.cl/v2/client/postTipoAusentismo' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '[
    { "codigoAusentismo": "LIC", "descripcionAusentismo": "LICENCIA DE PRUEBA" }
  ]'
```

```
{
  "timestamp": "2024-10-22T20:28:51.259+00:00",
  "message": "success",
  "details": "ID=demo_LIC OK,"
}
```

4.8 postHoraExtraManual / putHoraExtraManual

Registra horas extras ingresadas manualmente por API (no calculadas por el sistema).

`POST /v2/client/postHoraExtraManual` o `PUT /v2/client/putHoraExtraManual`

Body: `Array<HoraExtraManual>`

Campo	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT del empleado.
`fecha`	string `DD/MM/YYYY`	Sí	Fecha.
`horaExtraMinutos`	int	Sí	Cantidad de minutos extra (> 0).
`observaciones`	string	No	Texto libre. Si se omite y ya existe una hora extra manual ese día, se conserva su observación.

> **Clave de upsert:** `(rut, fecha)`.

```
curl -X POST 'https://api.staffattendance.cl/v2/client/postHoraExtraManual' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '[
    {
      "rut": 16058298,
      "fecha": "22/10/2024",
      "horaExtraMinutos": 60,
      "observaciones": "prueba desde api"
    }
  ]'
```

```
{
  "timestamp": "2024-10-22T21:19:02.832+00:00",
  "message": "success",
  "details": "ID=demo_22_10_2024_16058298 OK,"
}
```

4.9 Corrección de marcas con aceptación del trabajador

Permite corregir marcas con el mismo ciclo de aprobación de la pantalla **Modificación de marcas** de Staff Attendance, y gestionar las solicitudes de corrección que hacen los propios trabajadores desde la app o la web.

> **Habilitación:** esta función se habilita por empresa. Si no está habilitada, la creación y la resolución responden `La corrección de marcas por solicitud no está habilitada para la empresa`; solicite la habilitación a soporte.

Solo días anteriores a hoy: las correcciones (y la aprobación de solicitudes) solo se aceptan para días anteriores a la fecha actual en Chile, según la Resolución Exenta N° 38, art. 41 letra c.

Ciclo de una corrección del empleador:

1. La corrección queda **pendiente**: las marcas corregidas se registran sin efecto y las marcas originales siguen vigentes.
2. El trabajador recibe un correo y una notificación con el detalle (marca anterior, marca corregida y motivo) y los botones **Aceptar** y **Rechazar**.
3. Si la acepta, la corrección se aplica y el trabajador recibe el comprobante. Si la rechaza, no se aplica. **Si no responde en 48 horas, se entiende aceptada** (aceptación tácita): se aplica y recibe el comprobante, igual que las correcciones hechas desde la pantalla.

Ciclo de una solicitud del trabajador: el trabajador pide la corrección de sus marcas de un día con un motivo. Al crearla, el supervisor y el encargado de su local reciben un aviso por correo (con botones para aprobarla o rechazarla), notificación y push. Queda **pendiente** hasta que el empleador la aprueba o la rechaza con un motivo, que se le comunica al trabajador. La solicitud es una facilidad del sistema: **al aprobarla, lo pedido se registra como una corrección del empleador** y sigue el mismo ciclo descrito arriba (correo al trabajador para aceptar u oponerse; si no responde en 48 horas se entiende aceptada). Si el empleador quiere registrar otra hora, rechaza la solicitud y registra su propia corrección, que el trabajador debe aceptar.

Persona responsable (`rutResponsable`): toda corrección o resolución se registra a nombre de una persona del empleador, que debe tener acceso a **Modificación de marcas** con permiso de edición en la empresa. Queda en el historial de modificaciones de marcas como autora del cambio.

Para **resolver solicitudes de trabajadores**, además, la persona responsable solo puede resolver las de los locales que tiene a su cargo, igual que en la pantalla: un supervisor o encargado, las de los trabajadores de sus locales (como principal o adicional); un perfil de administración, las de toda la empresa. Una solicitud de un local que no está a su cargo responde `Solicitud no encontrada`.

Casilleros del día: cada corrección describe cómo deben quedar las cuatro marcas del día (`entrada`, `inicioColacion`, `finColacion`, `salida`), asignadas igual que en la pantalla:

Valor del casillero	Efecto
ausente o `null`	Sin cambio: se mantiene la marca actual.
`""` (vacío)	Sin marca: se elimina la marca actual de ese casillero.
`"HH:mm:ss"`	La marca queda con esa hora (se agrega si no había).

4.9.1 postSolicitudCorreccionMarca

`POST /v2/client/postSolicitudCorreccionMarca` (permiso "Corrección de marcas", escritura: `client.correccionMarcas.write`)

Body: `Array<Corrección>` (máximo 100 elementos; uno por trabajador y día)

Campo	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT del trabajador sin DV.
`fecha`	string `DD/MM/YYYY`	Sí	Día de las marcas.
`entrada`	string `HH:mm:ss`	No	Ver tabla de casilleros.
`inicioColacion`	string `HH:mm:ss`	No	Ver tabla de casilleros.
`finColacion`	string `HH:mm:ss`	No	Ver tabla de casilleros.
`salida`	string `HH:mm:ss`	No	Ver tabla de casilleros.
`motivo`	string	Sí	Motivo de la corrección (máx. 500 caracteres). El trabajador lo ve en el correo.
`rutResponsable`	int	Sí	RUT sin DV de la persona del empleador que registra la corrección.

```
curl -X POST 'https://api.staffattendance.cl/v2/client/postSolicitudCorreccionMarca' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '[
    {
      "rut": 13427038,
      "fecha": "24/10/2026",
      "salida": "18:00:00",
      "motivo": "Olvidó marcar la salida",
      "rutResponsable": 11111111
    }
  ]'
```

Respuesta (`200` , un resultado por elemento):

```
[
  {
    "rut": 13427038,
    "fecha": "24/10/2026",
    "result": "success",
    "details": "Corrección registrada; queda pendiente de la aceptación del trabajador",
    "idSolicitud": 1532
  }
]
```

Errores posibles por elemento (`result: "error"`):

Mensaje	Causa
`rutResponsable es obligatorio`	No se envió `rutResponsable` .
`rutResponsable sin permiso: ...`	La persona no existe, no está activa o no tiene permiso de edición en Modificación de marcas.
`No puedes modificar tus propias marcas: ...`	`rutResponsable` es el mismo trabajador: la corrección debe registrarla otra persona (procedimiento bilateral de la Resolución Exenta N° 38).
`RUT no pertenece a cliente.`	El trabajador no está registrado en la empresa.
`Fecha no válida. Formato DD/MM/YYYY` / `Hora no válida en <casillero>. Formato HH:mm:ss`	Formato incorrecto.
`El motivo es obligatorio` / `Indica al menos un casillero a corregir`	Faltan datos.

Mensaje	Causa
`No hay cambios respecto de las marcas actuales`	Lo enviado coincide con las marcas actuales.
`La fecha ... esta en periodo de corte...`	El día está en un periodo cerrado.
`Solo se pueden modificar marcas de días anteriores a hoy: ...`	La fecha es hoy o futura. La Resolución Exenta N° 38 (art. 41 letra c) permite corregir marcas recién desde el día siguiente.
`El colaborador no tiene un mail asociado...`	El trabajador no tiene correo para recibir la solicitud de aceptación.
`La corrección de marcas por solicitud no está habilitada para la empresa`	Función no habilitada.

4.9.2 getSolicitudesCorreccionMarca

`GET /v2/client/getSolicitudesCorreccionMarca` (permiso "Corrección de marcas", lectura: `client.correccionMarcas.read`)

Parámetro	Tipo	Requerido	Descripción
`fecha_inicio`	string `DD/MM/YYYY`	Sí	Desde (fecha de las marcas).
`fecha_fin`	string `DD/MM/YYYY`	Sí	Hasta (incluye el día completo). Rango máximo 60 días.
`rut`	int	No	Filtra por trabajador.
`iniciador`	string	No	`empleador` o `trabajador`.
`estado`	string	No	Filtra por estado (ver tabla).

Iniciador	Estados
`empleador`	`pendiente` (esperando al trabajador), `aceptada` (por el trabajador o, sin respuesta en 48 horas, en forma tácita), `rechazada`, `parcial` (aceptó una parte).
`trabajador`	`pendiente` (esperando al empleador), `aprobada`, `rechazada`, `anulada` (la retiró el trabajador). Las `aprobada` traen además `aceptacion` (`pendiente`, `aceptada`, `rechazada` o `parcial`): la respuesta del trabajador a la corrección.

```
[
  {
    "idSolicitud": 1533,
    "rut": 13427038,
    "nombre": "JUAN PÉREZ",
    "local": "TIENDA CENTRO",
    "fecha": "23/10/2026",
    "iniciador": "trabajador",
    "estado": "pendiente",
    "motivo": "Olvidé marcar la salida",
    "solicitado": { "entrada": null, "inicioColacion": null, "finColacion": null,
"salida": "18:05:00" },
    "anterior": { "entrada": "08:58:12", "inicioColacion": "", "finColacion": "",
"salida": "" },
    "creada": "23/10/2026 21:14:03",
    "resuelta": null,
    "motivoResolucion": null
  }
]
```


En `solicitado`, `null` = sin cambio y `""` = sin marca; en `anterior`, `""` = no había marca. `local` es el nombre del local del trabajador.

4.9.3 postResolverSolicitudCorreccionMarca

Aprueba o rechaza una solicitud **del trabajador** pendiente.

`POST /v2/client/postResolverSolicitudCorreccionMarca` (permiso "Corrección de marcas", escritura: `client.correccionMarcas.write`)

Campo	Tipo	Requerido	Descripción
`idSolicitud`	long	Sí	De `getSolicitudesCorreccionMarca`.
`aprobar`	boolean	Sí	`true` aprueba (se aplica lo pedido); `false` rechaza.
`motivo`	string	Al rechazar	Se comunica al trabajador.
`rutResponsable`	int	Sí	RUT sin DV de quien resuelve: mismo permiso que para corregir y el local del trabajador a su cargo (ver "Persona responsable" en 4.9).

```
curl -X POST 'https://api.staffattendance.cl/v2/client/
postResolverSolicitudCorreccionMarca' \
  -H 'Authorization: Bearer AUTH_TOKEN' \
  -H 'Content-Type: application/json' \
  -d '{ "idSolicitud": 1533, "aprobar": true, "rutResponsable": 11111111 }'
```

```
{
  "idSolicitud": 1533,
  "result": "success",
  "details": "Solicitud aprobada; el trabajador recibió la corrección para aceptarla u
oponerse (si no responde en 48 horas se entiende aceptada)"
}
```

Si la solicitud ya fue resuelta o anulada, o cambió algo que impide aplicarla (por ejemplo, el día quedó en un periodo de corte), responde `result: "error"` con el motivo y la solicitud sigue como estaba.

---

5. Endpoints de eliminación (DELETE)

> Los endpoints DELETE retornan 400 si el recurso no existe, 409 si hay dependencias que bloquean el borrado, y 500 ante errores no controlados (con el mensaje en el body).

5.1 deleteLocal

Borra un local. **Retorna 409 Conflict** si el local tiene empleados asociados.

`DELETE /v2/client/deleteLocal`

Parámetro	Tipo	Requerido	Descripción
`idLocal`	string	Sí	Código del local.

```
curl -X DELETE 'https://api.staffattendance.cl/v2/client/deleteLocal?idLocal=CX25' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
{
  "timestamp": "2024-10-22T15:56:26.912+00:00",
  "message": "success",
  "details": "codigoLocal:demo_CX25"
}
```

Conflicto:

```
{
  "message": "error",
  "details": "Local tiene 12 empleado(s) asociado(s)"
}
```

5.2 deleteEmpleado

Desactiva (soft-delete) a un empleado. Marca `active = false` preservando turnos, marcas, ausentismos y horas extras históricos.

`DELETE /v2/client/deleteEmpleado`

Parámetro	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT del empleado.

```
curl -X DELETE 'https://api.staffattendance.cl/v2/client/deleteEmpleado?rut=16058298' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
{
  "timestamp": "2024-10-22T19:47:02.486+00:00",
  "message": "success",
  "details": "rut:16058298 (desactivado)"
}
```

> **Nuevo en v2.1:** este endpoint cambió de hard-delete a soft-delete para no romper la integridad de los datos históricos. Para reactivar un empleado basta con hacer `postEmpleado` / `putEmpleado` con `activo: true`.

5.3 deleteJornadaPorDia

Borra la jornada de un empleado en un día específico. La eliminación queda registrada en el reporte de modificaciones de turnos.

`DELETE /v2/client/deleteJornadaPorDia`

Parámetro	Tipo	Requerido	Descripción
`diaTurno`	string `DD/MM/YYYY`	Sí	Día a borrar.
`rut`	int	Sí	RUT del empleado.

```
curl -X DELETE 'https://api.staffattendance.cl/v2/client/deleteJornadaPorDia?
diaTurno=22/10/2024&rut=16058298' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
{
  "timestamp": "2024-10-22T19:59:42.159+00:00",
  "message": "success",
  "details": "codigoTurno:demo_16058298_2024-10-22"
}
```

5.4 deleteCargo

Borra un cargo. **Retorna 409 Conflict** si el cargo tiene empleados asociados.

`DELETE /v2/client/deleteCargo`

Parámetro	Tipo	Requerido	Descripción
`codigoCargo`	string	Sí	Código del cargo.

```
curl -X DELETE 'https://api.staffattendance.cl/v2/client/deleteCargo?codigoCargo=876' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
{
  "timestamp": "2024-10-22T20:04:37.150+00:00",
  "message": "success",
  "details": "codigoCargo:876"
}
```

5.5 deleteAusentismo

Borra ausentismos de un RUT en una fecha específica, filtrando por tipo.

`DELETE /v2/client/deleteAusentismo`

Parámetro	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT del empleado.
`fecha`	string `DD/MM/YYYY`	Sí	Fecha del ausentismo.
`tipo`	string	Sí	Tipo del ausentismo (debe coincidir).

```
curl -X DELETE 'https://api.staffattendance.cl/v2/client/deleteAusentismo?
rut=16058298&fecha=10/10/2024&tipo=L' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
{
  "timestamp": "2024-10-22T20:14:19.353+00:00",
  "message": "success",
  "details": "ausentismo: afcc1f44-093b-45c3-b832-b811a0d1f8a7"
}
```

> **Nuevo en v2.1:** si existen múltiples ausentismos en ese día para el RUT, se borran **todos los que coinciden con el** `tipo`. Antes podía borrarse uno arbitrario del día aunque el tipo no coincidiera.

5.6 deleteTipoAusentismo

Borra un tipo de ausentismo del cliente.

`DELETE /v2/client/deleteTipoAusentismo`

Parámetro	Tipo	Requerido	Descripción
`codigoAusentismo`	string	Sí	Código del tipo.

```
curl -X DELETE 'https://api.staffattendance.cl/v2/client/deleteTipoAusentismo?
codigoAusentismo=LIC' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
{
  "timestamp": "2024-10-22T20:58:38.573+00:00",
  "message": "success",
  "details": "codigoTipoAusentismo:demo_LIC"
}
```

5.7 deleteHoraExtra

Borra una hora extra **calculada** (la que aparece en `getHorasExtras`).

`DELETE /v2/client/deleteHoraExtra`

Parámetro	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT del empleado.
`dia`	string `DD/MM/YYYY`	Sí	Día de la hora extra.

```
curl -X DELETE 'https://api.staffattendance.cl/v2/client/deleteHoraExtra?
rut=10789819&dia=10/10/2024' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
{
  "timestamp": "2025-01-10T12:00:00.000+00:00",
  "message": "success",
  "details": "horaExtra: 10789819_1728518400000"
}
```

> **Nuevo en v2.1:** este endpoint antes era un stub que siempre devolvía 200 sin hacer nada. Ahora borra efectivamente el registro.

5.8 deleteHoraExtraManual

Borra una hora extra **manual** (la ingresada por `postHoraExtraManual` / `putHoraExtraManual`).

`DELETE /v2/client/deleteHoraExtraManual`

Parámetro	Tipo	Requerido	Descripción
`rut`	int	Sí	RUT del empleado.
`fecha`	string `DD/MM/YYYY`	Sí	Fecha de la hora extra.

```
curl -X DELETE 'https://api.staffattendance.cl/v2/client/deleteHoraExtraManual?
rut=10789819&fecha=10/10/2024' \
  -H 'Authorization: Bearer AUTH_TOKEN'
```

```
{
  "timestamp": "2024-10-22T20:58:38.573+00:00",
  "message": "success",
  "details": "rut:10789819 fecha=10/10/2024"
}
```