Guía de Integración - API gMedic

Contenido

1. Introducción 2
2. URL base 2
3. Credenciales de integración 3
   3.1. Recomendaciones de seguridad 3
4. Gestión de Credenciales en gMedic 3
5. Obtención del token de acceso 5
6. Uso del token 6
7. Renovación del token 6
8. Headers 7
9. Códigos HTTP 7
10. Ejemplo: API de citas 8
    10.1. Consultar citas 8
    10.2. Consultar disponibilidad de citas 9
    10.3. Crear una cita 10
    10.4. Actualizar una cita 12
    10.5. Estados de una cita 12
    10.6. Modalidades de cita 13
    10.7. Consulta de agendas 13
11. Consulta de IDs de diccionarios 14
    11.1. Consultar una clase de diccionario 14
    11.2. Consultar un valor específico 15
    11.3. Consultar por ID 16
    11.4. Consultar todos los diccionarios 16
    11.5. Clases utilizadas por otros servicios 16

11.6. Recomendación para integradores 17
12. Flujo típico de integración 17
13. Manejo de errores de autenticación 18
14. WebHooks 19
15. Log de Invocaciones de API 21
16. Consideraciones para ambientes productivos 21
17. Documentación OpenAPI 21
18. Soporte 22

1. Introducción

La API de gMedic permite integrar aplicaciones y sistemas externos con diferentes funcionalidades de la plataforma gMedic mediante servicios REST.

Esta guía describe los conceptos generales necesarios para realizar una integración, incluyendo:

- Credenciales de acceso.
- Obtención del token de autenticación.
- Uso del token en las solicitudes al API.
- Formato general de las solicitudes y respuestas.
- Manejo de errores.
- Ejemplo de integración utilizando el API de citas.

La disponibilidad de los diferentes recursos y operaciones del API dependerá de los servicios habilitados para cada cliente.

---

2. URL base

Todos los servicios deben consumirse utilizando HTTPS.

La URL base será suministrada por gMedic de acuerdo con el ambiente correspondiente.

Ejemplo:

https://<servidor-gmedic>/api

Los ejemplos de este documento utilizan rutas relativas a esta URL. Actualmente se debe reemplazar por “gmedic.co”

---

3. Credenciales de integración

Para consumir los servicios protegidos de gMedic se asignará una credencial de integración.

La credencial está compuesta por:

- Client ID
- Client Secret

El Client ID identifica la integración.

El Client Secret es una credencial privada y debe almacenarse de forma segura en el sistema que realiza la integración.

3.1. Recomendaciones de seguridad

El Client Secret:

- No debe incluirse directamente en código fuente.
- No debe almacenarse en repositorios de código.
- No debe enviarse por correo electrónico, logs u otros medios no seguros.
- No debe exponerse desde aplicaciones web o móviles al usuario final.
- Debe almacenarse utilizando los mecanismos seguros disponibles en la infraestructura del sistema consumidor.

Las credenciales deben utilizarse desde un componente de backend controlado por el integrador.

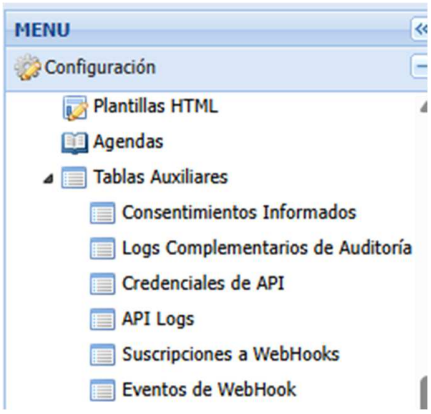
---

4. Gestión de Credenciales en gMedic

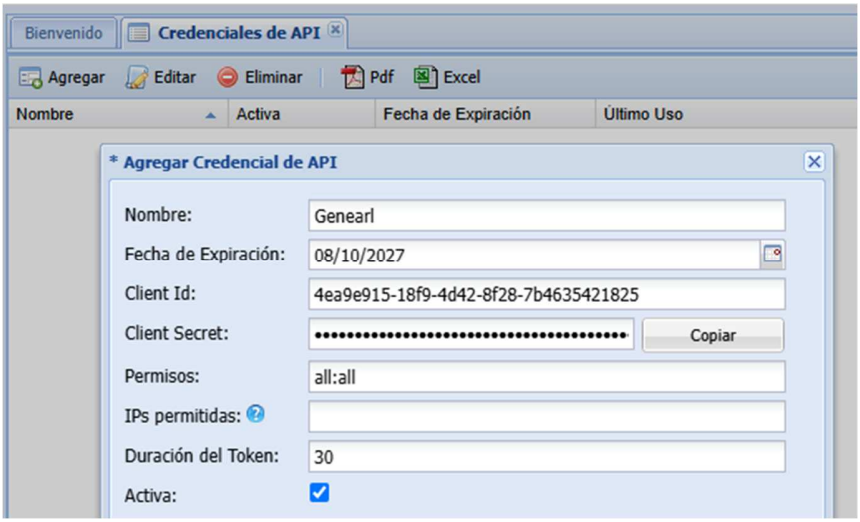
Para gestionar las credenciales y el API en general en gMedic el usuario necesita el rol +API Management con estas opciones disponibles:

- Credenciales de API: para crear el par de llave de de acceso: clientId+clientSecret
- Suscripciones a WebHooks: se detalla más adelante
- Eventos de WebHook: se detalla más adelante
- API Logs: para ver el log de consumo a través de API

Disponible en el menú Configuración gMedic, Tablas auxiliares:



Para crear una nueva credencial, en Credenciales API, Agregar:



Se debe asignar un nombre, fecha de vencimiento, permisos (all:all para todos los permisos y separados por coma para permisos específicos), IPs permitidas (vacío para todas y separadas por coma para más de una), duración del token (entre 10 y 60 min), y activa. El client Id y client secret se generan de forma automática y se deben copiar y almacenar de forma segura para la obtención del token de acceso al API.

Detalle de permisos:

Método	Endpoint	Scope
GET	/appointments	appointment:read
POST	/appointments	appointment:write
PUT	/appointments	appointment:write
GET	/appointments/agendas	agenda:read
GET	/appointments/available	agenda:read

GET	/appointments/available/days	agenda:read
GET	/appointments/services	catalog:read
GET	/common/arl	catalog:read
GET	/common/branchoffices	catalog:read
GET	/common/dictionaries	catalog:read
GET	/common/eps	catalog:read
GET	/common/healthpartners	catalog:read
GET	/common/occupations	catalog:read
GET	/common/services	catalog:read
GET	/common/patientnotes	patientnote:read
POST	/common/patientnotes	patientnote:write
PUT	/common/patientnotes	patientnote:write
GET	/patients	patient:read
POST	/patients	patient:write
PUT	/patients	patient:write
GET	/patients/consents	consent:read
GET	/patients/consents/byPatient	consent:read
GET	/sessions	session:read
POST	/sessions	session:write

---

5. Obtención del token de acceso

Antes de consumir los servicios protegidos del API, el sistema cliente debe obtener un token de acceso utilizando su Client ID y Client Secret.

Solicitud

```
POST /api/auth/token
Content-Type: application/json
Body:
{
  "clientId": "<client-id>",
  "clientSecret": "<client-secret>"
}
```

Respuesta exitosa

El servicio devuelve un token de acceso y su tiempo de vigencia.

Ejemplo:

```
{
  "accessToken": "<jwt>",
  "tokenType": "Bearer",
  "expiresIn": 1800
}
```

- accessToken contiene el token que debe utilizarse en las siguientes solicitudes.
- tokenType indica el esquema de autenticación utilizado.
- expiresIn indica la vigencia del token expresada en segundos.

El valor exacto de duración puede depender de la configuración asignada a la credencial.

---

6. Uso del token

Una vez obtenido el token, debe enviarse en el header HTTP Authorization de cada solicitud protegida.

Formato:

Authorization: Bearer <accessToken>

Ejemplo:

```
GET /api/private/appointments?id=123
Authorization: Bearer <accessToken>
```

El Client ID y el Client Secret no deben enviarse nuevamente para consumir cada operación.

Su función es obtener el token de acceso.

---

7. Renovación del token

Los tokens de acceso tienen una vigencia limitada.

Cuando un token expire, el sistema cliente debe solicitar uno nuevo utilizando nuevamente:

Client ID + Client Secret

No se debe asumir que un token puede utilizarse indefinidamente.

Una estrategia recomendada para el sistema consumidor es:

1. Obtener un token.
2. Almacenar temporalmente el token y su tiempo de expiración.
3. Reutilizarlo mientras continúe vigente.
4. Solicitar un nuevo token cuando haya expirado o esté próximo a expirar.

No es necesario solicitar un token nuevo para cada llamada al API. gMedic controla el RateLimit de tokens y no se puede emitir más de un token para la misma credencial en el lapso de 1 minuto.

---

8. Headers

Las solicitudes deben utilizar los headers correspondientes al tipo de operación.

Para solicitudes JSON:

```
Content-Type: application/json
Accept: application/json
Authorization: Bearer <accessToken>
```

Algunas operaciones pueden requerir adicionalmente:

```
X-App-User: <usuario>
```

X-App-User identifica al usuario o sistema que origina la operación dentro de la aplicación consumidora.

Cuando sea requerido, debe enviarse un valor que permita al integrador identificar posteriormente quién originó la operación.

Ejemplos:

```
X-App-User: jperez
```

o:

```
X-App-User: integracion-crm
```

---

9. Códigos HTTP

La API utiliza códigos HTTP estándar para indicar el resultado de las operaciones.

Código	Significado
200 OK	Consulta realizada correctamente.
201 Created	Recurso creado correctamente.

202 Accepted	Actualización procesada correctamente.
204 No Content	La consulta fue válida pero no produjo resultados.
400 Bad Request	La solicitud contiene parámetros faltantes o inválidos.
401 Unauthorized	No fue posible autenticar la solicitud.
404 Not Found	No se encontró el recurso solicitado.
500 Internal Server Error	Ocurrió un error interno procesando la solicitud.

El sistema consumidor debe validar siempre el código HTTP antes de procesar el contenido de la respuesta.

---

10. Ejemplo: API de citas

El API de citas permite consultar, crear y actualizar citas registradas en gMedic.

Ruta principal:

```
/api/private/appointments
```

10.1. Consultar citas

Operación

```
GET /api/private/appointments
```

La operación permite consultar una cita específica o buscar citas utilizando diferentes parámetros.

Entre los filtros disponibles se encuentran:

- Id
- agendaId
- patientId
- date
- pending
- pageNumber

Cuando se especifica id, los demás parámetros de búsqueda son ignorados.

Para búsquedas sin id debe utilizarse al menos uno de los filtros requeridos por la operación.

Ejemplo: consultar una cita por ID

```
GET /api/private/appointments?id=123
```

```
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
Accept: application/json
```

Ejemplo: consultar citas de un paciente

```
GET /api/private/appointments?patientId=456
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
Accept: application/json
```

Ejemplo: consultar citas de una agenda

```
GET /api/private/appointments?agendaId=20&date=20250125
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
Accept: application/json
```

El parámetro date utiliza el formato:

yyyyMMdd

Ejemplo:

20250125

Respuesta

Una respuesta exitosa contiene uno o varios objetos de cita.

Ejemplo simplificado:

```
[
  {
    "id": 123,
    "title": "Paciente: CARLOS DIAZ",
    "startDate": "20250401 13:30:00",
    "endDate": "20250401 14:00:00",
    "agendaId": 20,
    "agendaName": "Consulta Medicina General",
    "service": "Consulta Medicina General"
  }
]
```

Si la búsqueda es válida pero no existen resultados, el servicio puede responder:

204 No Content

10.2. Consultar disponibilidad de citas

Para consultar espacios disponibles puede utilizarse:

```
GET /api/private/appointments/available
```

La búsqueda admite:

- agendaId
- startDate
- endDate

Ejemplo:

```
GET /api/private/appointments/available?agendaId=20&startDate=20250501&endDate=20250507
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
```

Las fechas utilizan formato:

yyyyMMdd

La consulta está diseñada para trabajar preferiblemente con rangos cortos y, cuando sea posible, especificando una agenda.

También se encuentra disponible una operación para determinar qué días de un mes tienen disponibilidad:

```
GET /api/private/appointments/available/days
```

Ejemplo:

```
GET /api/private/appointments/available/days?agendaId=20&month=202509
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
```

El parámetro month utiliza formato:

yyyyMM

10.3. Crear una cita

Operación

```
POST /api/private/appointments
```

La información de la cita debe enviarse en formato JSON.

Ejemplo:

```
POST /api/private/appointments
Authorization: Bearer <accessToken>
```

```
Content-Type: application/json
Accept: application/json
X-App-User: integracion-crm
Body:
{
  "title": "Paciente: CARLOS DIAZ",
  "startDate": "20250401 13:30",
  "endDate": "20250401 14:00",
  "agendaId": 20,
  "patientId": 456
}
```

Los campos básicos requeridos para la creación son:

- Title
- startDate
- endDate
- agendaId
- patientId

Las fechas y horas utilizan el formato:

yyyyMMdd HH:mm

Ejemplo:

20250401 13:30

Dependiendo del proceso de agendamiento pueden enviarse otros atributos disponibles en el modelo de citas.

Entre ellos se encuentran:

- description
- desiredDate
- branchOfficeId
- notificationMail
- notificationNumber
- colorId
- location
- modality
- serviced
- authorization
- medicalAuthorizationId
- authorizationSessions
- invoice
- attentionTypeId
- cupCode
- consultationPurposeId
- externalCauseId
- medicalServiceId

- servicesGroupId
- serviceModalityId
- viaEntryId

Respuesta exitosa

Cuando la cita es creada correctamente:

201 Created

La respuesta contiene la información de la cita creada.

10.4. Actualizar una cita

La API permite actualizar información de una cita existente.

Operación

```
PUT /api/private/appointments
```

El identificador de la cita debe enviarse mediante el parámetro:

id

Dependiendo de la modificación pueden enviarse parámetros como:

- Status
- startDate
- endDate
- color
- rejectReasonCode

Ejemplo:

```
PUT /api/private/appointments?id=123&status=CONFIRMED
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
```

Una actualización procesada correctamente devuelve:

202 Accepted

10.5. Estados de una cita

Los estados disponibles actualmente incluyen:

- SCHEDULED
- ATTENDED
- UNATTENDED

- CANCELED
- PENALIZED
- CONFIRMED
- DECLINED

Cuando una cita es rechazada (DECLINED), algunas operaciones pueden utilizar un código de motivo de rechazo.

Los códigos documentados actualmente son:

- 01 - No puedo asistir en ese horario
- 02 - Quiero reagendar
- 03 - Ya no me interesa la cita
- 04 - Otro motivo personal

10.6. Modalidades de cita

Las modalidades disponibles son:

- FACE_TO_FACE
- VIRTUAL
- HOME_VISIT
- NOT_APPLICABLE

La modalidad debe enviarse utilizando exactamente uno de los valores soportados por el API.

10.7. Consulta de agendas

Antes de crear una cita puede ser necesario consultar las agendas disponibles.

Operación:

```
GET /api/private/appointments/agendas
```

La consulta permite utilizar filtros como:

- Id
- branchOfficeId
- serviced

Ejemplo:

```
GET /api/private/appointments/agendas?serviceId=25
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
```

La respuesta contiene las agendas disponibles y puede incluir información como:

- id
- name
- category
- service
- branchOffice
- selfServiceEnabled
- selfServiceAllowCancel
- selfServiceAllowReschedule

El id obtenido puede utilizarse posteriormente como agendaId al consultar disponibilidad o crear una cita.

---

11. Consulta de IDs de diccionarios

Algunas operaciones de la API de gMedic requieren enviar identificadores (id) correspondientes a valores de catálogos internos o diccionarios.

Por ejemplo, al crear o actualizar un paciente se utilizan campos como:

```
identificationTypeId
genderId
locationZoneTypeId
educationLevelId
maritalStatusId
ethnicityId
disabilityCategoryId
healthRegimeId
```

En estos casos, el valor esperado no corresponde al nombre visible del elemento sino al identificador (`id`) asignado por gMedic.

Para obtener estos identificadores se encuentra disponible el servicio de diccionarios:

```
GET /api/private/common/dictionaries
```

11.1. Consultar una clase de diccionario

El parámetro `class` permite consultar todos los valores disponibles para un tipo de diccionario.

Por ejemplo, para obtener los tipos de identificación:

```
GET /api/private/common/dictionaries?class=IdentificationType
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
Accept: application/json
```

La respuesta contiene los valores disponibles para esa clase. Cada elemento incluye información como:

```
[
  {
    "id": 14,
    "name": "CC",
    "label": "Cédula de Ciudadanía",
    "value": "CC"
  },
  {
    "id": 15,
    "name": "CE",
    "label": "Cédula de Extranjería",
    "value": "CE"
  }
]
```

Los valores anteriores son ilustrativos. Los IDs que deben utilizarse son siempre los retornados por el API del ambiente correspondiente.

Una vez identificado el elemento requerido, su `id` puede utilizarse en otras operaciones.

Por ejemplo:

```
{
  "identification": "91568987",
  "identificationTypeId": 14,
  "firstName": "Juan Carlos",
  "lastName": "Perez Rodríguez"
}
```

11.2. Consultar un valor específico

Cuando se conoce el nombre del valor requerido, pueden enviarse conjuntamente los parámetros `class` y `name`.

Ejemplo:

```
GET /api/private/common/dictionaries?class=IdentificationType&name=CC
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
Accept: application/json
```

Esto permite obtener directamente la entrada correspondiente y conocer su `id`.

Por ejemplo:

```
[
  {
    "id": 14,
    "name": "CC",
    "label": "Cédula de Ciudadanía",
    "value": "CC"
```

```
  }
]
```

El identificador retornado podrá utilizarse posteriormente como:

```
{
  "identificationTypeId": 14
}
```

11.3. Consultar por ID

También es posible consultar una entrada cuando ya se conoce su identificador:

```
GET /api/private/common/dictionaries?id=14
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
Accept: application/json
```

11.4. Consultar todos los diccionarios

Si no se especifican parámetros, el servicio permite obtener las entradas de diccionario agrupadas por clase:

```
GET /api/private/common/dictionaries
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
Accept: application/json
```

Para integraciones se recomienda consultar únicamente las clases necesarias, evitando descargar todos los diccionarios cuando no sea requerido.

11.5. Clases utilizadas por otros servicios

La documentación de cada modelo indica la clase de diccionario que debe consultarse cuando un atributo requiere un ID.

Por ejemplo, algunos campos utilizados por el API de citas hacen referencia a las siguientes clases:

Campo	Clase de diccionario
attentionTypeId	MedicalHistoryType
consultationPurposeId	ConsultationPurpose
externalCauseId	ExternalCause
medicalServiceId	MedicalService
servicesGroupId	ServicesGroup
serviceModalityId	ServiceModality
viaEntryId	ViaEntry

Por ejemplo, para conocer los valores disponibles para `consultationPurposeId`:

```
GET /api/private/common/dictionaries?class=ConsultationPurpose
Authorization: Bearer <accessToken>
X-App-User: integracion-crm
```

Posteriormente, el `id` retornado puede utilizarse al crear una cita:

```
{
  "title": "Paciente: CARLOS DIAZ",
  "startDate": "20260930 10:00",
  "endDate": "20260930 10:30",
  "agendaId": 20,
  "patientId": 456,
  "consultationPurposeId": 888
}
```

11.6. Recomendación para integradores

No se recomienda almacenar en el código de la integración los IDs de diccionario como valores fijos.

Cuando sea posible, la integración debe obtener los valores mediante el API de diccionarios y relacionarlos utilizando propiedades estables como `name` o `value`.

Por ejemplo, en lugar de asumir permanentemente:

```
CC = 14
```

Se recomienda consultar:

```
GET /api/private/common/dictionaries?class=IdentificationType&name=CC
```

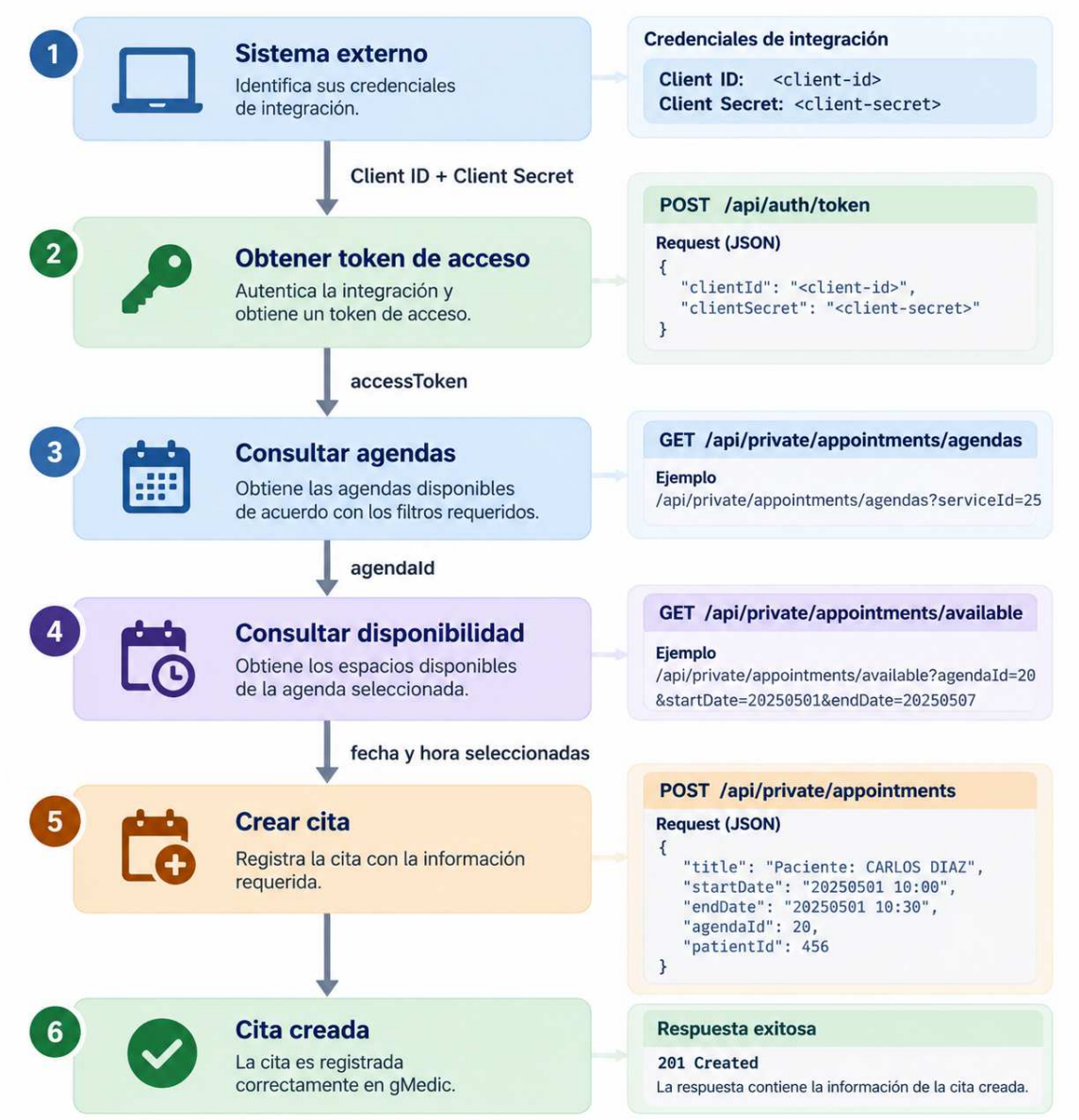
y utilizar el `id` retornado por gMedic.

Esto evita que la integración dependa de identificadores internos que pueden ser diferentes entre ambientes o instalaciones.

---

12. Flujo típico de integración

Un flujo típico para agendar una cita desde un sistema externo es:



13. Manejo de errores de autenticación

El token obtenido inicialmente puede reutilizarse durante su período de vigencia.

Si el token:

- No está presente.
- No es válido.
- Ha expirado.
- Corresponde a una credencial que ya no está activa.
- Corresponde a una credencial vencida.

La solicitud puede responder:

401 Unauthorized

Ante un token expirado, el sistema consumidor debe solicitar un nuevo token utilizando sus credenciales de integración.

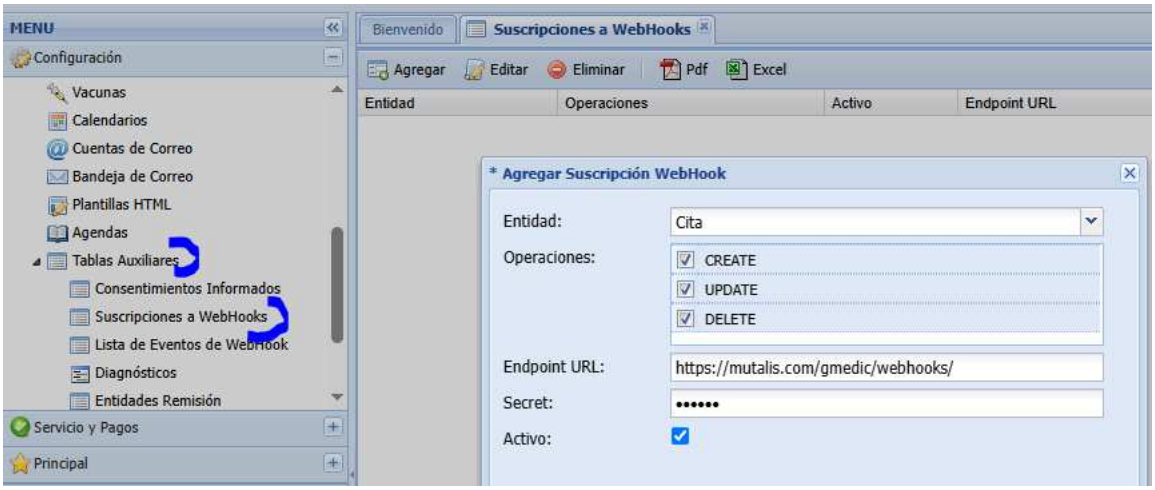
No se recomienda solicitar automáticamente nuevas credenciales de integración. El Client ID y Client Secret son credenciales de larga duración administradas por gMedic; el elemento que se renueva regularmente es el token de acceso.

---

14. WebHooks

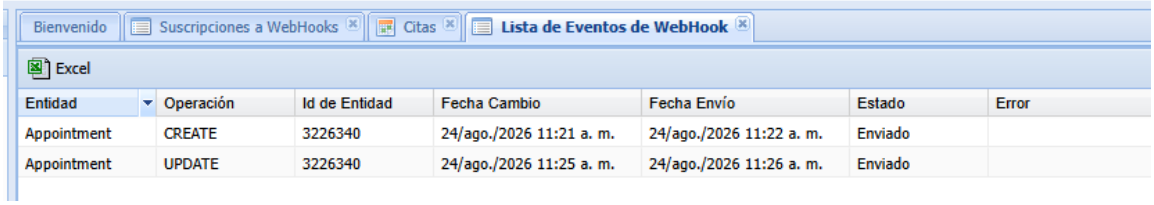
Los WebHookes permiten a terceros suscribirse a eventos de cambios de objetos en gMedic. Cuando un objeto cambia, será notificado al endpoint configurado.

Primero se deben crear las suscripciones. Configuración / Configuración gMedic / Tablas Auxiliares / Suscripciones de WebHooks:



Ingresar la entidad, las acciones a escuchar y la URL donde se notificará el evento. El secret es una clave que ustedes asignan para firmar los mensajes enviados y que puedan comprobar que son enviados de gMedic. No se usa bearer token, solo firma del mensaje.

Luego de creada la suscripción, al crear/modificar/eliminar objetos de ese tipo de entidad, se creará un registro de evento pendiente de notificar que podrán encontrar en Configuración / Configuración gMedic / Tablas Auxiliares / Eventos de WebHook



Allí se podrá revisar los eventos creados y su estado.

Un job que se ejecuta cada minuto evalúa los eventos pendientes y los notifica a la URL registrada con un evento como este:

```
{
  "eventId": 123, // ID único del Evento
  "entity": "Appointment", // Nombre de la entidad modificada (Appointment, Agenda,
Patient, PatientNote)
  "entityId": 32260340, // Id de la entidad modificada (con ese id pueden consultar cuál
fue la cita que se modificó para actualizar el estado)
  "operation": "UPDATE", // Tipo de operación (CREATE, UPDATE, DELETE)
  "changeDate": "2026-08-24T11:25:10-05:00" // Fecha y hora del cambio de la entidad
}
```

Por seguridad, no se incluyen datos de entidades en estos eventos (aunque deberían usarse endpoints con HTTPS un atacante podría interceptar los mensajes). Se deben consultar por id usando el API de gMedic donde se controla el acceso.

En la cabecera del mensaje HTTP se enviará:

```
X-GMedic-Timestamp
X-GMedic-Signature
```

La firma del mensaje está en X-GMedic-Signature. Para firmar el mensaje se usa "timestamp + payload" y el algoritmo HmacSHA256

Un ejemplo para validar la firma en Java. Como la firma es un digest, se debe volver a firmar el mensaje con el secret que se configura en la suscripción, y validar si el resultado es la misma firma que se envía en el header:

```
private boolean verifySignature(String timestamp, String payload, String signingSecret, String
receivedSignature) throws Exception {

    String expectedSignature = sign(timestamp, payload, signingSecret);

    return MessageDigest.isEqual(expectedSignature.getBytes(StandardCharsets.UTF_8),
receivedSignature.getBytes(StandardCharsets.UTF_8));
}

private String sign(String timestamp, String payload, String secret) throws Exception {

    String message = timestamp + "." + payload;
    Mac mac = Mac.getInstance("HmacSHA256");
    SecretKeySpec key = new SecretKeySpec(secret.getBytes(StandardCharsets.UTF_8),
"HmacSHA256");
    mac.init(key);
    byte[] digest = mac.doFinal(message.getBytes(StandardCharsets.UTF_8));

    return "sha256=" + toHex(digest);
}

private String toHex(byte[] bytes) {

    StringBuilder result = new StringBuilder(bytes.length * 2);

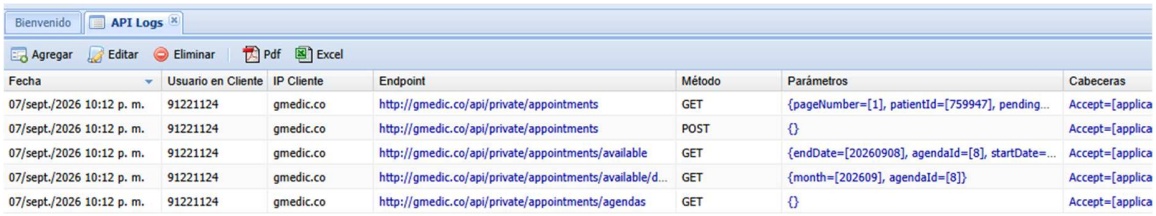
    for (byte b : bytes) {
```

```
        result.append(String.format("%02x", b));
    }

    return result.toString();
}
```

15. Log de Invocaciones de API

Toda invocación al API queda registrada en el menú Configuración, Configuración gMedic, Tablas Auxiliares, API Log:



Fecha	Usuario en Cliente	IP Cliente	Endpoint	Método	Parámetros	Cabeceras
07/sept./2026 10:12 p. m.	91221124	gmedic.co	http://gmedic.co/api/private/appointments	GET	{pageNumber=[1], patientId=[759947], pending...	Accept=[applica
07/sept./2026 10:12 p. m.	91221124	gmedic.co	http://gmedic.co/api/private/appointments	POST	{}	Accept=[applica
07/sept./2026 10:12 p. m.	91221124	gmedic.co	http://gmedic.co/api/private/appointments/available	GET	{endDate=[20260908], agendaId=[8], startDate=...	Accept=[applica
07/sept./2026 10:12 p. m.	91221124	gmedic.co	http://gmedic.co/api/private/appointments/available/d...	GET	{month=[202609], agendaId=[8]}	Accept=[applica
07/sept./2026 10:12 p. m.	91221124	gmedic.co	http://gmedic.co/api/private/appointments/agendas	GET	{}	Accept=[applica

16. Consideraciones para ambientes productivos

Las integraciones deben:

- Utilizar siempre HTTPS.
- Proteger adecuadamente el Client Secret.
- Reutilizar el token mientras se encuentre vigente.
- Implementar timeouts en las conexiones HTTP.
- Validar los códigos HTTP recibidos.
- Manejar adecuadamente expiraciones de token.
- Evitar registrar tokens completos o credenciales en logs.
- Evitar enviar solicitudes repetidas innecesariamente ante errores.
- Registrar internamente un identificador que permita rastrear las operaciones realizadas desde el sistema consumidor.

17. Documentación OpenAPI

gMedic dispone de una especificación OpenAPI que describe los recursos disponibles, parámetros de entrada, modelos de datos y respuestas de cada operación.

Se puede consultar en: https://gmedic.co/apidocs/

Allí mismo se puede obtener un token y usar ese token para pruebas en la misma interfaz gráfica.

La especificación OpenAPI debe utilizarse como referencia detallada para conocer todos los atributos soportados por cada servicio.

Esta guía tiene como objetivo explicar el mecanismo general de integración y no reemplaza la definición técnica de cada operación publicada en OpenAPI.

---

18. Soporte

Para consultas relacionadas con la integración con el API de gMedic:

Soporte API gMedic
Correo: soporte@kloudsolutions.com.co

Al reportar un inconveniente se recomienda incluir:

- Fecha y hora aproximada de la solicitud.
- Endpoint utilizado.
- Método HTTP.
- Código HTTP recibido.
- Descripción del comportamiento observado.
- Identificación del sistema o integración que realizó la solicitud.

Por razones de seguridad, no se deben enviar Client Secret ni tokens de acceso completos en solicitudes de soporte.