

ModuloGik

PLATAFORMA DE REGISTRO Y MONITORIZACIÓN

Manual de la API REST — Referencia completa

Guía de integración para desarrolladores: autenticación, exportación masiva de datos, dashboards, gráficas, alarmas y gestión de dispositivos IoT.

Versión 1.0 · Base URL: <https://gureak.webdatalogger.net> · Protocolo: HTTPS + JWT

Índice

1. Autenticación	3
1.1 Login	3
1.2 Cabecera Authorization	3
1.3 Roles y permisos	4
2. Exportación masiva de datos	5
2.1 GET /api/export/info — Pre-verificación	5
2.2 GET /api/export/lecturas	6
2.3 GET /api/export/gnss	7
2.4 GET /api/export/alarmas	7
2.5 GET /api/export/sensores	8
2.6 GET /api/export/modulos	8
2.7 Parámetros comunes	9
2.8 Ejemplos de uso	9
3. Datos en tiempo real	11
3.1 GET /api/modules	11
3.2 GET /api/sensors	11
3.3 GET /api/latest	12
3.4 GET /api/timeseries	12
3.5 GET /api/gnss	13
4. Alarmas	14
5. Gráficas personalizadas	15
6. Dashboards	16
7. Health check	17

8. Códigos de error	18
9. Límites y rate limiting	19
10. Ejemplos completos	20

1. Autenticación

ModuloGik utiliza **JSON Web Tokens (JWT)** transmitidos como cookie `HttpOnly mgk_token` o como cabecera `Authorization: Bearer <token>`. Todos los endpoints (salvo `/api/login` y `/api/health`) requieren autenticación.

1.1 Login

POST `/api/login`

Autentica al usuario y devuelve un JWT. El token también se establece como cookie `mgk_token`.

Campo (body JSON)	Tipo	Descripción
<code>username</code> *	string	Nombre de usuario o email
<code>password</code> *	string	Contraseña

`curl`

```
curl -X POST https://gureak.webdatalogger.net/api/login \ -H "Content-Type: application/json" \ -d '{"username":"admin","password":"mi_clave"}' \ -c cookies.txt  
# guardar cookie para requests siguientes
```

```
{ "ok": true, "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...", "rol": "admin",  
  "idp": 42, "username": "admin" }
```

❑ **Rate limiting:** máximo 5 intentos por minuto por IP. Tras 10 fallos consecutivos, la IP queda bloqueada 10 minutos.

1.2 Cabecera Authorization

Una vez obtenido el token, inclúyelo en cada petición de dos formas:

opción A – cookie (sesión de navegador)

```
curl https://gureak.webdatalogger.net/api/modules -b cookies.txt
```

opción B – header Bearer (integraciones / scripts)

```
curl https://gureak.webdatalogger.net/api/modules \ -H "Authorization: Bearer eyJhbGci..."
```

1.3 Roles y permisos

Rol	Acceso
root	Acceso total a todos los proyectos, clientes y configuración del sistema
admin	Acceso total limitado a su proyecto y proyectos del mismo cliente
admingrupo	Acceso total limitado a su grupo/zona
usuario	Solo lectura, limitado a su grupo/zona

❑ **Multi-tenant:** un *admin* solo ve los datos de su proyecto y los proyectos del mismo cliente. Nunca puede acceder a datos de otros clientes, independientemente del filtro que use.

2. Exportación masiva de datos

La API de exportación permite descargar grandes volúmenes de datos históricos de forma eficiente. Utiliza **streaming HTTP** y **cursores server-side de PostgreSQL** para manejar millones de filas sin saturar la memoria del servidor.

Característica	Detalle
Memoria del servidor	O(2 000 filas) independientemente del total
Primer byte	< 1 segundo aunque el export tarde minutos
Compresión gzip	Reduce el tráfico un 70-90 % (opcional)
Formatos	CSV (streaming), XLSX (≤50 000 filas), JSON (streaming)
Límite por defecto	100 000 filas
Límite máximo	500 000 filas por petición
Rate limit	1 export cada 30 segundos por usuario

2.1 GET /api/export/info — Pre-verificación

GET /api/export/info

Devuelve el conteo estimado de filas para cada recurso dado el rango de fechas y filtros. **Úsalo siempre antes de descargar** para confirmar el volumen. No consume el cooldown de 30 segundos.

curl

```
curl "https://gureak.webdatalogger.net/api/export/info?
desde=1748000000&hasta=1750000000" \ -H "Authorization: Bearer $TOKEN"
```

```
{ "desde": 1748000000, "hasta": 1750000000, "recursos": { "lecturas": { "filas":
635826, "tamaño_csv_estimado": "30.6 MB", "exportable": true, "aviso_grande":
true }, "gnss": { "filas": 4521, ... }, "alarmas": { "filas": 87, ... } },
"limites": { "max_filas": 500000, "xlsx_max_filas": 50000, "cooldown_s": 30 } }
```

2.2 GET /api/export/lecturas — Lecturas de sensores

GET /api/export/lecturas

Exporta las lecturas históricas de los sensores con nombre del sensor, módulo y unidad.

Columnas CSV: fecha_hora , epoch , modulo , sensor , codigo_sensor , valor , unidad

Parámetro	Tipo	Default	Descripción
desde *	int (epoch)	7 días atrás	Inicio del rango (Unix timestamp en segundos)
hasta *	int (epoch)	ahora	Fin del rango (Unix timestamp en segundos)
sensor_ids	string	todos	IDs de sensores separados por coma: 1,2,3
fmt	string	csv	csv xlsx json
gz	0 1	0	Comprimir respuesta con gzip
limit	int	100000	Máximo de filas (máx. 500 000)

curl – últimas 24h de todos los sensores, CSV comprimido

```
curl "https://gureak.webdatalogger.net/api/export/lecturas?desde=$(date -d '-1 day' +%s)&hasta=$(date +%s)&gz=1" \ -H "Authorization: Bearer $TOKEN" \ -o lecturas_$(date +%Y%m%d).csv.gz
```

Python – exportar sensor específico a DataFrame de pandas

```
import requests, pandas as pd, io, gzip token = "eyJhbGciOi..." resp = requests.get("https://gureak.webdatalogger.net/api/export/lecturas", params={"desde": 1748000000, "hasta": 1750000000, "sensor_ids": "1,2,3", "gz": "1"}, headers={"Authorization": f"Bearer {token}"}, stream=True) df = pd.read_csv(gzip.open(io.BytesIO(resp.content))) print(df.head())
```

2.3 GET /api/export/gnss — Posiciones GPS

GET /api/export/gnss

Exporta las coordenadas GPS registradas por los módulos.

Columnas CSV: fecha_hora , epoch , modulo , zona , latitud , longitud , altitud , valid , tiempo_conexion_s , reintentos

Parámetro	Tipo	Descripción
desde	int (epoch)	Inicio del rango
hasta	int (epoch)	Fin del rango
modulo_ids	string	IDs de módulos separados por coma
fmt , gz , limit	—	Ver parámetros comunes

2.4 GET /api/export/alarmas — Eventos de alarma

GET /api/export/alarmas

Exporta el historial de alarmas disparadas con su duración y estado.

Columnas CSV: fecha_disparo , fecha_fin , alarma , condicion , severidad , sensor , modulo , zona , valor_disparo , estado , tipo_evento , duracion_s

2.5 GET /api/export/sensores — Catálogo de sensores

GET /api/export/sensores

Exporta el catálogo completo de sensores con unidad, rango y jerarquía organizativa. No requiere rango de fechas.

Columnas CSV: id , codigo , nombre , modulo , zona , proyecto , unidad , unidad_simbolo , rango , rango_min , rango_max

2.6 GET /api/export/modulos — Catálogo de módulos

GET /api/export/modulos

Exporta el inventario de dispositivos IoT con hardware, batería y firmware. No requiere rango de fechas.

Columnas CSV: id, mac, nombre, zona, proyecto, hardware, bateria, firmware, build, ultimo_dato

2.7 Parámetros comunes a todos los endpoints de exportación

Parámetro	Tipo	Default	Descripción
fmt	string	csv	csv — streaming sin límite de memoria xlsx — Excel, máx. 50 000 filas json — array JSON en streaming
gz	0 1	0	Comprimir la respuesta con gzip. Reduce tamaño ~80 %. No compatible con fmt=xlsx.
limit	int	100 000	Máximo de filas. Valor máximo: 500 000. Para exports mayores, divide por rangos de fecha.

2.8 Ejemplos de uso completos

Bash — descargar todo el año 2025 comprimido

```
TOKEN="eyJhbGciOi..." # Verificar volumen primero curl "https://gureak.webdatalogger.net/api/export/info?desde=1735689600&hasta=1767225600" \ -H "Authorization: Bearer $TOKEN" | python3 -m json.tool # Descargar lecturas del año
curl "https://gureak.webdatalogger.net/api/export/lecturas?desde=1735689600&hasta=1767225600&gz=1&limit=500000" \ -H "Authorization: Bearer $TOKEN" \ -o lecturas_2025.csv.gz # Descomprimir gunzip lecturas_2025.csv.gz
```

JavaScript / Node.js — stream a fichero

```
const fs = require('fs'); const https = require('https'); const url = 'https://gureak.webdatalogger.net/api/export/lecturas' + '?desde=1748000000&hasta=1750000000&gz=1'; https.get(url, { headers: { 'Authorization': `Bearer ${TOKEN}` } }, res => { res.pipe(fs.createWriteStream('lecturas.csv.gz')); res.on('end', () => console.log('Descarga completa')); });
```

3. Datos en tiempo real

3.1 GET /api/modules — Lista de módulos

GET /api/modules

Devuelve todos los módulos IoT del proyecto con su estado actual y último dato.

Campo respuesta	Descripción
id , mac , nombre	Identificación del módulo
last_seen_epoch	Último dato (Unix timestamp). Nulo si nunca ha enviado datos
bateria	Nivel de batería en % (0-100). Nulo si no hay sensor de batería
app_nombre , app_build	Firmware activo y número de build
hardware , imagen	Tipo de hardware e imagen (URL)
grupo , proyecto	Zona y proyecto al que pertenece

3.2 GET /api/sensors — Lista de sensores

GET /api/sensors

?module_id=123

Devuelve todos los sensores con su unidad, rango y jerarquía organizativa.

Parámetro opcional: `module_id` para filtrar por módulo.

3.3 GET /api/latest — Último valor por sensor

GET /api/latest?sensor_ids=1,2,3

Devuelve el último valor registrado de cada sensor solicitado. Ideal para dashboards en tiempo real.

Parámetro	Descripción
-----------	-------------

`sensor_ids *`

IDs separados por coma. Máximo recomendado: 100

```
[ { "sensor_id": 1, "epoch": 1750000000, "valor": 23.4, "sensor": "temperatura1",  
  "unidad": "°C", "minimo": -10.0, "maximo": 50.0 } ]
```

3.4 GET /api/timeseries — Serie temporal

GET `/api/timeseries`

Devuelve los valores de sensores en un rango de tiempo. Usado para gráficas y análisis.

Parámetro	Tipo	Descripción
<code>sensor_ids *</code>	string	IDs separados por coma
<code>from</code>	int (epoch)	Inicio del rango
<code>to</code>	int (epoch)	Fin del rango
<code>limit</code>	int	Máximo de puntos (default 10 000, máx. 50 000)

❑ **Para descargas grandes** usa `/api/export/lecturas` en lugar de `/api/timeseries`. Este endpoint está optimizado para dashboards, no para exports masivos.

3.5 GET /api/gnss — Trayectoria GPS

GET `/api/gnss?module_id=5&from=1748000000&to=1750000000`

Devuelve los puntos GPS de un módulo en un rango de tiempo.

4. Alarmas

GET `/api/alarmas`

Lista las alarmas activas y su estado actual para el scope del usuario.

GET `/api/regalarmas?limit=200`

Historial de eventos de alarma (disparos, resoluciones). Parámetro `activas=1` para solo las alarmas en curso.

PUT `/api/regalarmas/<id>/resolver`

Marca una alarma activa como resuelta manualmente. Requiere rol admin o superior.

5. Gráficas personalizadas

GET `/api/charts`

Lista las gráficas guardadas del proyecto actual.

POST `/api/charts/query`

Ejecuta una consulta SQL personalizada (solo SELECT) y devuelve los datos. Requiere rol admin.

Campo (body JSON)	Descripción
<code>sql</code> *	Consulta SELECT. Parámetros: <code>%(from)s</code> y <code>%(to)s</code> (epoch en segundos)
<code>from</code>	Unix timestamp inicio
<code>to</code>	Unix timestamp fin

❑ **Seguridad:** solo se permiten consultas SELECT. Las palabras clave INSERT, UPDATE, DELETE, DROP, CREATE, ALTER, GRANT y REVOKE están bloqueadas.

6. Dashboards

GET `/api/dashboards`

Lista los dashboards accesibles para el usuario actual.

GET `/api/dashboards/<id>/full`

Devuelve el dashboard completo incluyendo todos sus paneles y su configuración.

7. Health check

GET `/api/health`

Verifica el estado del servicio. No requiere autenticación. Útil para monitorización y deploys.

HTTP	status	Significado
200	ok	API y base de datos operativas
207	degraded	API funcional, algún subsistema (ej. evaluador de alarmas) con problemas
503	error	Base de datos no responde — servicio no disponible

```
{ "status": "ok", "env": "production", "uptime_s": 86400, "elapsed_ms": 23.4,
  "checks": { "db": { "status": "ok", "latency_ms": 18.2 }, "alarm_evaluator": {
    "status": "ok" } } }
```

8. Códigos de error

Todos los errores devuelven JSON con el campo `error` :

```
{ "error": "descripción del error" }
```

Código HTTP	Significado	Causa habitual
200	OK	Petición procesada correctamente
201	Created	Recurso creado (POST)
400	Bad Request	Parámetro faltante, tipo incorrecto o rango inválido
401	Unauthorized	Sin token, token expirado o inválido
403	Forbidden	Token válido pero el rol no tiene permiso
404	Not Found	El recurso no existe o está fuera del scope
409	Conflict	Duplicado (MAC ya registrada, nombre ya existe...)
413	Payload Too Large	El export supera 500 000 filas o el límite XLSX
429	Too Many Requests	Rate limit de login (5/min) o de export (1/30s)
500	Server Error	Error interno; contactar soporte
503	Service Unavailable	Base de datos no disponible

9. Límites y rate limiting

Endpoint	Límite	Ventana	Respuesta al superarlo
/api/login	5 intentos fallidos	1 minuto por IP	429 — bloqueo 10 minutos
/api/reset-password-request	5 peticiones	1 minuto por IP	429
/api/export/*	1 export	30 segundos por usuario	429 + retry_after
API general	300 peticiones	1 minuto por IP	429 (gestionado por nginx)
Asistente IA	1 mensaje	15 segundos por usuario	429 + tiempo de espera

Export masivo por rangos: si necesitas más de 500 000 filas, divide la consulta en rangos de fechas y concatena los archivos:

?desde=1735689600&hasta=1743465600 (Q1 2025)

?desde=1743465600&hasta=1751328000 (Q2 2025)

?desde=1751328000&hasta=1759190400 (Q3 2025)

10. Ejemplos completos

Script Bash: exportar datos diarios automáticamente

bash – cron job diario

```
#!/bin/bash # Cron: 0 2 * * * /path/to/export_diario.sh BASE="https://gureak.webdatalogger.net" TOKEN="$MGK_TOKEN" # variable de entorno OUTPUT_DIR="/data/exports" HASTA=$(date +%s) DESDE=$((HASTA - 86400)) DATE=$(date +%Y%m%d) # Verificar volumen antes de descargar INFO=$(curl -s "$BASE/api/export/info?desde=$DESDE&hasta=$HASTA" \ -H "Authorization: Bearer $TOKEN") FILAS=$(echo $INFO | python3 -c "import sys,json; print(json.load(sys.stdin)['recursos']['lecturas']['filas'])") echo "Filas a exportar: $FILAS" # Descargar curl -s "$BASE/api/export/lecturas?desde=$DESDE&hasta=$HASTA&gz=1" \ -H "Authorization: Bearer $TOKEN" \ -o "$OUTPUT_DIR/lecturas_$DATE.csv.gz" echo "Export completado: lecturas_$DATE.csv.gz ($FILAS filas)"
```

Python: análisis con pandas y matplotlib

python

```
import requests, pandas as pd, io, gzip import matplotlib.pyplot as plt from datetime import datetime BASE = "https://gureak.webdatalogger.net" TOKEN = "eyJhbGciOi..." HEADS = {"Authorization": f"Bearer {TOKEN}"} # 1. Ver qué hay disponible info = requests.get(f"{BASE}/api/export/info", params={"desde": 1748000000, "hasta": 1750000000}, headers=HEADS).json() print(info["recursos"]["lecturas"]) # 2. Descargar y cargar en DataFrame resp = requests.get(f"{BASE}/api/export/lecturas", params={"desde": 1748000000, "hasta": 1750000000, "sensor_ids": "1,2", "gz": "1"}, headers=HEADS, stream=True) df = pd.read_csv(gzip.open(io.BytesIO(resp.content)), parse_dates=["fecha_hora"]) # 3. Análisis pivot = df.pivot_table(index=pd.Grouper(key="fecha_hora", freq="1h"), columns="sensor", values="valor", aggfunc="mean") pivot.plot(figsize=(14, 5), title="Temperatura media por hora") plt.tight_layout() plt.savefig("temperatura.png", dpi=150)
```

JavaScript / Node.js: integración con webhook

javascript (Node.js)

```

const https = require('https'); const fs = require('fs'); async function
exportarLecturas(desde, hasta, fichero) { const url = new URL('https://
gureak.webdatalogger.net/api/export/lecturas'); url.searchParams.set('desde',
desde); url.searchParams.set('hasta', hasta); url.searchParams.set('gz', '1');
url.searchParams.set('limit', '500000'); return new Promise((resolve, reject) =>
{ https.get(url.toString(), { headers: { Authorization: `Bearer $
{process.env.MGK_TOKEN}` } }, (res) => { if (res.statusCode !== 200) { reject(new
Error(`HTTP ${res.statusCode}`)); return; }
res.pipe(fs.createWriteStream(fichero)) .on('finish', resolve) .on('error',
reject); }); }); } // Uso exportarLecturas(1748000000, 1750000000,
'datos.csv.gz') .then(() => console.log('▣ Export completado')) .catch(err =>
console.error('▣ Error:', err));

```