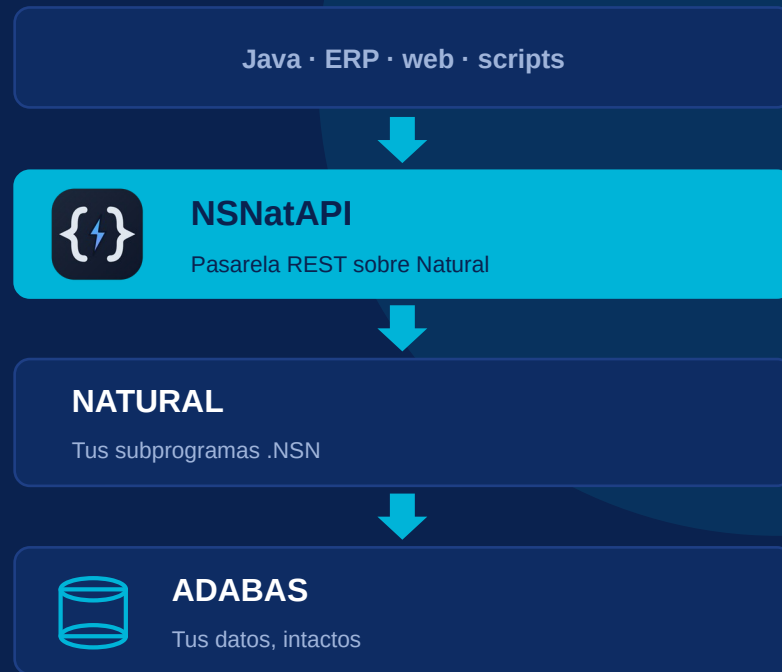


El dato de Adabas, expuesto como API REST

NSNatAPI convierte tus subprogramas Natural en endpoints REST seguros: tokens Bearer, RBAC, auditoría y multi-entorno. Sin EntireX, sin broker RPC, sin reescribir una línea de negocio.

[VER CÓMO FUNCIONA →](#)

26.1 build 185 · 2 modos de ejecución · ~1-5 ms por llamada



Un solo subprograma Natural, dos caminos de ejecución

Adabas es un gran almacén de datos, pero no es una API

01

El silo

Tus datos viven en Adabas y se consultan con subprogramas Natural. Para una app web, un ERP o un móvil, eso no es un contrato: es una INTEGRACIÓN.

02

El peaje

La vía habitual es EntireX o un broker RPC: licencias, un broker que mantener, y un despliegue que se rompe el día que cambia algo.

03

El riesgo

Abrir Natural a la red sin control: sin tokens, sin alcances, sin auditoría, sin límites de concurrencia ni de tiempo.

- Sin API, cada aplicación reinventa la conexión

- Con EntireX, la infraestructura es el proyecto

- Sin control, la seguridad es un problema

QUÉ ES

Una pasarela, no un proxy

NSNatAPI se coloca entre tus aplicaciones y Natural. Tus programas de negocio siguen siendo los mismos .NSN: no se reescriben, no se compilan con nsnatapi, no cambian de contrato. Lo que cambia es cómo se llega hasta ellos.

- ✓ Endpoints REST definidos en consola: método, ruta, entorno, programa y campos.
- ✓ Dos modos de ejecución por endpoint: batch (proceso efímero) o session (pool persistente).
- ✓ Tokens Bearer con alcance por endpoint, caducidad, revocación y allowlist de IP.
- ✓ Consola de administración con RBAC (roles y grupos), TOTP 2FA y configuración en caliente.
- ✓ Auditoría de cada llamada con metadatos: nunca los cuerpos de la request ni de la respuesta.

0

EntireX, brokers ni
demonios en Natural

1–5 ms

por llamada en modo
sesión (medido)

2

modos de ejecución
por endpoint

195k

llamadas auditadas
en la certificación

ARQUITECTURA

De la aplicación al dato: dos caminos, un mismo programa

Servicios consumidores

Java / Backend

ERP

Scripts y cron

Microservicios

Portal web

App móvil

iPaaS / SOAP



NSNatAPI — pasarela REST

REST /api/ext · Bearer + RBAC · rate limit · auditoría · OpenAPI interactivo

REST /api/ext

HTTP 200 · rc · output · duration_ms

MODO BATCH

Un proceso Natural efímero por llamada

LOGON → CALLNAT → FIN. Aislamiento total. 30–90 ms.

MODO SESSION

Pool de sesiones persistentes que abren conexión

LOGON a la librería y la sesión QUEDA ABIERTA, reutilizada. 1–5 ms.

NATURAL

Runtime Natural · librerías NSNAT · buffer pool NATBP · FUSER por entorno

Conexiones abiertas: 1 por sesión del pool (1 usuario del buffer pool cada una)



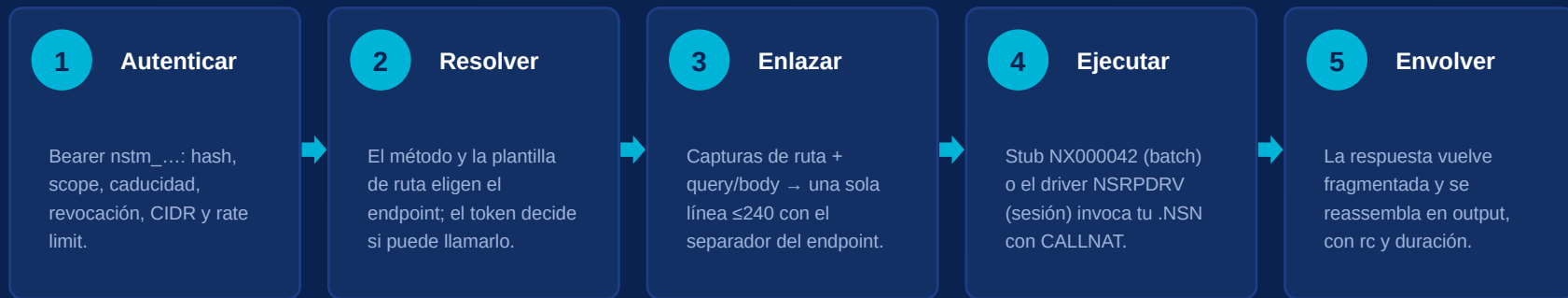
ADABAS

Ficheros de negocio: READ / WRITE / ST

Mismo subprograma .NSN, mismo contrato, mismos datos. Solo cambia quién abre la conexión y cuánto tiempo la mantiene.

CÓMO FUNCIONA

De la petición HTTP a tu subprograma



```
curl -H "Authorization: Bearer nstm_..." \
      http://host:11032/api/ext/v1/saldo/001234
```

```
HTTP/1.1 200 OK
{ "ok": true, "rc": 0,
  "output": "SALDO 1234.56",
  "duration_ms": 22 }
```

HTTP 200 = se ejecutó. El error de negocio viaja en rc: el cliente distingue siempre los dos.

`output · rc · data · duration_ms`

Dos modos, un mismo contrato de programa

BATCH (por defecto)

Un proceso Natural por llamada

Arranque	~30–90 ms en cada llamada
----------	---------------------------

Aislamiento	Total: proceso nuevo por llamada
-------------	----------------------------------

CPU / arranque	Alto con mucha carga
----------------	----------------------

Riesgo	Ninguno: proceso efímero
--------	--------------------------

Ideal para	Tráfico bajo o medio, aislamiento estricto
------------	--

SESSION (opt-in)

Pool de sesiones Natural persistentes

Arranque	~1–5 ms con el pool caliente
----------	------------------------------

Aislamiento	Sesión compartida entre llamadas
-------------	----------------------------------

CPU / arranque	Bajo: el pool se reutiliza
----------------	----------------------------

Riesgo	Un error fatal recicla la sesión (automático)
--------	---

Ideal para	Polling y consulta de alta frecuencia
------------	---------------------------------------

Elige el modo por endpoint, en la consola (Endpoints → Execution mode) o por API con `exec_mode`.

Tus programas no cambian. Solo la entrada y la salida.

```
0010* CPOLIZA - consulta de póliza
0020DEFINE DATA PARAMETER
00301 #PARM-IN (A) DYNAMIC
00401 #PARM-OUT (A) DYNAMIC
00501 #RESP (I4)
0060END-DEFINE
0070RESET #PARM-OUT #RESP
0080* ... tu lógica de negocio ...
0090MOVE 'SALDO 1234.56' TO #PARM-OUT
0100MOVE 0 TO #RESP
0110END TRANSACTION
0120END
```

✓ **Tres parámetros, siempre**

#PARM-IN (A) DYNAMIC, #PARM-OUT (A) DYNAMIC y #RESP (I4).
Es el convenio de la casa.

✓ **Termina con END**

Nunca STOP ni LOGON: en modo sesión matarían el driver.

✓ **Sin estado entre llamadas**

Sin globales ni AIVs; END TRANSACTION o BACKOUT siempre.

✓ **Nombre corto**

A-Z0-9-_, máximo 8 caracteres, en mayúsculas.

nsnatapi no recompila tus programas: solo genera y compila el stub o el driver que los invoca.

RESPUESTAS

El negocio decide el resultado; el HTTP solo el transporte

HTTP 200 · el envelope

ok	true	La llamada se ejecutó
rc	0	#RESP del negocio: 0 = ok, >0 = error de negocio
output	"SALDO 1234.56"	Texto crudo de #PARM-OUT (verbatim)
data	{ "saldo": 1234.56 }	Solo en los modos fields y json
duration_ms	22	Duración de la llamada

rc ≠ 0 sigue siendo HTTP 200. El cliente distingue el error de negocio del fallo técnico.

Errores de la API máquina

400**401****403****404****413****429****502****503****504**

Modos de respuesta

text

Solo output. El rc es lo que importa.

fields

El programa devuelve posiciones separadas y el endpoint las tipa en data.

json

El programa devuelve JSON válido y nsnatapi lo valida y lo publica en data.

Cada petición pasa por un filtro

Tokens Bearer

192 bits, hash SHA-256 con pepper, scope por endpoint, caducidad, revocación, rotación y allowlist de CIDR.

RBAC de consola

Roles → grupos → usuarios. Permisos de la unión. Sin is_admin. Anti-bloqueo del último administrador.

TOTP 2FA

Doble factor opcional por cuenta, con confirmación de contraseña para desactivarlo.

Validación estricta

Todo lo que llega a Natural o al FUSER se valida: sin shell con datos de usuario, sin path traversal.

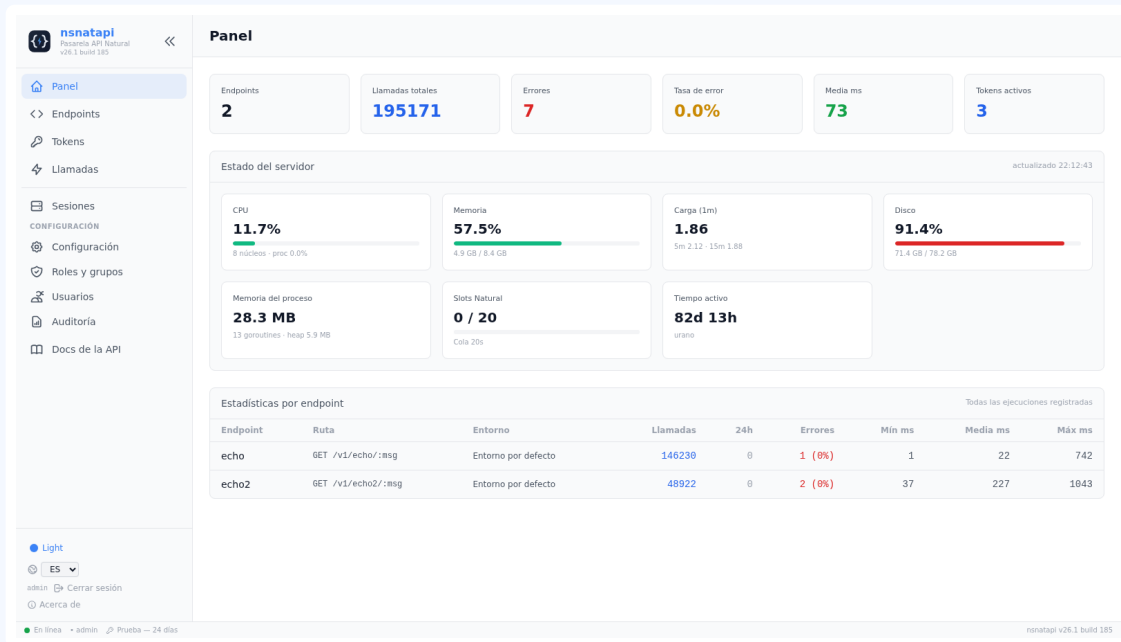
Auditoría sin cuerpos

Cada llamada deja metadatos: entorno, librería, modo, PID, duración, error. Nunca el body.

Licencia y red

Sin licencia válida, el login y la API máquina responden 403. Solo loopback es proxy de confianza por defecto.

Lo que pasa en Natural, a la vista



- Telemetría del servidor en vivo: CPU, memoria, carga, disco, RSS, heap y goroutines, refrescada cada 5 s.
- Panel de sesiones: PID, estado, uptime, última llamada, llamadas servidas, errores, latencia, CPU y memoria por sesión.
- Registro de llamadas con entorno, librería, modo de ejecución y PID de la sesión: cada llamada se correlaciona con su sesión.
- Acciones en caliente: precalentar, reiniciar, matar una sesión concreta o redimensionar el pool.

Todo desde la consola, sin entrar al servidor por SSH.

CAPACIDAD

Capacidad medida, no prometida

MODO SESSION · pool DEV de 4 sesiones

Clientes	rps	p50	p95	Errores
1	106	7,8 ms	19,5 ms	0
4	251	13,9 ms	31,3 ms	0
8	312	23,1 ms	47,7 ms	0
16	321	46,9 ms	79,9 ms	0

Certificado sesión: 16 clientes concurrentes (~320 rps, p95 80 ms, 0 errores)

MODO BATCH · un proceso por llamada

Clientes	rps	p50	p95	Errores
1	8,4	111 ms	191 ms	0
4	25,7	148 ms	230 ms	0
8	40,0	192 ms	285 ms	0
12	42,2	258 ms	483 ms	0

Certificado batch: 8 clientes concurrentes (~40 rps, p95 285 ms, 0 errores)

Carga de cierre sin think time, programas triviales: es el techo de la plataforma, no del negocio. Soak de 15 min: 61.225 llamadas, 0 errores.

MULTI-ENTORNO

Un endpoint, un entorno: DEV, TST y PRD no se mezclan

DEV

6 sesiones

NATBP · fuser=22,20 · FUSER
/data/ns/fusers/dev

FUSER + buffer pool propios

TST

2 sesiones

NATTST · fuser=22,30 · FUSER
/data/ns/fusers/tst

FUSER + buffer pool propios

PRD

a definir

Buffer pool y FUSER propios

FUSER + buffer pool propios

- ✓ Cada entorno declara su binario Natural, su sagenv, su buffer pool, su NATPARM y su FUSER.
- ✓ El endpoint elige entorno; el entorno por defecto cubre los que no lo eligen.
- ✓ El pool de sesiones se dimensiona por entorno: 3 sesiones en DEV y 8 en TST conviven sin mezclarse.
- ✓ Las llamadas quedan registradas con su entorno, y la página Calls filtra por él.

El mismo binario nsnatapi sirve para los tres: cambia el destino, no el código.

DEV · TST · PRD

Dónde aporta valor el día uno

Consultas de back-office

Saldos, pólizas, catalogación: una consulta desde la web sin abrir el terminal.

Alta y captura

Formularios de alta que escriben en Adabas con validación de negocio en el propio programa.

Pasarelas de pago

Un endpoint POST con su rc de negocio: la app distingue 'rechazado' de 'caído'.

Radar móvil

Consultas de alta frecuencia en modo sesión, a coste de milisegundos.

Integración ERP / iPaaS

El ERP llama por HTTP; si habla JSON, el endpoint publica data ya tipado.

Modernización progresiva

Se migra endpoint a endpoint, sin reescribir el negocio Natural ni pararlo.

Puesta en marcha en cuatro pasos

1 Instala y arranca

```
nsnatapi init ./data  
nsnatapi start ./data
```

2 Crea el endpoint

```
POST /api/endpoints  
{  
  method, path_template,  
  library, program,  
  environment, fields_json  
}
```

3 Despliega el stub

```
POST /api/endpoints/1/deploy  
  
(ftouch + STOW en el  
FUSER del entorno)
```

4 Emite el token y llama

```
nsnatapi token ./data \  
  --name java-prd \  
  --endpoint saldo  
  
curl -H "Authorization: \  
  Bearer nstm_..." \  
  http://host:11032/api/ext/  
  v1/saldo/123
```

La consola web (Configuration, Endpoints, Tokens, Sessions) cubre el resto: RBAC, entornos, auditoría y ajustes en caliente.

REQUISITOS

Lo que necesita tu entorno

Requisitos

- Linux RHEL 8 o 9 (x86-64).
- Software AG Natural y cliente Adabas (sagenv), con el buffer pool arrancado.
- FUSER con los subprogramas de negocio, con permiso de lectura y escritura para desplegar.
- Reverse proxy con HTTPS en producción (nsnatapi no sirve TLS por sí mismo).
- Un Natural por entorno: DEV, TST y PRD pueden apuntar a instalaciones distintas.

Licencia

- Licencia por instalación, con huella de máquina y vigencia.
- Sin licencia válida, el login y la API máquina responden 403.
- Prueba de 30 días al instalar, sin clave.
- Precio de lista: 7.900 € / 6.320 € de renovación anual.

La licencia se comprueba en el login y en cada llamada máquina: el interior de Adabas sigue protegido.

SIGUIENTE PASO

Tus datos ya están en Adabas. Solo falta la puerta.

NSNatAPI la abre: con tokens, con alcance, con auditoría y con los dos modos de ejecución que necesitas. Sin reescribir tu negocio Natural y sin EntireX.

1. Demo de 30 minutos con tus propios subprogramas
2. Prueba en DEV con un endpoint real
3. Despliegue en TST y, después, en PRD

PEDIR INFORMACIÓN · nacarsoft.com

Nacarsoft Soluciones Informáticas S.L. · Suite Adabas / Natural · nsnatapi 26.1 build 185

