FIDELITAS S.A. · API PLATFORM

Deploy en Cloud

face-match-node → apibatch.fidelitas.com.ar

Paso a paso completo para transferir el microservicio face-match-node a un servidor Ubuntu 24.04 en la nube, configurar Apache como reverse proxy con SSL/Let's Encrypt, dashboard de estadísticas, prueba de carga y verificación end-to-end. El servidor no compila nada — recibe el código listo para ejecutar.

FECHA
2026-11-07

VERSIÓN
1.2

DOMINIO DESTINO
apibatch.fidelitas.com.ar

SERVIDOR
Ubuntu Server 24.04 LTS · Node.js 26 LTS

CONFIDENCIAL — USO INTERNO

CÓDIGO	VERSIÓN	FECHA	ESTADO	EMISOR	APROBADOR
DOC-OPS-007	1.2	2026-11-07	VIGENTE	Equipo de Tecnología	Dirección de Tecnología

1 Qué se hace y qué NO se hace en el servidor

> **Regla fundamental — el servidor cloud NO compila nada**
>
> No tiene build-essential, no tiene gcc. Todo llega compilado desde la estación de desarrollo. El servidor solo ejecuta.

Lo que se transfiere (vía SFTP o rsync)

```
face-match-node/
├── src/                  ← código fuente JavaScript
│   ├── app.js
│   ├── server.js
│   ├── config/config.js
│   ├── middleware/auth.js
│   ├── routes/faceMatch.js
│   ├── routes/stats.js   ← dashboard + /stats/*
│   └── services/
├── node_modules/         ← binarios compilados para Ubuntu 24.04
├── dashboard.html        ← se sirve en /dashboard (protegido por IP)
├── package.json
├── ecosystem.config.js
├── .env.example
└── .npmrc
```

Lo que NO se transfiere — se crea en el servidor

```
.env          ← credenciales — nunca viaja por red
storage/      ← se crea vacío
data/         ← la DB SQLite se genera al primer request exitoso
logs/         ← se crea vacío
load_test.sh  ← herramienta de desarrollo local, no va al cloud
compile.sh    ← herramienta de desarrollo local, no va al cloud
```

Arquitectura final

```
Internet
    │  HTTPS 443
    ▼
Apache (apibatch.fidelitas.com.ar)
    │  certbot / Let's Encrypt — TLS termina acá
    │
    ├── ProxyPass /v3/face-match/ → localhost:3100
    ├── ProxyPass /dashboard      → localhost:3100  (protegido por IP)
    ├── ProxyPass /stats/         → localhost:3100  (protegido por IP)
    └── ProxyPass /health         → localhost:3100
    ▼
face-match-node (PM2 cluster — un worker por CPU core)
    │  puerto 3100 — solo escucha en 127.0.0.1
    ▼
```

```
RENAPER API (externa)
    + SQLite WAL (/var/www/html/apibatch/data/fide-facematch.db)
    + storage/ (3 archivos por TCN: .txt + 2 × .json)
```

2 Prerequisitos — qué necesitás antes de empezar

En la estación de desarrollo

Qué	Por qué
`node_modules/` generado en Ubuntu 24.04	Verificar con: `ls node_modules/better-sqlite3/prebuilds/linux-x64.node`
Acceso SSH/SFTP al servidor	Clave privada o password del proveedor
IP o hostname del servidor	Te lo da el proveedor al crear la instancia
DNS configurado	`apibatch.fidelitas.com.ar` debe apuntar a la IP del servidor antes de pedir el certificado SSL

En el servidor cloud

Qué	Por qué
Ubuntu Server 24.04 LTS	Mismo OS que se usó para compilar `node_modules`
Puerto 22 abierto	Para acceso SSH/SFTP
Puerto 80 abierto	Para que Let's Encrypt valide el dominio
Puerto 443 abierto	Para HTTPS de producción
Usuario con sudo	Para instalar Apache y Node.js

```
# Verificar DNS desde tu máquina local antes de continuar
dig apibatch.fidelitas.com.ar +short
# Debe devolver la IP del servidor — si no, configurar DNS y esperar 5-30 min
```

3 Preparar el paquete en la estación de desarrollo

```
# Ir a la carpeta del proyecto
cd /home/memyself/Downloads/fide-workspace/web-php/fide-api/face-match-node

# Verificar que el binario nativo de SQLite está compilado para Linux x64
ls node_modules/better-sqlite3/prebuilds/linux-x64.node

# Verificar que las 7 dependencias de producción están todas
node -e "
  ['fastify','@fastify/formbody','@fastify/helmet','@fastify/rate-limit',
   'better-sqlite3','undici','dotenv'].forEach(p=>{require(p);console.log('OK:',p)});
"
```

4 Acceder al servidor cloud por primera vez

```
ssh ubuntu@SERVER_IP

# Con clave .pem (AWS, Azure, etc.)
chmod 400 ~/mi-clave-cloud.pem
ssh -i ~/mi-clave-cloud.pem ubuntu@SERVER_IP
```

```
# Verificar OS y actualizar el sistema
lsb_release -d          # debe mostrar: Ubuntu 24.04.x LTS
sudo apt-get update
sudo apt-get upgrade -y  # puede tardar 2-5 minutos
```

5 Instalar Node.js 26 en el servidor

Solo para **ejecutar** el código. No compila nada.

```
curl -fsSL https://deb.nodesource.com/setup_26.x | sudo -E bash -
sudo apt-get install -y nodejs

# Verificar — DEBE mostrar v26.x.x
node --version
npm --version
```

> **Versión crítica**
>
> Node.js en el servidor debe ser la misma rama (26.x) con la que se generó `node_modules`. Si difieren, los binarios nativos fallan con "invalid ELF header".

6 Instalar PM2

PM2 mantiene el servicio en background, lo reinicia si se cae, y distribuye la carga entre un worker por CPU core.

```
sudo npm install -g pm2
pm2 --version
```

7 Crear la estructura de carpetas en el servidor

```
sudo mkdir -p /var/www/html/apibatch
sudo chown $USER:$USER /var/www/html/apibatch   # dueño = tu usuario, NO www-data

# Crear subcarpetas que no se transfieren
mkdir -p /var/www/html/apibatch/storage
mkdir -p /var/www/html/apibatch/data
mkdir -p /var/www/html/apibatch/logs
```

```
ls -la /var/www/html/apibatch/
```

Por qué el dueño es $USER y no www-data

Apache no lee los archivos del servicio — solo hace proxy de red a localhost:3100. Quien necesita acceso al sistema de archivos es el proceso Node.js lanzado por PM2 con tu usuario.

8 Transferir el proyecto al servidor

Opción A — rsync (recomendado para actualizaciones frecuentes)

Desde tu máquina local — transfiere solo lo que cambió:

```
cd /home/memyself/Downloads/fide-workspace/web-php/fide-api

rsync -avz --progress \
  --exclude='.env' \
  --exclude='storage/' \
  --exclude='data/' \
  --exclude='logs/' \
  --exclude='compile.sh' \
  --exclude='load_test.sh' \
  --exclude='load_test.js' \
  face-match-node/ \
  ubuntu@SERVER_IP:/var/www/html/apibatch/

# Con clave .pem:
rsync -avz --progress -e "ssh -i ~/mi-clave-cloud.pem" \
  --exclude='.env' --exclude='storage/' --exclude='data/' \
  --exclude='logs/' --exclude='compile.sh' --exclude='load_test.sh' --
exclude='load_test.js' \
  face-match-node/ ubuntu@SERVER_IP:/var/www/html/apibatch/
```

Opción B — SFTP

Usando FileZilla, WinSCP o cliente SFTP. Subir al directorio `/var/www/html/apibatch/` los siguientes archivos y carpetas:

Qué subir	Qué NO subir
`src/` (carpeta completa)	`.env` — se crea en el servidor
`node_modules/` (carpeta completa)	`storage/` — se crea en el servidor
`dashboard.html`	`data/` — se crea en el servidor
`package.json`	`compile.sh` — solo en dev
`ecosystem.config.js`	`load_test.sh` — solo en dev
`.env.example`	`load_test.js` — solo en dev
`.npmrc`	

Verificar que todo llegó

```
# En el servidor
ls /var/www/html/apibatch/node_modules/better-sqlite3/prebuilds/linux-x64.node
# Debe existir — confirma que el binario llegó intacto
```

9 Crear el .env de producción en el servidor

El `.env` se crea manualmente en el servidor. **Nunca viaja por red ni va en git.**

```
cp /var/www/html/apibatch/.env.example /var/www/html/apibatch/.env
nano /var/www/html/apibatch/.env
```

```
PORT=3100
NODE_ENV=production

APIKEY_MASTER=completar_con_el_valor_provisto_por_el_responsable

# IPs de los sistemas que van a llamar al endpoint /v3/face-match/
AUTHORIZED_IPS=["127.0.0.1","::1","IP_DEL_CONSUMIDOR"]

RENAPER_TOKEN_URL=https://apirenaper.idear.gov.ar/CHUTROFINAL/API_ABIS/Autorizacion/
token.php
RENAPER_TCN_URL=https://apirenaper.idear.gov.ar/CHUTROFINAL/API_ABIS/apiInline_v3.php
RENAPER_RESULT_URL=https://apirenaper.idear.gov.ar/CHUTROFINAL/API_ABIS/resultadoTCN.php
RENAPER_USERNAME=USUARIO_RENAPER_PRODUCCION
RENAPER_PASSWORD=PASSWORD_RENAPER_PRODUCCION

# Rutas absolutas
STORAGE_DIR=/var/www/html/apibatch/storage
DB_PATH=/var/www/html/apibatch/data/fide-facematch.db

# Rate limiting — ajustar según acuerdo con RENAPER
RATE_LIMIT_PER_IP_PER_MIN=600
RATE_LIMIT_GLOBAL_PER_MIN=30000

# Dashboard — IPs autorizadas para acceder a /dashboard y /stats/*
# Agregar la IP de la oficina y/o VPN
DASHBOARD_ALLOWED_IPS=["127.0.0.1","TU_IP_DE_OFICINA"]

# Polling TCN
TCN_MAX_RETRIES=5
TCN_RETRY_DELAY_MS=500
```

```
# Proteger el .env — solo el dueño puede leerlo
chmod 600 /var/www/html/apibatch/.env
ls -la /var/www/html/apibatch/.env
# Debe mostrar: -rw------- (solo lectura/escritura para el dueño)
```

10 Arrancar el servicio con PM2

1 Verificar arranque manual primero

```
cd /var/www/html/apibatch
node src/server.js
```

Respuesta esperada: `{"msg":"Server listening at http://127.0.0.1:3100"}`. Si hay error de variable faltante, el mensaje dice cuál. Corregir en `.env` y reintentar. Detener con `Ctrl+C`.

2 Arrancar con PM2 en modo cluster

```
cd /var/www/html/apibatch
pm2 start ecosystem.config.js --env production
pm2 status
```

Una fila por CPU core del servidor, todas con status `online`.

3 Verificar que responde localmente

```
curl http://localhost:3100/health
# {"status":"ok","service":"fide-facematch-api","uptime":5.2}
```

11 Instalar y configurar Apache con SSL

1 Instalar Apache y habilitar módulos

```
sudo apt-get install -y apache2
sudo a2enmod proxy         # reenviar requests a otro servidor
sudo a2enmod proxy_http    # soporte HTTP en el proxy
sudo a2enmod headers       # modificar headers (pasar IP real del cliente)
sudo a2enmod ssl           # soporte HTTPS
sudo a2enmod rewrite       # redirección HTTP → HTTPS
sudo systemctl restart apache2
sudo systemctl status apache2   # Active: active (running)
```

2 Instalar Certbot y obtener certificado SSL

```
sudo apt-get install -y certbot python3-certbot-apache
sudo certbot --apache -d apibatch.fidelitas.com.ar
```

Certbot verifica que el dominio apunta al servidor — por eso el DNS debía estar configurado antes. El certificado se renueva automáticamente cada 90 días.

12 Configurar el VirtualHost para apibatch.fidelitas.com.ar

1 Crear el archivo de configuración

```
sudo nano /etc/apache2/sites-available/apibatch-fidelitas.conf
```

2 Pegar la configuración completa

```
# VirtualHost HTTP — redirige todo a HTTPS
<VirtualHost *:80>
    ServerName apibatch.fidelitas.com.ar
    RewriteEngine On
    RewriteRule ^(.*)$ https://%{HTTP_HOST}$1 [R=301,L]
</VirtualHost>

# VirtualHost HTTPS — proxy al microservicio Node.js
<VirtualHost *:443>
    ServerName apibatch.fidelitas.com.ar

    ErrorLog  ${APACHE_LOG_DIR}/apibatch-error.log
    CustomLog ${APACHE_LOG_DIR}/apibatch-access.log combined

    SSLEngine on
    SSLCertificateFile    /etc/letsencrypt/live/apibatch.fidelitas.com.ar/
fullchain.pem
    SSLCertificateKeyFile /etc/letsencrypt/live/apibatch.fidelitas.com.ar/
privkey.pem

    Header always set Strict-Transport-Security "max-age=31536000;
includeSubDomains"
    Header always set X-Content-Type-Options "nosniff"
    Header always set X-Frame-Options "DENY"

    ProxyPreserveHost On
    RequestHeader set X-Forwarded-For  "%{REMOTE_ADDR}s"
    RequestHeader set X-Real-IP        "%{REMOTE_ADDR}s"
    RequestHeader set X-Forwarded-Proto "https"

    # Endpoint principal de face-match
    ProxyPass        /v3/face-match/ http://127.0.0.1:3100/v3/face-match/
    ProxyPassReverse /v3/face-match/ http://127.0.0.1:3100/v3/face-match/

    # Dashboard de estadísticas — protegido por IP en Node.js
(DASHBOARD_ALLOWED_IPS)
    ProxyPass        /dashboard http://127.0.0.1:3100/dashboard
    ProxyPassReverse /dashboard http://127.0.0.1:3100/dashboard
    ProxyPass        /stats/    http://127.0.0.1:3100/stats/
    ProxyPassReverse /stats/    http://127.0.0.1:3100/stats/

    # Health check
    ProxyPass        /health http://127.0.0.1:3100/health
    ProxyPassReverse /health http://127.0.0.1:3100/health

    # Timeout generoso para polling de RENAPER (5 retries × 500ms + latencia)
    ProxyTimeout 30
</VirtualHost>
```

3 Habilitar el VirtualHost y recargar Apache

```
sudo a2dissite 000-default
sudo a2dissite 000-default-le-ssl 2>/dev/null || true
sudo a2ensite apibatch-fidelitas.conf

sudo apache2ctl configtest   # debe mostrar: Syntax OK
sudo systemctl reload apache2
```

13 Verificar que todo funciona end-to-end

Desde tu máquina local — probando a través de HTTPS como lo haría cualquier consumidor.

Test 1 — Health check (sin autenticación)

```
curl -s https://apibatch.fidelitas.com.ar/health | python3 -m json.tool
# {"status":"ok","service":"fide-facematch-api","uptime":120.5}
```

Si falla con	Causa	Solución
Could not resolve host	DNS no propagó	Esperar y reintentar
Connection refused	Apache no corre	`sudo systemctl status apache2`
502 Bad Gateway	Node.js no corre	`pm2 status`

Test 2 — Rechazo de apikey incorrecta → debe ser 401

```
curl -s -X POST https://apibatch.fidelitas.com.ar/v3/face-match/ \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "apikey=INCORRECTA&selfie_image=abc&nationalid=12345678&gender=M"
# {"Status":"ERROR","Message":"API key inválida o no provista.",...}
```

Test 3 — Dashboard (desde tu IP de oficina)

```
# En el browser:
https://apibatch.fidelitas.com.ar/dashboard

# Si da 403: tu IP no está en DASHBOARD_ALLOWED_IPS del .env del servidor
```

Test 4 — Request completo con imagen y credenciales RENAPER reales

```
cd /home/memyself/Downloads/fide-workspace/web-php/fide-api/face-match-node

curl -s -X POST https://apibatch.fidelitas.com.ar/v3/face-match/ \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "apikey=TU_APIKEY_PRODUCCION" \
  --data-urlencode "selfie_image=$(cat storage/selfie_test_b64.txt)" \
  --data-urlencode "nationalid=22222222" \
  --data-urlencode "gender=M" \
  | python3 -m json.tool
```

Test 5 — Verificar DB y archivos en el servidor

```
ls -la /var/www/html/apibatch/storage/
# TCN-xxx.txt  TCN-xxx_step1.json  TCN-xxx_step2.json
```

```
node -e "
const db=new(require('/var/www/html/apibatch/node_modules/better-sqlite3'))
  ('/var/www/html/apibatch/data/fide-facematch.db');
console.table(db.prepare(
  'SELECT id,tcn_id,national_id,status,score,time_total_ms,created_at FROM
face_match_results ORDER BY id DESC LIMIT 5'
).all());"
```

14 Configurar arranque automático con systemd

Para que el servicio arranque solo cuando el servidor se reinicia.

```
# Generar script de startup — el comando imprime el sudo que hay que ejecutar
pm2 startup systemd -u $USER --hp $HOME

# Copiar y ejecutar el comando sudo que imprimió (el tuyo puede ser diferente):
sudo env PATH=$PATH:/usr/bin /usr/lib/node_modules/pm2/bin/pm2 startup systemd -u ubuntu
--hp /home/ubuntu

# Guardar la lista de procesos para restaurar al reiniciar
pm2 save

# Verificar
sudo systemctl status pm2-$USER
# Active: active (running) + enabled
```

15 Troubleshooting — problemas frecuentes

Diagnóstico general rápido

```
pm2 status                              # ¿Node.js corre?
curl http://localhost:3100/health       # ¿Responde localmente?
sudo systemctl status apache2           # ¿Apache corre?
sudo tail -50 /var/log/apache2/apibatch-error.log
pm2 logs fide-facematch --lines 50
```

502 Bad Gateway

```
ss -tlnp | grep 3100
# Si no muestra nada → el servicio no corre
pm2 restart fide-facematch
```

Error "invalid ELF header"

`node_modules` compilado en OS diferente. En la estación de desarrollo: `./compile.sh` y volver a transferir.

403 en /dashboard

```
nano /var/www/html/apibatch/.env
# Agregar tu IP a DASHBOARD_ALLOWED_IPS
pm2 reload fide-facematch
```

La DB no se crea

```
ls -la /var/www/html/apibatch/data/     # dueño debe ser $USER, no root
sudo chown $USER:$USER /var/www/html/apibatch/data/
pm2 restart fide-facematch
# La DB se crea en el primer request exitoso, no al arranque
```

— Comandos de operación diaria

```
# Estado y logs
pm2 status
pm2 monit
pm2 logs fide-facematch
pm2 logs fide-facematch --lines 200

# Reinicio
pm2 reload fide-facematch    # zero-downtime (para deploys)
pm2 restart fide-facematch   # reinicio completo

# Cambiar configuración sin código
nano /var/www/html/apibatch/.env
pm2 reload fide-facematch

# Apache
sudo apache2ctl configtest
sudo systemctl reload apache2
sudo tail -50 /var/log/apache2/apibatch-error.log

# Health check
curl https://apibatch.fidelitas.com.ar/health

# Dashboard
# https://apibatch.fidelitas.com.ar/dashboard

# DB — últimas 10 operaciones
node -e "
const db=new(require('/var/www/html/apibatch/node_modules/better-sqlite3'))
  ('/var/www/html/apibatch/data/fide-facematch.db');
console.table(db.prepare(
  'SELECT id,tcn_id,national_id,status,score,time_total_ms,created_at FROM
face_match_results ORDER BY id DESC LIMIT 10'
).all());"
```

16 Prueba del circuito completo — imagen real vs imagen ficticia

16.1 Condiciones de la selfie real

Requisito	Detalle
Fondo	Liso y claro (blanco o gris) — sin objetos detrás
Posición	Frontal, rostro centrado en el encuadre
Iluminación	Uniforme, sin sombras sobre la cara
Expresión	Neutra, boca cerrada, ojos abiertos
Lentes / sombrero	Sin ninguno

Requisito	Detalle
Formato	JPEG · resolución mínima 400×400 px · máximo 5 MB

```
# Convertir la foto a base64
base64 -w 0 selfie_real.jpg > selfie_real_b64.txt
wc -c selfie_real_b64.txt   # no debe superar ~7 MB

# Prueba con imagen real — debe dar HIT si el rostro coincide con el DNI
curl -s -X POST https://apibatch.fidelitas.com.ar/v3/face-match/ \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "apikey=TU_APIKEY" \
  --data-urlencode "selfie_image=$(cat selfie_real_b64.txt)" \
  --data-urlencode "nationalid=TU_DNI" \
  --data-urlencode "gender=M" \
  | python3 -m json.tool
```

16.2 Prueba con imagen ficticia

```
# Generar imagen sintética con ImageMagick
convert \
  -size 400x500 xc:'#f0e6d2' \
  -fill '#d4a574' -draw "ellipse 200,220 100,130 0,360" \
  -fill '#1a0a00' -draw "ellipse 160,190 15,10 0,360" \
  -fill '#1a0a00' -draw "ellipse 240,190 15,10 0,360" \
  -fill '#8b5e3c' -draw "ellipse 200,250 25,8 0,360" \
  -quality 85 selfie_ficticia.jpg

base64 -w 0 selfie_ficticia.jpg > selfie_ficticia_b64.txt

curl -s -X POST https://apibatch.fidelitas.com.ar/v3/face-match/ \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "apikey=TU_APIKEY" \
  --data-urlencode "selfie_image=$(cat selfie_ficticia_b64.txt)" \
  --data-urlencode "nationalid=22222222" \
  --data-urlencode "gender=M" \
  | python3 -m json.tool
```

Con imagen ficticia RENAPER puede devolver `NO.HIT` con score bajo, o error de calidad de imagen. Ambas respuestas son correctas — confirman que el motor evalúa la imagen.

17 Rate limiting y prueba de carga

17.1 Dónde se configura — solo en el .env, sin tocar código

Variable en .env	Default	Qué controla
`RATE_LIMIT_PER_IP_PER_MIN`	600	Requests/min por IP individual — protección anti-abuso
`RATE_LIMIT_GLOBAL_PER_MIN`	30000	Techo global (500 rps × 60s) — negociar con RENAPER antes de subir

```
# Para ajustar — solo editar .env y recargar sin downtime
nano /var/www/html/apibatch/.env
pm2 reload fide-facematch
```

17.2 El cuello de botella real es RENAPER

El servicio Node.js puede manejar 500 rps sin problema. El limitante real es RENAPER: cada request hace **2 llamadas HTTP externas síncronas** (TCN + resultado). El pool de conexiones está en `src/services/renaperService.js`:

```
const renaperAgent = new Agent({
  connections: 50,  // máximo 50 llamadas simultáneas a RENAPER
});
```

Aumentar `connections` solo después de confirmar con RENAPER cuánta concurrencia permite.

17.3 Script de prueba de carga — load_test.sh

Script bash en la estación de desarrollo. Lee su configuración del `.env` del proyecto — no hay que editar el script.

```
# Variables relevantes en .env (estación de desarrollo):
# LOAD_TEST_RPS=50
# LOAD_TEST_DURATION=5
# LOAD_TEST_URL=http://localhost:3100   ← cambiar a la URL del cloud para probar allá
# LOAD_TEST_NATIONALID=22222222
# LOAD_TEST_GENDER=M

# Ejecución
cd /home/memyself/Downloads/fide-workspace/web-php/fide-api/face-match-node
bash load_test.sh

# Override puntual sin editar .env
LOAD_TEST_URL=https://apibatch.fidelitas.com.ar bash load_test.sh
LOAD_TEST_RPS=100 LOAD_TEST_DURATION=10 bash load_test.sh
```

Resultado de ejemplo — 50 rps × 5s

```
═══════════════════════════════════════════════════════
  RESULTADO PRUEBA DE CARGA — face-match-node
═══════════════════════════════════════════════════════
  Enviados     : 250    Completados  : 250
  Errores red  : 0      Rate limited : 0 (HTTP 429)
  Throughput   : 48.1 rps reales
───────────────────────────────────────────────────────
  Latencia (ms):
    Promedio   : 138.2 ms     p50 : 105.7 ms
    p95        : 327.3 ms     p99 : 392.8 ms
───────────────────────────────────────────────────────
  Códigos HTTP :
    502 : 250 (100%)  ← esperado sin credenciales RENAPER reales
═══════════════════════════════════════════════════════
```

Métrica	Qué indica	Valor de alerta
`Errores red`	Conexiones rechazadas o timeout	> 0
`Rate limited`	El rate limit se activó	> 0 → subir límite o reducir rps
`Throughput`	rps reales alcanzados	Debe ser ≈ al objetivo
`p95`	Latencia del 95% de los requests	> 2000ms con RENAPER real
Código `429`	Rate limit activado	Ajustar `.env`
Código `200`	Requests exitosos	Debe ser 100% con credenciales reales