FIDELITAS S.A. · API PLATFORM

Runbook de Instalación y Operación

Microservicio face-match-node · v1.1

Guía completa para desarrolladores junior: compilación en estación de desarrollo con Podman (Ubuntu 24.04 container), transferencia al servidor, configuración de Apache, medición de performance y recuperación de base de datos SQLite.

FECHA
2026-11-05

AUDIENCIA
Developer Junior

PROYECTO
face-match-node · fide-api

STACK
Node.js 20 · Fastify 5 · SQLite WAL · PM2 · Apache

CONFIDENCIAL — USO INTERNO

CÓDIGO	VERSIÓN	FECHA	ESTADO	EMISOR	APROBADOR
DOC-OPS-006	1.1	2026-11-05	VIGENTE	Equipo de Tecnología	Dirección de Tecnología

1 Qué es este servicio y qué hace

El servicio `face-match-node` es un **microservicio Node.js** que actúa como proxy seguro entre los consumidores de la API de Fidelitas y la API de reconocimiento facial de RENAPER.

```
Consumidor (Postman / app cliente)
        │  POST /v3/face-match/
        │  apikey + selfie_image (base64) + nationalid + gender
        ▼
Apache 443 (HTTPS — TLS termina acá)
        │  ProxyPass → localhost:3100
        ▼
face-match-node (puerto 3100 interno)
        │  1. Valida apikey y IP
        │  2. Obtiene token RENAPER (cache o nuevo)
        │  3. Envía imagen → recibe TCN
        │  4. Consulta resultado con TCN (polling)
        │  5. Guarda archivos + SQLite
        ▼
Respuesta JSON unificada al consumidor
```

Archivo / Registro	Contenido
`storage/TCN-xxx.txt`	Imagen selfie en base64
`storage/TCN-xxx_step1.json`	Respuesta RENAPER paso 1 (TCN)
`storage/TCN-xxx_step2.json`	Respuesta RENAPER paso 2 (resultado)
`data/fide-facematch.db`	SQLite: todos los campos + timings en ms

2 Prerequisitos en la estación de desarrollo

Herramienta	Versión mínima	Verificación
Node.js	26.x LTS	`node --version` → v26.x.x
npm	10.x	`npm --version` → 10.x.x
Podman	cualquiera	`podman --version`
git / ssh / rsync	cualquiera	`git --version`

```
# Instalar Node.js 26 en Fedora 44
curl -fsSL https://rpm.nodesource.com/setup_26.x | sudo bash -
sudo dnf install -y nodejs
node --version   # v26.x.x
```

```
npm --version    # 10.x.x
# Podman ya viene instalado en Fedora 44 Golden ISO
```

3 Clonar el repositorio y preparar el código local

1 Actualizar el repo

```
cd ~/Downloads/fide-workspace/web-php
git pull origin master
```

El código vive en: `web-php/fide-api/face-match-node/`

2 Crear el .env local

```
cd web-php/fide-api/face-match-node
cp .env.example .env
nano .env
```

```
PORT=3100
NODE_ENV=development
APIKEY_MASTER=completar_con_el_valor_provisto_por_el_responsable
AUTHORIZED_IPS=["127.0.0.1","::1"]
RENAPER_USERNAME=tu_usuario_renaper
RENAPER_PASSWORD=tu_password_renaper
```

Prohibido commitear el .env

Nunca va en git. El hook pre-push lo detecta y bloquea automáticamente.

4 Compilar el proyecto en la estación de desarrollo

> **Regla Fidelitas (§19 CONTEXTO) — el servidor NO compila nada**
>
> Los servidores de producción no tienen compiladores ni toolchains instalados. Todo el código llega compilado y listo para ejecutar. La compilación ocurre en la estación de desarrollo dentro de un contenedor Podman que replica exactamente el sistema operativo del servidor.

Por qué compilar en un contenedor Ubuntu 24.04

`better-sqlite3` contiene código C++ que genera un binario específico para el OS y la versión de Node.js de destino. Si compilás en Fedora 44 y ejecutás en Ubuntu 24.04, el binario no funciona. La imagen `node:26-bookworm-slim` (Debian Bookworm, base de Ubuntu 24.04) produce binarios 100% compatibles con el servidor.

Qué instala — solo las 7 dependencias de producción estrictamente necesarias

Paquete	Para qué sirve
`fastify`	Framework HTTP (el servidor web Node.js)
`@fastify/formbody`	Parsear formularios application/x-www-form-urlencoded
`@fastify/helmet`	11 headers de seguridad HTTP automáticos
`@fastify/rate-limit`	Límite de requests por IP — protección contra abuso
`better-sqlite3`	SQLite WAL integrado — rápido, sin servidor, sin overhead
`undici`	Cliente HTTP para llamar a RENAPER — el más rápido de Node.js
`dotenv`	Leer variables del archivo .env

Las dependencias de desarrollo (`--omit=dev`) **no se instalan**. El `.npmrc` del proyecto las excluye por defecto.

1 Ejecutar la compilación

```
cd ~/Downloads/fide-workspace/web-php/fide-api/face-match-node
./compile.sh
```

La primera vez descarga la imagen (≈200 MB) y tarda 2-3 minutos. Las siguientes veces usa caché de Podman y tarda menos de un minuto.

Al terminar: `⟹ Compilación exitosa — listo para transferir al servidor.`

2 Verificar que el binario nativo compiló

```
# Debe existir este archivo .node (binario C++ de SQLite)
ls node_modules/better-sqlite3/prebuilds/linux-x64.node
```

Si existe, la compilación fue exitosa.

3 Verificar arranque local (opcional, para validar lógica)

```
# Instalar también para Fedora (para correr en dev local)
npm install
node src/server.js
```

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

curl -X POST http://localhost:3100/v3/face-match/ \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "apikey=WRONG&selfie_image=abc&nationalid=12345678&gender=M"
# {"Status":"ERROR","Message":"API key inválida..."}
```

Detener con `Ctrl+C`.

5 Empaquetar y transferir al servidor

El `node_modules/` generado por `compile.sh` ya está compilado para Ubuntu 24.04. Se transfiere junto con el código fuente — el servidor solo tiene que ejecutarlo.

1 Crear la carpeta destino en el servidor

```
ssh aburgol@10.10.30.50
sudo mkdir -p /var/www/html/apibatch
sudo chown aburgol:aburgol /var/www/html/apibatch
exit
```

2 Transferir código + node_modules compilado con rsync

```
cd ~/Downloads/fide-workspace/web-php/fide-api

rsync -avz --progress \
  --exclude='.env' \
  --exclude='storage/' \
  --exclude='data/' \
  --exclude='logs/' \
  face-match-node/ \
  aburgol@10.10.30.50:/var/www/html/apibatch/
```

Qué se transfiere: código fuente + `node_modules/` compilado para Ubuntu 24.04.
Qué NO se transfiere: `.env` (nunca viaja), `storage/` y `data/` (se crean en el servidor).

> **El servidor no necesita compilar nada**
> Una vez que rsync termina, el servicio está listo para arrancar. No hay `npm install` ni `build-essential` ni compiladores en el servidor.

6 Prerequisitos en el servidor Ubuntu 24.04

El servidor **no instala compiladores ni toolchains**. Los prerequisitos son mínimos: Node.js para ejecutar, PM2 para gestionar el proceso, y Apache para el proxy.

```
ssh aburgol@10.10.30.50
```

6.1 Actualizar el sistema operativo

```
sudo apt-get update
sudo apt-get upgrade -y
```

6.2 Instalar Node.js 26 LTS (solo para ejecutar — no para compilar)

```
curl -fsSL https://deb.nodesource.com/setup_26.x | sudo -E bash -
sudo apt-get install -y nodejs
node --version    # debe coincidir con la rama usada en compile.sh (v26.x.x)
```

6.3 Instalar PM2

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

6.4 Instalar Apache y módulos de proxy

```
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       # manipular headers HTTP
sudo a2enmod ssl           # soporte HTTPS
sudo systemctl restart apache2
sudo systemctl status apache2   # Active: active (running)
```

6.5 Crear directorios del servicio

```
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/
```

7 Desplegar el servicio en el servidor

1 Crear el .env de producción

```
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
AUTHORIZED_IPS=["10.10.20.5","10.10.30.10"]
RENAPER_USERNAME=usuario_produccion
RENAPER_PASSWORD=password_produccion
STORAGE_DIR=/var/www/html/apibatch/storage
DB_PATH=/var/www/html/apibatch/data/fide-facematch.db
TCN_MAX_RETRIES=5
TCN_RETRY_DELAY_MS=500
```

2 Verificar arranque manual

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

Si arranca sin errores, detener con `Ctrl+C`. Los errores de configuración aparecen acá.

3 Arrancar con PM2

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

Deberías ver una fila por cada CPU core del servidor, todas con status `online`.

4 Configurar arranque automático

```
pm2 startup systemd -u $USER --hp $HOME
# Copiar y ejecutar el comando sudo que imprime el paso anterior
# Ejemplo: sudo env PATH=$PATH:/usr/bin pm2 startup systemd -u aburgol --hp /
home/aburgol
pm2 save
```

Flujo de actualización (próximos deploys)

```
# En la estación de desarrollo:
cd ~/Downloads/fide-workspace/web-php/fide-api/face-match-node
```

```
./compile.sh   # solo si cambió package.json

rsync -avz --progress --exclude='.env' --exclude='storage/' --exclude='data/' --
exclude='logs/' \
  . aburgol@10.10.30.50:/var/www/html/apibatch/

# En el servidor — zero-downtime reload:
pm2 reload fide-facematch
```

8 Configurar Apache como reverse proxy

1 Encontrar el VirtualHost correcto

```
ls /etc/apache2/sites-enabled/
sudo cat /etc/apache2/sites-enabled/fide-api.conf
```

2 Agregar ProxyPass dentro del VirtualHost 443

```
sudo nano /etc/apache2/sites-enabled/NOMBRE.conf
```

Agregar dentro de `<VirtualHost *:443>`, antes del cierre `</VirtualHost>`:

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

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/

ProxyPass        /face-match-health http://127.0.0.1:3100/health
ProxyPassReverse /face-match-health http://127.0.0.1:3100/health
```

3 Verificar sintaxis y recargar sin downtime

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

# Verificar que el proxy funciona
curl -sk https://api.fidelitas.com.ar/face-match-health
# {"status":"ok","service":"fide-facematch-api","uptime":300}
```

9 Verificar que todo funciona

Test 1 — apikey incorrecta → debe responder 401

```
curl -X POST https://api.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.","Response":
[],"Transaction_id":null}
```

Test 2 — campos faltantes → debe responder 400

```
curl -X POST https://api.fidelitas.com.ar/v3/face-match/ \
  -H "Content-Type: application/x-www-form-urlencoded" \
  -d "apikey=TU_APIKEY_REAL&nationalid=12345678&gender=M"
```

Test 3 — request completo con imagen real

```
base64 -w 0 foto_dni.jpg > foto_b64.txt
SELFIE=$(cat foto_b64.txt)
curl -X POST https://api.fidelitas.com.ar/v3/face-match/ \
  -H "Content-Type: application/x-www-form-urlencoded" \
  --data-urlencode "apikey=TU_APIKEY_REAL" \
  --data-urlencode "selfie_image=$SELFIE" \
  --data-urlencode "nationalid=22222222" \
  --data-urlencode "gender=M"
```

Test 4 — verificar persistencia

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

sqlite3 /var/www/html/apibatch/data/fide-facematch.db \
  "SELECT tcn_id, status, score, time_total_ms, created_at FROM face_match_results ORDER
BY id DESC LIMIT 5;"
```

10 Medir performance y consultar métricas

10.1 Instalar Apache Bench

```
sudo apt-get install -y apache2-utils
```

10.2 Benchmark de health check (throughput máximo teórico)

```
ab -n 10000 -c 100 -k https://api.fidelitas.com.ar/face-match-health
```

Métrica	Qué significa	Valor objetivo
Requests per second	Throughput — cuántos req/seg puede manejar	> 500
Time per request (mean)	Latencia promedio por request en ms	< 200 ms
Percentil 50%	La mitad de los usuarios ve este tiempo	< 100 ms
Percentil 99%	El 1% más lento	< 500 ms
Failed requests	Requests que fallaron	0

10.3 Monitorear recursos durante la carga

```
pm2 monit           # monitor interactivo CPU/RAM de cada worker
htop                # estadísticas del sistema en tiempo real
ss -tnp | grep :3100  # conexiones activas hacia el servicio
```

10.4 Consultar timings reales desde SQLite

```
sqlite3 /var/www/html/apibatch/data/fide-facematch.db << 'EOF'
SELECT
  COUNT(*)                          AS total_operaciones,
  ROUND(AVG(time_token_ms), 0)      AS avg_token_ms,
  ROUND(AVG(time_step1_ms), 0)      AS avg_step1_renaper_ms,
  ROUND(AVG(time_step2_ms), 0)      AS avg_step2_resultado_ms,
  ROUND(AVG(time_total_ms), 0)      AS avg_total_ms,
  MAX(time_total_ms)                AS peor_caso_ms,
  MIN(time_total_ms)                AS mejor_caso_ms
FROM face_match_results
WHERE created_at ⩾ datetime('now', '-1 hour', 'localtime');
EOF
```

11 Recuperar la base de datos SQLite ante un fallo

11.1 Detectar corrupción

```
sqlite3 /var/www/html/apibatch/data/fide-facematch.db "PRAGMA integrity_check;"
# Respuesta normal: ok
```

11.2 Recuperar desde WAL journal (crash reciente)

```
pm2 stop fide-facematch
sqlite3 /var/www/html/apibatch/data/fide-facematch.db "PRAGMA wal_checkpoint(TRUNCATE);"
sqlite3 /var/www/html/apibatch/data/fide-facematch.db "PRAGMA integrity_check;"
pm2 start fide-facematch
```

11.3 Recuperar desde backup

```
pm2 stop fide-facematch
cp /var/www/html/apibatch/data/fide-facematch.db \
   /var/www/html/apibatch/data/fide-facematch.db.corrupto.$(date +%Y%m%d_%H%M%S)
cp /media/BACKUP_DISK/fide-facematch-db-FECHA.db \
   /var/www/html/apibatch/data/fide-facematch.db
sqlite3 /var/www/html/apibatch/data/fide-facematch.db "PRAGMA integrity_check;"
pm2 start fide-facematch
```

11.4 Script de backup diario automático

```
sudo nano /usr/local/bin/backup-facematch-db.sh
```

```
#!/bin/bash
FECHA=$(date +%Y%m%d_%H%M%S)
DB_ORIGEN="/var/www/html/apibatch/data/fide-facematch.db"
BACKUP_DIR="/backup/fide-facematch"
mkdir -p "$BACKUP_DIR"
sqlite3 "$DB_ORIGEN" ".backup $BACKUP_DIR/fide-facematch-db-$FECHA.db"
find "$BACKUP_DIR" -name "*.db" -mtime +30 -delete
echo "Backup: $BACKUP_DIR/fide-facematch-db-$FECHA.db"
```

```
sudo chmod +x /usr/local/bin/backup-facematch-db.sh
(crontab -l 2>/dev/null; echo "0 3 * * * /usr/local/bin/backup-facematch-db.sh") |
crontab -
```

12 Comandos de operación diaria — referencia rápida

Estado y logs

```
pm2 status                              # estado de todos los workers
pm2 show fide-facematch                 # detalles de un proceso
pm2 monit                               # monitor interactivo CPU/RAM
pm2 logs fide-facematch                 # logs en tiempo real
pm2 logs fide-facematch --lines 500     # últimas 500 líneas
pm2 flush fide-facematch                # limpiar logs acumulados
```

Ciclo de vida

```
pm2 reload fide-facematch    # zero-downtime reload (para deploys)
pm2 restart fide-facematch   # reinicio completo
pm2 stop fide-facematch
pm2 start ecosystem.config.js --env production
```

Verificación rápida

```
curl -s http://localhost:3100/health | python3 -m json.tool
```

SQLite — consultas frecuentes

```
# Últimas 10 operaciones
sqlite3 /var/www/html/apibatch/data/fide-facematch.db \
  "SELECT tcn_id, status, score, time_total_ms, created_at FROM face_match_results ORDER
BY id DESC LIMIT 10;"

# Total del día
sqlite3 /var/www/html/apibatch/data/fide-facematch.db \
  "SELECT COUNT(*) FROM face_match_results WHERE DATE(created_at) =
DATE('now','localtime');"
```

Apache

```
sudo apache2ctl configtest              # verificar sintaxis antes de recargar
sudo systemctl reload apache2
sudo tail -50 /var/log/apache2/error.log
```

Backup manual

```
/usr/local/bin/backup-facematch-db.sh
```