

SensIA - Sensor Inteligente de Anonimización

Manual técnico y de despliegue

Versión 1.0 · Aplicable a SensIA 1.0.0-rc.1

Índice

- Control de versiones
- 1. Introducción
  - 1.1. Objeto y destinatarios
  - 1.2. Estado del proyecto
  - 1.3. Documentación relacionada
- 2. Arquitectura
  - 2.1. Componentes
  - 2.2. Flujo de procesamiento
  - 2.3. Comunicaciones y puertos
  - 2.4. Datos almacenados
- 3. Requisitos
  - 3.1. Hardware
  - 3.2. Software
  - 3.3. Red, dominio y certificados
- 4. Instalación con Docker Compose
  - 4.1. Instalación de Docker
  - 4.2. Preparación del directorio
  - 4.3. Obtención de SensIA
  - 4.4. Configuración del entorno
  - 4.5. Arranque
  - 4.6. Comprobaciones tras la instalación
- 5. Configuración
  - 5.1. Variables de entorno
  - 5.2. Servidor del motor
  - 5.3. Valores por defecto del procesamiento
  - 5.4. Patrones de detección
  - 5.5. Lista blanca global
  - 5.6. Modelos y diccionarios
  - 5.7. Reconocimiento de entidades
  - 5.8. Correo electrónico
  - 5.9. Observabilidad
- 6. Seguridad
  - 6.1. Exposición de red
  - 6.2. Certificados y autenticación mutua
  - 6.3. Secretos
  - 6.4. Protección de la aplicación
  - 6.5. Datos personales tratados
  - 6.6. Desarrollo seguro y actualizaciones de seguridad
- 7. Operación y mantenimiento
  - 7.1. Comandos habituales
  - 7.2. Logs
  - 7.3. Modo mantenimiento
  - 7.4. Copias de seguridad
  - 7.5. Actualización
  - 7.6. Actualización de modelos y diccionarios
  - 7.7. Resolución de problemas
  - 7.8. Consideraciones de operación
- 8. Limitaciones conocidas

- Anexo A. Requisitos funcionales y no funcionales
  - A.1. Requisitos funcionales
    - A.1.1. Entrada de documentos
    - A.1.2. Reconocimiento óptico de caracteres (OCR)
    - A.1.3. Extracción de texto
    - A.1.4. Detección de datos
    - A.1.5. Códigos QR y de barras
    - A.1.6. Imágenes y rostros
    - A.1.7. Ocultación y seudonimización
    - A.1.8. Vista previa y revisión
    - A.1.9. Sanitización
    - A.1.10. Aplicación web
    - A.1.11. Servicio web
  - A.2. Requisitos no funcionales
    - A.2.1. Arquitectura y despliegue
    - A.2.2. Seguridad
    - A.2.3. Rendimiento
    - A.2.4. Disponibilidad
    - A.2.5. Observabilidad y trazabilidad
    - A.2.6. Interoperabilidad
    - A.2.7. Protección de datos
    - A.2.8. Marco legal
- Anexo B. Especificación del servicio web SOAP
  - B.1. Datos generales
  - B.2. Autenticación
  - B.3. Operaciones
  - B.4. Aspectos relevantes para la administración
  - B.5. Comprobación del servicio
  - B.6. Información para los sistemas integradores

Control de versiones

Versión	Fecha	Descripción de los cambios
1.0	01/10/2026	Primera edición pública, aplicable a SensIA 1.0.0-rc.1.

1. Introducción

1.1. Objeto y destinatarios

Este documento describe la arquitectura de SensIA y los pasos necesarios para instalarlo, configurarlo y mantenerlo. Incluye, además, un resumen del servicio web SOAP orientado a su administración. El contrato completo y los ejemplos de integración se describen en el *Manual del integrador*.

Está dirigido al personal técnico responsable de sistemas, infraestructura y seguridad del organismo que implanta la herramienta, así como al personal técnico de los sistemas que se integren con ella.

1.2. Estado del proyecto

SensIA 1.0.0-rc.1 es una versión candidata. Se ha probado en un piloto en un organismo público y está pendiente de despliegue generalizado. Durante el piloto ha mostrado una tasa de detección alta, aunque se han identificado omisiones en casos concretos. Por ello, se recomienda validarla en un entorno de preproducción, con documentos representativos del organismo, antes de su uso en producción. En todo caso, debe mantenerse la revisión humana de los resultados. Las limitaciones conocidas se resumen en el apartado 8.

SensIA se distribuye bajo licencia EUPL v1.2. Los componentes de terceros conservan sus propias licencias, relacionadas en `NOTICE.md`.

1.3. Documentación relacionada

Documento	Contenido
`README.md`	Descripción general, estado del proyecto, limitaciones conocidas y referencia completa de las variables de configuración.
`CHANGELOG.md`	Cambios de cada versión.
`SECURITY.md`	Política de seguridad y comunicación de vulnerabilidades.
`NOTICE.md`	Componentes de terceros y sus licencias.
`.env.example`	Variables de configuración y valores por defecto del asistente de instalación.
Contrato WSDL	Definición formal del servicio SOAP, publicada por el propio servicio en `https://<dominio>/services/SensiaService?wsdl`. Se resume en el Anexo B y se describe en detalle en el *Manual del integrador*.
Manual de usuario	Uso de la aplicación web.
Manual del integrador	Contrato completo del servicio web SOAP, opciones de procesamiento y ejemplos de integración.

2. Arquitectura

2.1. Componentes

SensIA se compone de servicios independientes que se ejecutan en contenedores Docker. Los servicios se conectan por una red interna dedicada (`sensia_net`). El proxy inverso es el único punto de entrada; ningún otro servicio publica puertos hacia el exterior.

Componente	Tecnología	Función
Proxy inverso (`proxy-nginx`)	NGINX 1.30	Es el único punto de entrada. Termina el TLS de la aplicación web y del servicio SOAP con token, y aplica las cabeceras de seguridad. Redirige HTTP a HTTPS. Reenvía sin descifrar (*passthrough*) el puerto del servicio SOAP con certificado de cliente.
`frontend-app`	Angular 21, servida con Caddy	Aplicación web para usuarios y administradores.
`backend-app`	Node.js 24, SQLite	Gestiona usuarios, autenticación, preferencias y plantillas. Envía el correo y se comunica con el motor mediante el servicio SOAP.
`backend-ws`	Java 21, Jetty 12, Apache CXF	Motor de procesamiento: extracción de texto, OCR, detección, ocultación y sanitización. Expone el servicio web SOAP.
`backend-ner`	Python 3.12, ONNX Runtime, gRPC	Reconocimiento de entidades nombradas con el modelo público `BSC-LT/roberta_model_for_anonimization` del BSC-CNS. Se despliega siempre; su uso se decide en cada petición.
Observabilidad (opcional)	OpenTelemetry, Grafana, Loki, Tempo, Alloy	Trazas y logs centralizados. Se activa con el perfil `obs` .

El motor utiliza, entre otras, las siguientes librerías:

- Apache PDFBox, para el tratamiento de PDF.
- Tesseract OCR 4 mediante Tess4J, para el OCR.
- OpenCV 4.13.0 con el modelo YuNet, para la detección facial.
- ZXing, para los códigos QR y de barras.

Todos los contenedores están configurados con la política de reinicio `unless-stopped` . De este modo, se reinician automáticamente tras un fallo o un reinicio del servidor. Además, cuentan con estas medidas:

- Se ejecutan sin privilegios de root, salvo NGINX.
- Usan la opción `no-new-privileges` .
- No disponen de capacidades del kernel innecesarias.
- Tienen límites de memoria.

2.2. Flujo de procesamiento

1. La aplicación web, a través de `backend-app` , o un sistema integrador envía el documento al motor mediante la operación `submitBatch` del servicio SOAP.

2. El motor registra un trabajo asíncrono por cada documento y devuelve un identificador de seguimiento (*ticket*). El estado y el resultado se consultan con la operación `getStatus`.
3. El motor extrae el texto del PDF. Si la primera página contiene menos de 10 caracteres extraíbles, considera que el documento está escaneado y aplica OCR con Tesseract.
4. Detecta datos personales mediante:
    - los patrones configurados;
    - el diccionario de nombres;
    - el reconocimiento de entidades, si se ha solicitado;
    - la lectura de códigos QR y de barras;
    - la detección de rostros en las imágenes.
5. Aplica las listas blancas y las zonas de exclusión, y elimina las detecciones duplicadas o solapadas.
6. Según el modo solicitado:
    - **Vista previa:** genera un documento con contornos de color, sin ocultar nada. El rojo señala los elementos que se ocultarían y el verde, las excepciones y zonas protegidas.
    - **Definitivo:** oculta las detecciones con cajas opacas, negras en general y blancas en las zonas fijas. Si se ha solicitado, sustituye los datos por seudónimos. Además, elimina imágenes y oculta los rostros con un mosaico o un relleno sólido.
7. Sanitiza el resultado convirtiendo cada página en imagen. Así, el documento final no conserva capa de texto, capas ocultas ni metadatos del original.

Cuando un documento procesado con vista previa se confirma tras la revisión, el motor aplica directamente las marcas confirmadas, sin repetir la detección. Antes, comprueba mediante una huella que las marcas corresponden al mismo documento original (*Manual del integrador*, apartado 7).

Las operaciones `redactImage` (ocultación de rostros en una imagen) y `extractImages` (extracción de las imágenes de un PDF) son síncronas y devuelven el resultado en la propia respuesta.

2.3. Comunicaciones y puertos

Origen y destino	Protocolo	Puerto	Exposición
Usuarios e integradores → proxy	HTTP	80	Externa. Solo redirige la web a HTTPS y atiende la validación de Let's Encrypt. Las rutas `/api/` y `/services/` se rechazan con HTTP 403.
Usuarios e integradores → proxy	HTTPS	443	Externa
Integradores con certificado de cliente → proxy → `backend-ws`	HTTPS con mTLS, de extremo a extremo (el proxy no descifra la conexión)	8443	Externa, solo si se utilizan integraciones con mTLS
Proxy → `frontend-app`	HTTP	80	Interna
Proxy → `backend-app`	HTTP (REST/JSON)	3000	Interna
Proxy → `backend-ws`	HTTPS (SOAP con token)	443	Interna
`backend-app` → `backend-ws`	HTTPS (SOAP con token)	443	Interna

`backend-ws` → `backend-ner`	gRPC con mTLS (activo por defecto)	50051	Interna
Servicios → Tempo	OTLP	4317 (gRPC) y 4318 (HTTP)	Interna, solo con la observabilidad activa
Administradores → Grafana	HTTP	3030	Solo en el propio servidor (`127.0.0.1`). Se accede mediante un túnel SSH (apartado 5.9).

Los puertos publicados 80, 443 y 8443 son los valores por defecto. Pueden modificarse en el asistente de instalación mediante las variables `PORT_PROXY_HTTP` , `PORT_PROXY_HTTPS` y `PORT_WS_MTLS` .

`backend-app` se dirige al motor por el nombre del dominio. Dentro de la red interna, ese nombre se resuelve directamente al contenedor `backend-ws` . Por este motivo, el motor presenta también el certificado del dominio, tanto en el puerto 443 interno como en el 8443.

2.4. Datos almacenados

Dato	Ubicación	Observaciones
Usuarios, preferencias, plantillas e histórico	Base de datos SQLite, volumen `sensia_backend-db`	Contiene datos personales de los usuarios de la aplicación.
Documentos subidos y resultados	No se almacenan de forma permanente en el servidor. Durante el procesamiento se mantienen como archivos temporales del motor.	Contienen datos personales de terceros.
Archivos temporales del motor	Directorio `java.io.tmpdir` del contenedor `backend-ws`	Se eliminan al finalizar su periodo de retención y en la parada controlada del servicio.
Logs	Volumen `sensia_backend-logs` (`backend-app`), ficheros internos de `backend-ws` (`/app/logs` , máximo 200 MB), logs de Docker y, con la observabilidad activa, Loki (30 días)	Pueden contener fragmentos de los documentos (apartado 6.5).
Secretos	Fichero `.env` y sus copias `.env.bak.*`	Credenciales y claves generadas en la instalación.
Certificados y almacenes de claves	`/opt/sensia/pki`	Un subdirectorio por servicio. Las claves de las CA (`ca/`) y los certificados de cliente (`clientes/`) no se montan en ningún contenedor.
Certificados de Let's Encrypt	Volumen `certbot-data`	Solo si se utiliza Let's Encrypt. Contiene la cuenta y los certificados emitidos.

3. Requisitos

3.1. Hardware

El despliegue se ha probado en un servidor `amd64` con 4 vCPU, 8 GB de RAM y Ubuntu 26.04.

Recurso	Mínimo	Recomendado
CPU	4 vCPU	Más núcleos para cargas concurrentes
Memoria	8 GB	16 GB si se activa la observabilidad
Disco	30 GB libres, necesarios para construir las imágenes	100 GB, con espacio adicional para logs, copias de seguridad y actualizaciones

Los límites de memoria de los contenedores se definen en `docker-compose.yml`:

- `backend-ws`: 3 GB.
- `backend-ner`: 2 GB.
- `backend-app`: 1 GB.
- Proxy y aplicación web: 256 MB cada uno.

El servicio `backend-ner` se ejecuta siempre, aunque no se solicite el reconocimiento de entidades.

El OCR y el análisis de imágenes hacen un uso intensivo de CPU. La capacidad de procesamiento simultáneo depende principalmente del número de núcleos.

3.2. Software

- Linux `amd64`. Probado en Ubuntu Server / Minimal 26.04.
- Docker Engine reciente y Docker Compose v2.24 o superior.
- Git, OpenSSL y curl.
- `flock` (paquete `util-linux`), solo si se utiliza Let's Encrypt.
- OpenSSL 3.2 o superior, para que el asistente genere el almacén de confianza del motor. Las versiones actuales de Ubuntu cumplen este requisito, que puede comprobarse con `openssl version`. Con versiones anteriores, el asistente necesita la utilidad `keytool` (paquete `default-jre-headless`).

3.3. Red, dominio y certificados

- **Dominio.** Un nombre de dominio para SensIA (por ejemplo, `sensia.tuorganismo.es`) que resuelva a la dirección del servidor.
- **Certificado TLS** para ese dominio, mediante una de estas tres opciones:
  - Un certificado del organismo o de una autoridad pública, en formato PEM, con la cadena intermedia y la clave privada sin contraseña.
  - Un certificado autofirmado con la CA interna que genera el asistente, adecuado para pruebas o intranet.
  - Un certificado de Let's Encrypt. En este caso, el dominio debe resolver públicamente al servidor y el puerto 80 debe ser accesible desde Internet.

- **Acceso a Internet durante la construcción de las imágenes**, o a réplicas internas, para descargar dependencias desde GitHub, Hugging Face, npm, Maven Central, PyPI, download.pytorch.org, Docker Hub y los repositorios de Ubuntu. En redes con proxy corporativo, debe configurarse el proxy tanto para el servicio Docker como para la construcción de imágenes. En ejecución, SensIA no necesita acceso a Internet, salvo para Let's Encrypt y para el servidor de correo, si se utilizan.
- **Rango de direcciones para las redes internas de Docker**, acordado con el equipo de red del organismo (apartado 4.1).
- **Servidor SMTP**, si se va a utilizar el envío de correo (por ejemplo, para la recuperación de contraseñas).

4. Instalación con Docker Compose

4.1. Instalación de Docker

Si el servidor no dispone de Docker, puede instalarse con los siguientes comandos o siguiendo las instrucciones oficiales de Docker para la distribución utilizada:

```
sudo apt update && sudo apt upgrade -y
sudo apt install -y ca-certificates curl gnupg git openssl

sudo install -m 0755 -d /etc/apt/keyrings
curl -fsSL https://download.docker.com/linux/ubuntu/gpg \
  | sudo gpg --dearmor -o /etc/apt/keyrings/docker.gpg

echo "deb [arch=$(dpkg --print-architecture) signed-by=/etc/apt/keyrings/docker.gpg] \
https://download.docker.com/linux/ubuntu $(. /etc/os-release && echo "$VERSION_CODENAME") stable" \
  | sudo tee /etc/apt/sources.list.d/docker.list > /dev/null

sudo apt update
sudo apt install -y docker-ce docker-ce-cli containerd.io \
  docker-buildx-plugin docker-compose-plugin
```

Subredes de Docker. Por defecto, Docker asigna subredes de los rangos 172.17.0.0 a 172.31.255.255 y 192.168.0.0/16. Si coinciden con rangos utilizados en el organismo (puestos de usuario, VPN o servidores), la aplicación dejará de responder desde esas redes. Antes del despliegue, conviene acordar con el equipo de red un rango reservado y fijarlo en `/etc/docker/daemon.json`. En el ejemplo siguiente, los rangos son ilustrativos y deben sustituirse por los asignados:

```
{
  "bip": "10.200.0.1/24",
  "default-address-pools": [
    { "base": "10.201.0.0/16", "size": 24 }
  ]
}
```

```
sudo systemctl restart docker
```

4.2. Preparación del directorio

Se recomienda instalar SensIA en `/opt/sensia` y administrarlo mediante un grupo específico:

```
sudo groupadd sensia-admins
sudo usermod -aG sensia-admins,docker "$USER"

sudo mkdir -p /opt/sensia
sudo chown root:sensia-admins /opt/sensia
sudo chmod 2775 /opt/sensia
```

Después de añadir el usuario a los grupos, debe cerrarse y abrirse de nuevo la sesión. Con el comando `id` puede comprobarse que el usuario pertenece a los grupos `docker` y `sensia-admins`. La pertenencia al grupo `docker` equivale en la práctica a privilegios de administrador del servidor, por lo que debe limitarse al personal autorizado.

Tras la instalación, la estructura del directorio será la siguiente:

```
/opt/sensia/
├── sensia.git/      código, docker-compose.yml y .env
├── pki/             certificados y claves (los genera env.sh)
│   ├── ca/          claves de las CA; no se monta en ningún contenedor
│   ├── clientes/    certificados de cliente para entregar; no se monta
│   ├── publico/     certificado del dominio (proxy y motor)
│   ├── ws/          almacenes de claves y cliente gRPC (backend-ws)
│   ├── ner/         servidor gRPC (backend-ner)
│   └── app/         certificado público de la CA interna (backend-app)
└── backups/         copias de seguridad (apartado 7.4)
```

Si se va a utilizar un certificado del organismo, basta con que sus ficheros sean accesibles en el servidor. El asistente solicita su ruta y los copia en `pki/publico`.

4.3. Obtención de SensIA

Se recomienda instalar una versión etiquetada:

```
cd /opt/sensia
git clone https://github.com/ctt-gob-es/sensia.git sensia.git
cd sensia.git
git checkout v1.0.0-rc.1
```

El repositorio debe clonarse dentro de `/opt/sensia`, porque el asistente crea la PKI en el directorio superior (`/opt/sensia/pki`).

4.4. Configuración del entorno

Si el organismo utiliza valores fijos (dominio, servidor de correo, puertos, etc.), puede ajustarlos previamente en `.env.example`. El asistente los propondrá como valores por defecto. Los secretos que figuren en ese fichero se ignoran siempre.

Ejecute el asistente de configuración:

```
cd /opt/sensia/sensia.git
./env.sh
```

El asistente comprueba las dependencias del sistema y solicita, por este orden:

1. **Dominio y puertos** publicados (HTTP y HTTPS).

2. **Autoridades de certificación.** Se generan dos CA, con una validez de 10 años, que se reutilizan en ejecuciones posteriores:
   - La CA interna firma el certificado autofirmado del dominio y los certificados internos.
   - La CA de clientes firma los certificados de cliente del servicio SOAP.
3. **Certificado HTTPS del dominio**, con tres opciones:
   - *Autofirmado* con la CA interna, para pruebas o intranet. Requiere importar `pki/ca/ca_interna.crt` como CA de confianza en los equipos que usen la web.
   - *Certificado del organismo*, en formato PEM con la cadena intermedia y la clave sin contraseña. El asistente comprueba que la clave corresponde al certificado, su caducidad, el nombre del dominio y la presencia de la cadena.
   - *Let's Encrypt*, que se obtiene con `certbot.sh`. Permite utilizar el entorno de pruebas (*staging*), cuyos certificados no aceptan los navegadores. La instalación no programa su renovación, que se realiza a mano (apartado 6.2).
4. **Certificados internos** para la comunicación gRPC entre `backend-ws` y `backend-ner`, con una validez de 825 días.
5. **Servicio SOAP con certificado de cliente (mTLS).** Si se activa, el asistente:
   - solicita el puerto;
   - genera el primer certificado de cliente (`cliente_mtls.p12` , con su contraseña en `cliente_mtls.p12.pass` y una validez de 365 días);
   - registra su huella en `AUTH_CERT_WHITELIST` ;
   - admite opcionalmente una CA adicional emisora de certificados de cliente, por ejemplo la del organismo.
6. **Almacenes de claves de Java** del motor:
   - `keystore.p12` , con el certificado del dominio;
   - `truststore.p12` , con las CA de clientes.
7. **Credenciales y secretos.** El asistente solicita el usuario administrador inicial y genera de forma aleatoria:
   - la contraseña del administrador;
   - el token del servicio SOAP;
   - las claves de sesión (JWT) y de verificación ALTCHA;
   - la contraseña de Grafana;
   - las contraseñas de los almacenes de claves.

   Si ya existen secretos de una instalación anterior, ofrece conservarlos.
8. **Correo saliente:** servidor SMTP con usuario y contraseña, OAuth2, relay del organismo sin autenticación, o sin correo (apartado 5.8).
9. **Observabilidad** (opcional).

Al finalizar, el asistente realiza estas operaciones:

1. Aplica los permisos de la PKI.
2. Muestra un resumen para su confirmación.
3. Escribe el fichero `.env` con permisos 640, tras guardar una copia del anterior en `.env.bak.<fecha>` .
4. Comprueba que Docker Compose acepta la configuración.

El asistente puede volver a ejecutarse en cualquier momento. Reutiliza las CA, el certificado de cliente y los secretos existentes, salvo que se solicite regenerarlos. En cada ejecución vuelve a generar el certificado autofirmado (si se ha elegido esa opción), los certificados internos y los almacenes de claves. Regenerar los secretos cierra todas las sesiones abiertas, y regenerar el token SOAP obliga a comunicar el nuevo a los integradores.

Las credenciales no se muestran por pantalla y deben consultarse en el fichero `.env`. Antes de continuar:

- Revise el contenido del fichero `.env`.
- Comunique por un canal seguro la contraseña inicial (`DEFAULT_ADMIN_PASSWORD`) al administrador y el token (`SOAP_API_TOKEN`) a los integradores.
- Si se ha activado el mTLS, entregue el certificado de cliente (`pki/clientes/cliente_mtls.p12`) y su contraseña (`cliente_mtls.p12.pass`) por canales distintos.

4.5. Arranque

Construya las imágenes y arranque los servicios:

```
cd /opt/sensia/sensia.git
docker compose up -d --build
```

La primera construcción compila OpenCV desde el código fuente y descarga y exporta el modelo de reconocimiento de entidades, por lo que puede prolongarse. Las siguientes reutilizan la caché de Docker.

Compruebe el estado de los servicios:

```
docker compose ps
docker compose logs -f
```

Todos los servicios deben aparecer como `healthy`. El motor espera a que `backend-ner` esté operativo antes de arrancar, por lo que la puesta en marcha completa puede tardar varios minutos.

Una vez iniciado, SensIA estará disponible en las siguientes direcciones:

```
Aplicación web:               https://sensia.tuorganismo.es
Servicio SOAP (token):        https://sensia.tuorganismo.es/services/SensiaService
Servicio SOAP (certificado):  https://sensia.tuorganismo.es:8443/services/SensiaService
Contrato WSDL:                https://sensia.tuorganismo.es/services/SensiaService?wsdl
```

4.6. Comprobaciones tras la instalación

1. Todos los servicios aparecen como `healthy` en `docker compose ps`.
2. La aplicación web es accesible y el navegador considera válido el certificado.
3. Es posible iniciar sesión con el administrador inicial. Cambie su contraseña en el primer acceso.
4. En el panel de administración, revise los controles globales: sitio activo, registro, recuperación de contraseña y captcha (apartado 6.4).
5. El acceso por HTTP se comporta como se espera y el contrato WSDL es accesible por HTTPS:

```
DOMINIO=sensia.tuorganismo.es

# HTTP: la web se redirige a HTTPS (301) y el servicio SOAP se rechaza (403)
curl -sI "http://$DOMINIO" | head -1
curl -s -o /dev/null -w '%{http_code}\n' "http://$DOMINIO/services/SensiaService?wsdl"

# El WSDL publicado anuncia la dirección HTTPS del servicio
curl -s "https://$DOMINIO/services/SensiaService?wsdl" | grep soap:address
```

Con certificado autofirmado, añada `--cacert /opt/sensia/pki/ca/ca_interna.crt` a las órdenes `curl` que usan HTTPS.

6. Si se ha activado el mTLS, el servicio responde en el puerto 8443 con el certificado de cliente:

```
curl -s --cert /opt/sensia/pki/clientes/cliente_mtls.crt \
     --key /opt/sensia/pki/clientes/cliente_mtls.key \
     "https://$DOMINIO:8443/services/SensiaService?wsdl" | head -3
```

7. Un documento de prueba con datos ficticios se procesa correctamente con vista previa, y el comparador muestra las marcas esperadas.
8. Si se ha configurado el correo, la recuperación de contraseña envía el mensaje.
9. El servicio SOAP responde a la consulta de comprobación con el token de la instalación (Anexo B.5).

5. Configuración

5.1. Variables de entorno

La configuración del despliegue y los secretos se definen en el fichero `.env`, que genera el asistente `env.sh`.

Se recomienda modificar la configuración volviendo a ejecutar el asistente. La edición manual del fichero debe reservarse para ajustes puntuales, como añadir huellas a `AUTH_CERT_WHITELIST`. Cualquier cambio se aplica con `docker compose up -d --force-recreate`; el comando `docker compose restart` no vuelve a leer el fichero `.env`.

La lista completa de variables figura en `.env.example`, y su relación con cada servicio, en el apartado «Referencia de configuración» del `README.md`. Las principales son:

Variable	Descripción
`DOMAIN_NAME`	Dominio de la instalación.
`PORT_PROXY_HTTP`, `PORT_PROXY_HTTPS`, `PORT_WS_MTLS`	Puertos publicados por el proxy (por defecto, 80, 443 y 8443).
`CERT_FILE`, `KEY_FILE`	Certificado y clave del dominio en `pki/publico`.
`REQUIRE_MTLS`	Activa la vía del servicio SOAP con certificado de cliente (8443) y cierra la del puerto 443 (`/services/` responde 403). La aplicación web no se ve afectada, porque `backend-app` llega al motor por la red interna.
`AUTH_CERT_WHITELIST`	Huellas SHA-256 de los certificados de cliente autorizados, separadas por `;`.
`SOAP_API_TOKEN`	Token de autenticación del servicio SOAP.
`DEFAULT_ADMIN_USER`, `DEFAULT_ADMIN_PASSWORD`	Administrador inicial. La contraseña solo se utiliza al crearlo por primera vez.
`JWT_SECRET`	Clave de firma de las sesiones de la aplicación. Si se cambia, se invalidan las sesiones activas.
`ALTCHA_HMAC_KEY`	Clave del sistema de verificación ALTCHA.
`ENFORCE_SECURITY`	Activa el mTLS en la comunicación gRPC entre el motor y `backend-ner`. Por defecto, `true`.
`ENV`	Debe valer `production`. El valor `development` desactiva el mTLS en `backend-ner`, lo que impide su comunicación con el motor mientras `ENFORCE_SECURITY` sea `true`.
`LOG_LEVEL`	Nivel de log de los servicios. Por defecto, `INFO`.
`HARD_MAINTENANCE`	Activa el modo de mantenimiento estricto (apartado 7.3).

`SMTP_*`, `EMAIL_FROM`, `OAUTH_*`	Correo saliente (apartado 5.8).
`COMPOSE_PROFILES`	El valor `obs` activa la observabilidad.
`GRAFANA_PASSWORD`	Contraseña inicial del administrador de Grafana.

Los parámetros del servidor del motor (`JETTY_*`) y los límites de tamaño de las peticiones no se definen en `.env`, sino en `docker-compose.yml` (apartado 5.2).

5.2. Servidor del motor

El servidor Jetty del motor se configura en `server.properties`. Cada parámetro puede sustituirse mediante una variable de entorno, cuyo nombre se forma en mayúsculas sustituyendo los puntos por guiones bajos (por ejemplo, `jetty.qos.suspendMs` → `JETTY_QOS_SUSPENDMS`).

En el despliegue con Docker Compose, estos valores se fijan en el entorno del servicio `backend-ws` de `docker-compose.yml`:

Variable	Valor en el despliegue	Descripción
`JETTY_HTTP_ENABLED`	`false`	Conector HTTP sin cifrar. Está desactivado; solo se utiliza en desarrollo.
`JETTY_HTTPS_ENABLED` / `JETTY_HTTPS_PORT`	`true` / `443`	Conector HTTPS interno del servicio SOAP con token. Lo utilizan el proxy y `backend-app`.
`JETTY_SSL_ENABLED` / `JETTY_SSL_PORT`	Valor de `REQUIRE_MTLS` / `8443`	Conector HTTPS con certificado de cliente. Solo se activa si se habilita el mTLS en el asistente.
`JETTY_SSL_REQUIRECLIENTAUTH`	`true`	Exige certificado de cliente en el conector 8443.
`JETTY_SSL_VALIDATEPEERCERTS`, `JETTY_SSL_ENABLECRLDP`, `JETTY_SSL_ENABLEOCSP`	`false`	No se consultan CRL ni OCSP. Un cliente se revoca retirando su huella de `AUTH_CERT_WHITELIST`.
`JETTY_THREADPOOL_MAXTHREADS` / `JETTY_THREADPOOL_MINTHREADS`	`200` / `10`	Número máximo y mínimo de hilos del servidor.
`JETTY_THREADPOOL_IDLETIMEOUTMS`	`60000`	Tiempo tras el que se liberan los hilos inactivos.
`JETTY_CONNECTOR_IDLETIMEOUTMS`	`30000`	Tiempo tras el que se cierran las conexiones inactivas.
`JETTY_QOS_MAXREQUESTS`	`8`	Número máximo de peticiones SOAP procesadas simultáneamente.
`JETTY_QOS_SUSPENDMS`	`30000`	Tiempo máximo de espera en cola. Transcurrido este tiempo, se responde con

		HTTP 503.
`JETTY_HTTP_MAXPOSTSIZE`	`73400320` (70 MB)	Tamaño máximo de la petición.

Fuera de Docker se aplican los valores de `server.properties`: HTTP activo en el puerto 80, HTTPS en el 7443, mTLS en el 8443, 20 peticiones simultáneas.

El contenedor del motor tiene un límite de memoria de 3 GB, y la máquina virtual de Java utiliza hasta el 75 % (unos 2,25 GB de *heap*). Si el contenedor se detiene por falta de memoria (`OOMKilled`), puede reducirse ese porcentaje o aumentarse el límite.

Tamaño máximo de documento. Por defecto es de 50 MB. Los documentos viajan codificados en Base64, que aumenta su tamaño en torno a un 33 %. Por ello, el límite debe mantenerse coherente en estos cuatro puntos:

Ubicación	Parámetro	Valor para 50 MB
`docker-compose.yml` , backend-app	`MAX_UPLOAD_MB` (la web lo recibe de `/api/auth/status`)	`50`
`docker-compose.yml` , servicio `backend-app`	`MAX_BODY_SIZE`	`70mb`
`docker-compose.yml` , servicio `backend-ws`	`JETTY_HTTP_MAXPOSTSIZE`	`73400320`
`proxy/nginx.conf` , rutas `/api/` y `/services/`	`client_max_body_size`	`75m`

El límite del motor se aplica a la petición completa. En una petición por lotes (`submitBatch`), la suma de todos los documentos del lote, codificados en Base64, no puede superarlo.

5.3. Valores por defecto del procesamiento

Los parámetros de procesamiento del motor se definen en `app.properties` , junto con `server.properties` y `security.properties` . Estos valores se aplican cuando una petición no indica una opción: umbrales de detección facial, intensidad del pixelado, zona del logotipo, umbral del reconocimiento de entidades, etc. En la aplicación web, las preferencias de cada usuario y las plantillas prevalecen sobre ellos.

Los parámetros pueden modificarse sin recompilar el motor:

- **Valores concretos:** mediante variables de entorno del servicio `backend-ws` . El nombre de la variable es la clave en mayúsculas, con guiones bajos en lugar de puntos. Por ejemplo, `processing.document.redactNERScore` se convierte en `PROCESSING_DOCUMENT_REDACTNERSCORE` .
- **Un perfil completo:** montando un fichero en `/app/app.properties` . Debe ser una copia completa del original: si existe, sustituye al interno, y las claves que falten toman el valor por defecto del código.

Para que estos ajustes no se vean afectados por las actualizaciones, se recomienda incorporarlos en un fichero `docker-compose.override.yml` . Docker Compose lo combina automáticamente con el principal. Ejemplo, con valores y ruta ilustrativos:

```
services:
  backend-ws:
    environment:
      PROCESSING_DOCUMENT_REDACTNERSCORE: "0.8"
    volumes:
      - ./config/app.properties:/app/app.properties:ro
```

El parámetro `processing.document.logoExclusionZone` define una zona del documento que no se anonimiza, pensada para preservar el logotipo. Conviene ajustarla a las plantillas del organismo.

Los cambios se aplican recreando el servicio: `docker compose up -d backend-ws`.

5.4. Patrones de detección

El fichero `patrones.txt` contiene las expresiones regulares que detectan datos con estructura conocida:

- documentos de identidad (NIF, NIE, CIF);
- tarjetas y cuentas bancarias;
- direcciones y referencias catastrales;
- códigos de verificación e identificadores;
- correos electrónicos y teléfonos;
- matrículas y bastidores de vehículos;
- códigos de barras de pago de 42 dígitos.

Su formato es el siguiente:

- Una entrada por línea, con el formato ETIQUETA::expresión. La etiqueta identifica la clase de dato (por ejemplo, NIF, IBAN o TELF) y la expresión regular sigue la sintaxis de Java.
- Las líneas que empiezan por `#` se ignoran.
- Algunos patrones se incluyen desactivados porque generan falsos positivos: cuentas en formato CCC, huellas hexadecimales, tarjeta genérica, número de afiliación a la Seguridad Social, códigos postales y expedientes
- El patrón «Candidato a nombre» (etiqueta NOMBRE) no debe modificarse, porque lo utiliza el código para la detección por diccionario y por reconocimiento de entidades.

Al añadir o modificar patrones, pruébelos con documentos de prueba. Evite las expresiones con cuantificadores anidados, que pueden degradar el rendimiento. El fichero forma parte de los recursos del motor (apartado 5.6), por lo que los cambios requieren reconstruir su imagen: `docker compose up -d --build backend-ws`.

5.5. Lista blanca global

El fichero `lista_blanca.txt` contiene los términos que no se ocultan en ningún documento. Se aplica además de las listas blancas de cada usuario, plantilla o petición.

- Una entrada por línea. Las líneas que empiezan por `#` se ignoran.
- Las entradas literales se comparan en minúsculas y sin acentos.
- Las entradas con el prefijo `regex:` se interpretan como expresiones regulares.

La lista incluida en el repositorio contiene entradas propias del organismo que desarrolló SensIA: nombres de entidades, direcciones, teléfonos y cargos. Cada organismo debe adaptarla, eliminando las entradas que no le correspondan y añadiendo sus propios nombres, direcciones, teléfonos públicos y cargos. Las entradas demasiado amplias pueden dejar visibles datos que deberían ocultarse, por lo que conviene revisarlas con documentos de prueba. Al igual que los patrones, los cambios requieren reconstruir la imagen del motor.

5.6. Modelos y diccionarios

Recurso	Uso
Diccionario de nombres y apellidos	Detección de nombres por diccionario. Fuente: INE.
Modelo de reconocimiento de entidades `BSC-LT/roberta_model_for_anonimization` (BSC-CNS), en formato ONNX	Servicio `backend-ner`.
Modelo YuNet	Detección facial.
Datos de Tesseract (`spa.traineddata` , `eng.traineddata`)	OCR.

Los modelos y diccionarios se cargan en memoria al arrancar los servicios. El diccionario, el modelo YuNet y los datos de Tesseract forman parte de los recursos del motor (`backend-ws`).

El modelo de reconocimiento de entidades se descarga de Hugging Face y se exporta a ONNX durante la construcción de la imagen de `backend-ner` . Se utiliza la revisión fijada en `backend-ner/scripts/.env.build` (variables `MODEL_NAME` y `MODEL_REVISION`), lo que garantiza que la construcción sea reproducible. En ejecución, el servicio no accede a Internet.

En la versión actual, el diccionario, los patrones y el modelo de entidades están orientados al castellano, y el OCR dispone de los modelos de castellano e inglés. Si el organismo trata documentos en valenciano o catalán, conviene valorar la incorporación del modelo de OCR correspondiente (`cat.traineddata`) y validar los resultados con documentos propios.

5.7. Reconocimiento de entidades

El servicio `backend-ner` identifica personas, organizaciones, lugares y otras entidades. Su uso se solicita en cada petición:

- por categorías, con las opciones `redactNERNames` , `redactNEROrg` , `redactNERLoc` y `redactNERMisc` ;
- con un umbral de confianza (`redactNERScore`). Si la petición no lo indica, se aplica el valor configurado en el motor (processing.document.redactNERScore, 0,65 en la configuración estándar; apartado 5.3).

En el despliegue con Docker Compose, el servicio se ejecuta siempre, y el motor no arranca hasta que `backend-ner` está operativo.

La comunicación entre el motor y `backend-ner` se realiza mediante gRPC con mTLS, activo por defecto (`ENFORCE_SECURITY=true`). Utiliza los certificados `ner_server` y `ws_client` , emitidos por la CA interna y generados por el asistente. El tiempo máximo de respuesta por llamada es de 10 segundos (`GRPC_TIMEOUT_SECONDS`).

Si se solicita el reconocimiento de entidades y el servicio no está disponible, el documento termina en error y no se procesa sin él.

5.8. Correo electrónico

La aplicación utiliza el correo, entre otras funciones, para la recuperación de contraseñas. El asistente de configuración admite cuatro opciones:

- **Servidor SMTP con usuario y contraseña.**
- **OAuth2**, para proveedores que lo requieren, como Gmail o Microsoft 365. Se indican el Client ID, el Client Secret, el Refresh Token y la URL del token. Por ejemplo:
  - Google: `https://oauth2.googleapis.com/token`.
  - Microsoft 365: `https://login.microsoftonline.com/<id-del-tenant>/oauth2/v2.0/token`.
- **Relay del organismo sin autenticación.** El relay debe autorizar el envío desde la dirección IP del servidor de SensIA.
- **Sin correo.** Las funciones que envían mensajes no estarán operativas.

El modo de cifrado se determina por el puerto:

- Con el puerto 465, la conexión usa TLS desde el inicio (`SMTP_SECURE=true`).
- Con el resto de puertos, se utiliza STARTTLS.

El remitente por defecto es `noreply@<dominio>`.

Si no se configura el correo, debe desactivarse la opción **Recuperar Contraseña** en el panel de administración.

5.9. Observabilidad

La observabilidad es opcional y se activa en el asistente de configuración (perfil `obs` de Docker Compose). Se compone de los siguientes elementos:

- Los servicios están instrumentados con OpenTelemetry y envían las trazas a Tempo (puertos 4317 y 4318).
- Alloy recoge los logs de los contenedores y los envía a Loki. Accede a Docker a través de un proxy de solo lectura (`docker-proxy`), sin acceso directo al socket de Docker.
- Grafana permite consultar trazas y logs de forma centralizada.

Grafana solo escucha en el propio servidor. Para acceder desde el equipo del administrador, establezca un túnel SSH:

```
ssh -L 3030:127.0.0.1:3030 usuario@servidor
```

A continuación, abra `http://localhost:3030` con el usuario `admin` y la contraseña `GRAFANA_PASSWORD` del fichero `.env`. Esa contraseña solo se aplica en el primer arranque; después debe cambiarse desde Grafana.

Loki conserva los logs durante 30 días (`observability/loki.yaml`). Como pueden contener datos personales, el plazo de conservación debe fijarlo el responsable del tratamiento, con el asesoramiento del delegado de protección de datos del organismo.

Con la observabilidad activa, se recomienda disponer de 16 GB de memoria (apartado 3.1).

6. Seguridad

6.1. Exposición de red

El proxy inverso es el único servicio que publica puertos hacia el exterior:

Puerto	Uso	Recomendación
80	Redirección de la aplicación web a HTTPS y validación de Let's Encrypt. Las rutas `/api/` y `/services/` se rechazan con HTTP 403.	Mantener accesible si se utiliza Let's Encrypt o si se desea la redirección automática.
443	Aplicación web y, sin mTLS, servicio SOAP con token. Con `REQUIRE_MTLS=true`, `/services/` responde 403 en este puerto.	Accesible para usuarios e integradores.
8443	Servicio SOAP con certificado de cliente y token (mTLS).	Accesible solo desde las redes de los integradores que utilicen mTLS.

El resto de servicios (`backend-app`, `backend-ws`, `backend-ner`, Tempo, Loki y Alloy) no publica puertos. Grafana solo escucha en la interfaz local del servidor (`127.0.0.1`).

Docker gestiona sus propias reglas de filtrado. Por ello, los puertos publicados son accesibles aunque el cortafuegos del sistema (por ejemplo, UFW) los deniegue. El acceso debe limitarse por alguna de estas vías:

- En el cortafuegos perimetral del organismo.
- En el propio proxy, para el puerto 8443, descomentando y ajustando las directivas `allow` y `deny` del fichero `proxy/soap-tls.stream-template`.

Si no se activa el mTLS, el asistente publica el puerto 8443 solo en la interfaz local (`127.0.0.1`), por lo que no es accesible desde el exterior.

Para comprobar los puertos publicados, ejecute:

```
cd /opt/sensia/sensia.git
docker compose ps --format "table {{.Name}}\t{{.Ports}}"
```

En el resultado, solo `proxy-nginx` debe mostrar puertos asociados a todas las interfaces (`0.0.0.0` o `[::]`). El resto de servicios muestra únicamente puertos internos, por ejemplo `3000/tcp`. Si la observabilidad está activa, Grafana aparece asociado a `127.0.0.1:3030`.

6.2. Certificados y autenticación mutua

Certificado del servidor. El certificado del dominio lo utilizan el proxy y el motor. El motor lo incorpora en su almacén de claves (`pki/ws/keystore.p12`), que se regenera en cada renovación. El procedimiento de renovación depende del tipo de certificado:

Tipo	Validez	Renovación

Autofirmado (CA interna)	825 días	Ejecute `./env.sh`, elija la opción 1 y aplique el cambio con `docker compose up -d --force-recreate`.
Del organismo	La que fije la autoridad emisora	Deje el nuevo certificado, con la cadena intermedia, y su clave en `pki/publico`. Después, ejecute `./env.sh`, elija la opción 2 y aplique el cambio con `docker compose up -d --force-recreate`.
Let's Encrypt	Periodo corto, fijado por Let's Encrypt	Manual, con `./certbot.sh renovar` (véase a continuación).

SensIA no renueva ningún certificado de forma automática: su renovación corresponde al organismo. Si el certificado es de Let's Encrypt, ejecute la renovación a mano con el usuario que realizó la instalación, propietario de `/opt/sensia/pki`:

```
cd /opt/sensia/sensia.git
./certbot.sh renovar
```

El script funciona del siguiente modo:

- Solo actúa cuando corresponde renovar.
- Si el proxy está en marcha, valida por *webroot*, sin interrumpir el servicio.
- Tras renovar, regenera el almacén de claves del motor, recarga NGINX sin cortar conexiones y reinicia `backend-ws`.
- Requiere que el puerto 80 sea accesible desde Internet. Si se publicó HTTP en otro puerto, la validación fallará salvo que exista una redirección externa al puerto 80.

Let's Encrypt ya no envía avisos de caducidad por correo, por lo que conviene comprobar periódicamente la fecha de caducidad con la orden que se indica más abajo.

Certificados internos. Los certificados de la comunicación gRPC (`ner_server` y `ws_client`) tienen una validez de 825 días, y `./env.sh` los regenera en cada ejecución. Las dos CA de la instalación tienen una validez de 10 años.

Para revisar las fechas de caducidad de todos los certificados, ejecute:

```
cd /opt/sensia/pki
for c in ca/*.crt publico/*.crt publico/fullchain.pem ner/ner_server.crt ws/ws_client.crt
clientes/*.crt; do
  [ -f "$c" ] && echo "$c: $(openssl x509 -noout -enddate -in "$c")"
done
```

Certificados de cliente (mTLS). El acceso por el puerto 8443 exige dos credenciales:

- Un certificado de cliente emitido por la CA de clientes de la instalación, o por la CA adicional configurada en el asistente, cuya huella SHA-256 figure en `AUTH_CERT_WHITELIST`.
- El token del servicio en la cabecera `AuthHeader` (Anexo B).

El certificado lo valida directamente el motor (Jetty); el proxy no descifra la conexión.

`env.sh` genera el primer certificado de cliente (`cliente_mtls.p12`, con su contraseña en `cliente_mtls.p12.pass`) y registra su huella. Para dar de alta a otros integradores, siga estos pasos:

1. Emita el certificado con la CA de clientes. OpenSSL solicitará la contraseña de exportación del fichero `.p12`.

```
   cd /opt/sensia/pki/clientes
   openssl genrsa -out cliente2.key 2048
   openssl req -new -key cliente2.key -subj "/CN=Organismo 2" -out cliente2.csr
   printf
'basicConstraints=CA:FALSE\nkeyUsage=critical,digitalSignature\nextendedKeyUsage=clientAuth\n' >
cliente2.ext
   openssl x509 -req -in cliente2.csr -CA ../ca/ca_clientes.crt -CAkey ../ca/ca_clientes.key \
     -CAcreateserial -days 365 -sha256 -extfile cliente2.ext -out cliente2.crt
   openssl pkcs12 -export -in cliente2.crt -inkey cliente2.key \
     -certfile ../ca/ca_clientes.crt -name "Organismo 2" -out cliente2.p12
   rm cliente2.csr cliente2.ext
   chmod 600 cliente2.key cliente2.p12
```

2. Calcule su huella:

```
   openssl x509 -noout -fingerprint -sha256 -in cliente2.crt | cut -d= -f2 | tr -d ':' | tr 'A-F'
'a-f'
```

3. Añada la huella a `AUTH_CERT_WHITELIST` en el fichero `.env`, separada de las demás por `;`.
4. Aplique el cambio con `docker compose up -d backend-ws`.

El asistente conserva las huellas añadidas manualmente en sus ejecuciones posteriores.

Además, deben observarse estas pautas:

- **Entrega.** El fichero `.p12` y su contraseña deben entregarse al integrador por canales distintos.
- **Revocación.** No se utilizan listas de revocación (CRL) ni OCSP. Para revocar un certificado, elimine su huella de `AUTH_CERT_WHITELIST` y aplique el cambio con `docker compose up -d backend-ws`.
- **Renovación (365 días).**
  - Para el certificado generado por el asistente, ejecute `./env.sh`, responda «no» a reutilizarlo y entregue el nuevo `.p12`. El asistente sustituye la huella anterior, por lo que el certificado antiguo deja de ser válido al aplicar el cambio. Conviene coordinar el momento con el integrador.
  - Para el resto, emita un certificado nuevo y añada su huella. Retire la anterior cuando el integrador haya completado el cambio.

Comunicación interna gRPC. La comunicación entre el motor y `backend-ner` utiliza mTLS por defecto, con certificados emitidos por la CA interna. No debe desactivarse en producción.

6.3. Secretos

- **Permisos del fichero `.env`.** El asistente crea el fichero `.env` con permisos 640, perteneciente al grupo del directorio (`sensia-admins`). Las copias `.env.bak.*` se crean con permisos 600. Puede comprobarse con:

```
cd /opt/sensia/sensia.git
ls -l .env .env.bak.*
```

Si el grupo o los permisos no son los esperados, corríjalos con `chgrp sensia-admins .env && chmod 640 .env`.

- **Custodia.** No incorpore `.env` ni `.env.bak.*` a ningún repositorio, ni los incluya sin cifrar en las copias de seguridad. Elimine las copias `.env.bak.*` que ya no sean necesarias, porque contienen secretos anteriores.
- **Contraseña inicial del administrador.** `DEFAULT_ADMIN_PASSWORD` solo se utiliza al crear el administrador por primera vez. Cámbiela en el primer acceso; a partir de ese momento, el valor del `.env` deja de ser válido.
- **Renovación del token SOAP.** Si el token puede haber quedado expuesto, genere uno nuevo. Los valores del `.env` van entre comillas simples:

```
cd /opt/sensia/sensia.git
NUEVO=$(openssl rand -hex 24)
sed -i "s/^SOAP_API_TOKEN=.*/SOAP_API_TOKEN='$NUEVO'/" .env
unset NUEVO
docker compose up -d
```

`backend-app` y `backend-ws` se recrean con el nuevo valor. Comunique el nuevo token a los integradores por un canal seguro.

- **Renovación de todos los secretos.** Ejecute `./env.sh` y responda «sí» a «¿Regenerar los secretos?». Se renuevan el token SOAP, las claves de sesión y ALTCHA y la contraseña de Grafana. Aplique después el cambio con `docker compose up -d --force-recreate`. Tenga en cuenta dos efectos:
  - Se cierran todas las sesiones abiertas.
  - La contraseña de Grafana solo se aplica en el primer arranque, por lo que debe cambiarse también desde la propia Grafana.
- **Claves de las CA.** El directorio `/opt/sensia/pki/ca` contiene las claves de las CA, que son el secreto más sensible de la instalación. Su acceso está restringido (permisos 700) y debe incluirse cifrado en las copias de seguridad.

6.4. Protección de la aplicación

En el panel de administración:

- Mantenga activado el **Captcha** si la aplicación es accesible desde Internet.
- Desactive el **Registro** si las cuentas deben crearse solo desde la administración.
- Desactive **Recuperar Contraseña** si no hay correo configurado.
- Asigne el rol de administrador solo a las personas que lo necesiten.

6.5. Datos personales tratados

- **Documentos.** Los documentos subidos, los resultados y los archivos temporales contienen datos personales de terceros. Debe definirse su plazo de conservación, y se recomienda que los usuarios eliminen los documentos de su bandeja una vez descargados. En el servidor no se almacenan documentos de forma permanente, y los archivos temporales se eliminan periódicamente de forma automática.

- **Logs.** Los logs pueden contener fragmentos de texto de los documentos, por ejemplo, las entidades descartadas por el reconocimiento de entidades. En producción, se recomienda:
  - mantener el nivel de log en `INFO`;
  - restringir el acceso a los logs;
  - fijar un periodo de retención coherente con el de los documentos.

  Con la observabilidad activa, Loki conserva los logs durante 30 días (apartado 5.9).
- **Registro de datos sensibles** (`SENSIA_LOG_SENSIBLE`). Está desactivado por defecto y solo debe activarse para diagnósticos puntuales (apartado 7.2).
- **Copias de seguridad.** Contienen datos personales y secretos, por lo que deben almacenarse cifradas y con acceso restringido.
- **Procesamiento local.** Todo el procesamiento (OCR, detección, reconocimiento de entidades y detección facial) se realiza en los contenedores de la instalación, sin enviar los documentos a servicios externos.
- **Recursos de la aplicación web.** La aplicación web sirve desde su propia imagen las fuentes y el componente que muestra los PDF, con sus tablas de caracteres, por lo que el navegador no descarga recursos de terceros. La única excepción son los iconos de las anotaciones de tipo nota: al mostrar un PDF que las contiene, el visor descarga el icono de `unpkg.com`. Esa petición no envía el documento ni su texto, salvo el nombre del icono que figura en la propia nota, normalmente un valor fijo. Sí revela a ese servidor la dirección IP del usuario y el dominio de la instalación.

La implantación debe incorporarse al registro de actividades de tratamiento del organismo (artículo 30 del RGPD) y, en su caso, a la correspondiente evaluación de impacto relativa a la protección de datos (artículo 35 del RGPD).

6.6. Desarrollo seguro y actualizaciones de seguridad

Antes de publicar cada versión, las dependencias del código (npm, Maven y Python) y las imágenes de los contenedores se analizan con Trivy. Las vulnerabilidades altas o críticas con corrección disponible se resuelven antes de la publicación, y el resultado del análisis se indica en `CHANGELOG.md`.

Se recomienda reconstruir periódicamente las imágenes para incorporar las actualizaciones de seguridad de las imágenes base:

```
cd /opt/sensia/sensia.git
docker compose build --pull
docker compose up -d
```

Tras reconstruirlas, el organismo puede analizar las imágenes de su instalación con Trivy, sin instalarlo, desde su imagen de Docker (versión 0.75.0 o posterior):

```
cd /opt/sensia/sensia.git
for img in $(docker compose config --images); do
  docker run --rm -v trivy-cache:/root/.cache/ -v /var/run/docker.sock:/var/run/docker.sock \
  aquasec/trivy:0.75.0 image --scanners vuln --severity HIGH,CRITICAL "$img"
done
```

Las vulnerabilidades deben comunicarse según lo indicado en `SECURITY.md`.

7. Operación y mantenimiento

7.1. Comandos habituales

Los comandos se ejecutan desde el directorio de la instalación (`/opt/sensia/sensia.git`):

Acción	Comando
Estado de los servicios	`docker compose ps`
Logs de todos los servicios	`docker compose logs -f`
Logs de un servicio	`docker compose logs -f backend-ws`
Reiniciar los servicios, sin cambios de configuración	`docker compose restart`
Aplicar cambios del `.env` o de los certificados	`docker compose up -d --force-recreate`
Detener los servicios	`docker compose down`
Arrancar los servicios	`docker compose up -d`
Reconstruir y arrancar	`docker compose up -d --build`

`docker compose restart` no vuelve a leer el fichero `.env` ni los almacenes de claves, por lo que no sirve para aplicar cambios de configuración.

`docker compose down` conserva los volúmenes. No debe utilizarse la opción `-v`, que elimina, entre otros, la base de datos.

7.2. Logs

Docker gestiona los logs de cada contenedor con el controlador `json-file`, con un límite de 10 MB por fichero y 10 ficheros por contenedor. Además:

- `backend-app` escribe ficheros, con rotación, en el volumen `sensia_backend-logs`.
- El motor registra su actividad mediante SLF4J y Logback. Escribe ficheros, con rotación y un máximo de 200 MB, dentro del propio contenedor (`/app/logs`). Estos ficheros se pierden al recrear el contenedor.
- Si la observabilidad está activa, los logs y las trazas pueden consultarse de forma centralizada en Grafana.

El nivel de detalle se ajusta con `LOG_LEVEL`, que por defecto vale `INFO`.

Registro de datos sensibles (`SENSIA_LOG_SENSIBLE`). Está fijado a `OFF` en `docker-compose.yml`. Las pautas para su uso son las siguientes:

- Solo debe activarse para diagnósticos puntuales.
- Debe indicarse un nivel de Logback, por ejemplo `DEBUG`, y nunca `true` o `false`. Un valor no válido como `false` lo activaría en nivel `DEBUG`.
- Los ficheros se guardan como máximo 2 días dentro del contenedor y desaparecen al recrearlo.

- Al terminar el diagnóstico, debe restablecerse el valor `OFF`.

7.3. Modo mantenimiento

- **Desde la aplicación:** al desactivar **Sitio ACTIVO** en el panel de administración, los usuarios no administradores no pueden acceder.
- **Modo estricto:** la aplicación rechaza las peticiones durante las intervenciones. Para activarlo:

  1. Establezca `HARD_MAINTENANCE='true'` en el fichero `.env`.
  2. Aplique el cambio con `docker compose up -d backend-app`.

  Para desactivarlo, restablezca el valor `'false'` y aplique de nuevo el cambio.

7.4. Copias de seguridad

Deben incluirse en las copias de seguridad:

- **La base de datos SQLite** (volumen `sensia_backend-db`).
- `/opt/sensia/pki/ca`: claves de las CA. Sin ellas, hay que reemitir todos los certificados, entregar de nuevo los certificados de cliente y, si se usa el certificado autofirmado, volver a distribuir la CA a los equipos.
- `/opt/sensia/pki/clientes`: certificados entregados a los integradores.
- `/opt/sensia/pki/publico`, si el certificado es del organismo. El resto de la PKI lo regenera `env.sh`. Con Let's Encrypt, puede guardarse también el volumen `certbot-data`, aunque el certificado puede volver a emitirse.
- **El fichero** `.env`, que contiene todos los secretos.
- **Los ficheros de configuración propios del organismo:**

  - `docker-compose.override.yml`;
  - el fichero `app.properties` montado, si existe;
  - `.env.example`, si se ha adaptado;
  - los ficheros versionados que se hayan modificado (`patrones.txt`, `lista_blanca.txt` y `proxy/soap-tls.stream-template`).

  Se recomienda mantener los ficheros versionados modificados en una rama propia del organismo, porque `git pull` puede entrar en conflicto con ellos.

Para obtener una copia coherente de la base de datos, detenga brevemente `backend-app`:

```
cd /opt/sensia/sensia.git
mkdir -p /opt/sensia/backups && chmod 750 /opt/sensia/backups

docker compose stop backend-app
docker run --rm -v sensia_backend-db:/datos:ro -v /opt/sensia/backups:/copia alpine \
  tar czf /copia/backend-db-$(date +%F).tar.gz -C /datos .
docker compose start backend-app
```

La PKI y el fichero `.env` pueden empaquetarse y cifrarse, por ejemplo, con GnuPG. Utilice la herramienta corporativa de cifrado si el organismo dispone de ella:

```
cd /opt/sensia
tar czf - pki sensia.git/.env \
  | gpg --symmetric --cipher-algo AES256 -o backups/sensia-config-$(date +%F).tar.gz.gpg
```

Para restaurar la base de datos, sustituya `AAAA-MM-DD` por la fecha de la copia:

```
cd /opt/sensia/sensia.git
docker compose stop backend-app
docker run --rm -v sensia_backend-db:/datos -v /opt/sensia/backups:/copia:ro alpine \
  sh -c 'rm -rf /datos/* && tar xzf /copia/backend-db-AAAA-MM-DD.tar.gz -C /datos'
docker compose start backend-app
```

Las copias deben cifrarse y almacenarse con acceso restringido (apartado 6.5).

7.5. Actualización

1. Consulte en `CHANGELOG.md` los cambios de la nueva versión, en particular los que afecten al contrato SOAP o a la configuración.
2. Realice una copia de seguridad (apartado 7.4).
3. Descargue la nueva versión:

```
cd /opt/sensia/sensia.git
git fetch --tags
git checkout vX.Y.Z
```

4. Ejecute `./env.sh` y acepte reconfigurar. El asistente parte de los valores y secretos actuales e incorpora al `.env` las variables nuevas.
5. Reconstruya y arranque los servicios:

```
docker compose up -d --build --force-recreate
```

6. En producción, la base de datos no modifica su esquema al arrancar. Si la nueva versión lo cambia, proceda así, después de la copia de seguridad:
    1. Añada temporalmente `DB_SYNC_ALTER: "true"` al entorno de `backend-app`, por ejemplo en `docker-compose.override.yml`.
    2. Ejecute `docker compose up -d`.
    3. Revise los logs del servicio.
    4. Retire la variable y vuelva a aplicar el cambio.
7. Realice las comprobaciones del apartado 4.6.

Para volver a la versión anterior, seleccione su etiqueta con `git checkout` y reconstruya los servicios. Si la versión nueva modificó el esquema, restaure además la copia de la base de datos.

7.6. Actualización de modelos y diccionarios

- **Recursos del motor** (diccionario de nombres, YuNet, datos de Tesseract, patrones y lista blanca). Sustituya los ficheros en los recursos de `backend-ws` y reconstruya el servicio con `docker compose up -d --build backend-ws`.
- **Modelo de reconocimiento de entidades.** Ajuste `MODEL_NAME` y `MODEL_REVISION` en `backend-ner/scripts/.env.build`. Las rutas `MODEL_PATH` y `TOKENIZER_PATH` deben coincidir con las del servicio. Después, reconstruya la imagen:

```
docker compose build --no-cache backend-ner
docker compose up -d backend-ner
```

Antes de utilizar los nuevos modelos o diccionarios en producción, compruebe los resultados con documentos de prueba.

7.7. Resolución de problemas

Síntoma	Causa probable	Actuación
La aplicación no responde tras el arranque	Construcción en curso, servicio reiniciándose o `backend-ner` aún arrancando (el motor espera a que esté operativo)	Revisar `docker compose ps` y los logs del servicio afectado.
La aplicación no responde desde algunas redes del organismo	Las subredes de Docker coinciden con rangos del organismo	Fijar un rango reservado en `/etc/docker/daemon.json` (apartado 4.1).
Error de certificado en el navegador o en los integradores	Certificado autofirmado, CA interna no reconocida o falta de la cadena intermedia	Distribuir `ca_interna.crt` a los equipos y sistemas cliente, o incluir la cadena intermedia en el certificado.
HTTP 403 en `/api/` o `/services/`	Acceso por HTTP	Utilizar siempre HTTPS.
Los integradores reciben HTTP 503	Se ha alcanzado el límite de peticiones simultáneas	Revisar el uso de CPU. Ajustar `JETTY_QOS_MAXREQUESTS` y `JETTY_QOS_SUSPENDMS` (apartado 5.2), o pedir a los integradores que reintenten con espera progresiva.
Documentos grandes rechazados	Límite de tamaño en el proxy, la aplicación o el motor	Revisar los cuatro límites de forma coherente, teniendo en cuenta la codificación Base64 (apartado 5.2).
Los documentos con reconocimiento de entidades terminan en error	`backend-ner` detenido, modelo no cargado o mTLS no coincidente (`ENV` distinto de `production`)	Revisar `docker compose logs backend-ner`, comprobar `ENV='production'` y reiniciar el servicio.
`backend-ws` se reinicia de forma inesperada	Falta de memoria (`OOMKilled`)	Comprobar con `docker inspect --format '{{.State.OOMKilled}}' $(docker compose ps -q backend-ws)` y ajustar la memoria (apartado 5.2).

El OCR no reconoce el texto	Baja resolución del escaneo o idioma no disponible	Escanear a 300 ppp o más y revisar los datos de idioma de Tesseract.
No llegan los correos	Configuración SMTP incorrecta o relay que no autoriza la IP del servidor	Revisar los logs de `backend-app` y la configuración de correo (apartado 5.8).
Se rechaza a un cliente con mTLS	Huella no registrada, CA no incluida en el almacén de confianza, mTLS no activado, certificado caducado o falta del token	Revisar el registro del certificado de cliente (apartado 6.2).
Falla la renovación de Let's Encrypt	Puerto 80 no accesible desde Internet, DNS incorrecto o usuario sin permiso de escritura en `/opt/sensia/pki`	Revisar la salida de `./certbot.sh renovar` y ejecutarlo con el usuario propietario de la PKI.
Los cambios del `.env` no se aplican	Se ha utilizado `docker compose restart`	Aplicar con `docker compose up -d --force-recreate`.

7.8. Consideraciones de operación

- **Trabajos en curso.** El motor mantiene en memoria la cola y los resultados pendientes de recoger. Reiniciar `backend-ws` (por ejemplo, tras renovar el certificado) descarta los trabajos en curso y los resultados no recogidos; los integradores recibirán `NOT_FOUND` y deberán reenviar. Conviene renovar o actualizar en una franja sin actividad.
- **Certificado de una CA privada del organismo.** `backend-app` se conecta al motor por HTTPS con el nombre del dominio y confía en las CA del fichero `pki/app/ca.crt` (`NODE_EXTRA_CA_CERTS`). Si el certificado del dominio lo emite una CA privada del organismo, esa CA debe estar incluida en ese fichero.
- **IPv6.** NGINX escucha también en `[::]`, por lo que el kernel del servidor debe tener IPv6 habilitado, aunque no se utilice.
- **Límite de intentos de acceso.** Las rutas `/api/auth/` admiten 10 peticiones por minuto por dirección IP (ráfaga de 20). Si los usuarios acceden a través de NAT o de un proxy corporativo, todos comparten la misma IP y el límite: en ese caso, ajuste `limit_req` en `proxy/nginx.conf`.
- **Pruebas desde el propio servidor.** Las peticiones lanzadas desde el servidor llegan a NGINX con la IP de la pasarela de la red de Docker (por ejemplo, `172.22.0.1`), no con `127.0.0.1`. Téngalo en cuenta en las reglas `allow` / `deny` y en el límite de intentos.
- **Restricción del servicio SOAP por IP.** Se recomienda limitar `/services/` (puerto 443) y el puerto 8443 a las redes de los integradores, con `allow` / `deny` en `proxy/nginx.conf` y `proxy/soap-tls.stream-template`, o en el cortafuegos perimetral.

8. Limitaciones conocidas

- Algunos códigos QR situados junto a bordes o líneas de tabla pueden no detectarse.
- Los datos repartidos en varias líneas dentro de una celda de tabla pueden no detectarse completos, porque el PDF los almacena como fragmentos separados.
- En documentos escaneados de baja resolución, el OCR puede no reconocer todo el texto y omitir parte de los datos.
- El texto contenido en imágenes de baja resolución incrustadas en el PDF puede no ocultarse.
- Los rostros girados (por ejemplo, en fotografías de documentos de identidad escaneadas de lado) o muy pequeños dentro de imágenes de gran tamaño pueden no detectarse.
- En documentos que combinan páginas con texto y páginas escaneadas, las páginas escaneadas podrían no someterse a OCR, porque la decisión se toma a partir de la primera página.
- El diccionario, los patrones y el modelo de entidades están orientados al castellano.
- El documento final no contiene capa de texto, por lo que no permite búsquedas ni cumple por sí mismo los requisitos de accesibilidad.

Estas limitaciones pueden mitigarse de varias formas:

- La pantalla de revisión permite añadir marcas manuales.
- En documentos basados en plantilla, las zonas fijas, la lista negra o la eliminación de imágenes cubren las zonas recurrentes.

Se agradece la comunicación de nuevos casos a través de las incidencias (*issues*) del repositorio.

Por estas razones, SensIA está concebido para su uso con revisión humana antes de la publicación de los documentos.

Anexo A. Requisitos funcionales y no funcionales

Este anexo describe las capacidades de SensIA 1.0.0-rc.1 en forma de requisitos. Los valores indicados corresponden a la configuración por defecto del despliegue con Docker Compose. Entre paréntesis se indican, cuando procede, los elementos del servicio web que permiten controlar cada capacidad (*Manual del integrador*). Los identificadores permiten relacionar los casos de prueba con los requisitos que verifican.

A.1. Requisitos funcionales

A.1.1. Entrada de documentos

ID	Requisito
RF-ENT-01	El sistema admite documentos PDF, nativos o escaneados, e imágenes JPG y PNG.
RF-ENT-02	Los documentos pueden enviarse de forma individual desde la aplicación web o por lotes mediante el servicio web (operación `submitBatch`).
RF-ENT-03	El servicio web recibe el contenido de los ficheros codificado en Base64 (`fileContent`), junto con su nombre (`fileName`) y su tipo MIME (`contentType`).
RF-ENT-04	Los documentos PDF dañados, cifrados o protegidos con contraseña se rechazan con el estado `INVALID_DOCUMENT_ERROR`.

A.1.2. Reconocimiento óptico de caracteres (OCR)

ID	Requisito
RF-OCR-01	El sistema determina automáticamente si un documento requiere OCR a partir del texto extraíble de la primera página. Si contiene menos de 10 caracteres, se aplica OCR a todo el documento.
RF-OCR-02	Antes de aplicar el OCR, analiza la resolución de las imágenes del documento para ajustar el proceso.
RF-OCR-03	El OCR se realiza con Tesseract y genera una versión temporal del documento con capa de texto, sobre la que se aplica la detección.
RF-OCR-04	El OCR dispone de los modelos de idioma español e inglés.

A.1.3. Extracción de texto

ID	Requisito
RF-EXT-01	El texto extraído se normaliza, descartando caracteres anómalos, antes de la detección.

RF-EXT-02	El texto se agrupa en bloques que pueden abarcar varias líneas, para detectar datos divididos por un salto de línea. La detección en celdas de tabla con saltos de línea tiene limitaciones (apartado 8).
RF-EXT-03	Se detectan las fuentes no incrustadas o no estándar y se ajustan las tolerancias de espaciado para evitar que las palabras se fragmenten.
RF-EXT-04	La posición y el tamaño de cada detección se calculan a partir de las posiciones de los caracteres, teniendo en cuenta la rotación del texto (0, 90 y 270 grados) y las métricas de la fuente.
RF-EXT-05	Si las métricas de una fuente no son válidas, como ocurre en algunas capas de OCR, se aplican alturas por defecto para que la caja cubra la palabra.
RF-EXT-06	Se corrigen los desajustes entre la posición de un dato en el texto extraído y su posición gráfica en la página.
RF-EXT-07	Se descartan los fragmentos sin contenido útil: cadenas de más de 500 caracteres sin espacios, secuencias repetitivas de signos y texto no visible.

A.1.4. Detección de datos

ID	Requisito
RF-DET-01	El sistema detecta datos con estructura conocida mediante las expresiones regulares de `patrones.txt`: documentos de identidad, tarjetas y cuentas bancarias, direcciones y referencias catastrales, códigos de verificación e identificadores, correos electrónicos, teléfonos, matrículas, bastidores y códigos de barras de pago. El número de afiliación a la Seguridad Social se incluye desactivado.
RF-DET-02	Los términos de la lista negra (`blockedWords`) se ocultan en todo el documento, buscando palabras completas y sin distinguir mayúsculas y minúsculas.
RF-DET-03	Opcionalmente (`redactNames`), los nombres y apellidos se detectan por diccionario. Se identifican grupos de 2 a 4 palabras con mayúscula inicial, admitiendo partículas como «de», «del» o «la». El grupo se marca si al menos dos de sus palabras, y como mínimo la mitad, figuran en el diccionario.
RF-DET-04	Opcionalmente, se detectan personas, organizaciones, lugares y otras entidades mediante un modelo de reconocimiento de entidades. Las categorías se seleccionan con `redactNERNames` , `redactNEROrg` , `redactNERLoc` y `redactNERMisc` , y el umbral de confianza es configurable (`redactNERScore`).
RF-DET-05	Se excluyen de la ocultación los términos de la lista blanca global y de las listas de cada usuario, plantilla o petición (`allowedWords`). Las entradas literales se comparan sin distinguir mayúsculas ni acentos, y se admiten expresiones regulares con el prefijo `regex:` . Estas detecciones se marcan como `WHITELISTED` .
RF-DET-06	Se resuelven las detecciones solapadas. Si una caja está contenida en otra, se conserva la mayor. Si coinciden, prevalece la exclusión de la lista blanca.

A.1.5. Códigos QR y de barras

ID	Requisito

RF-QR-01	Cuando se solicita (`redactQRCodes`), el sistema detecta y oculta códigos QR, PDF417, Data Matrix, Code 128, Code 39, EAN-13 e ITF.
RF-QR-02	La resolución de análisis se ajusta a las dimensiones de la página, entre 150 y 300 ppp.
RF-QR-03	El número de códigos buscados se limita en la primera página, en la última y en las intermedias. Por defecto, los límites son 4, 3 y 2, y se ajustan con `maxQrFirstPage` , `maxQrLastPage` y `maxQrPage` .
RF-QR-04	Se analiza primero la página completa. Si no se alcanza el límite, se divide en franjas horizontales y verticales con solapamiento. En la primera página se analiza además la zona inferior derecha.
RF-QR-05	Se omite el análisis de las zonas sin contenido gráfico suficiente.
RF-QR-06	Si se detecta un código en el margen de la segunda página, en las páginas siguientes solo se comprueba esa misma posición (opción `cloneQRMargin`).
RF-QR-07	Se eliminan los códigos duplicados cuyos centros distan menos de 50 píxeles.
RF-QR-08	La caja de cada código se amplía a partir de los píxeles adyacentes para cubrirlo por completo, con criterios distintos para códigos lineales y bidimensionales.

A.1.6. Imágenes y rostros

ID	Requisito
RF-IMG-01	El sistema elimina las imágenes incrustadas del documento cuando se solicita (`removeImages`).
RF-IMG-02	No se eliminan las imágenes situadas en la zona del logotipo (`preserveLogo` , `logoExclusionZone`) ni en las zonas de exclusión de imágenes (`imageExclusionZones`).
RF-IMG-03	No se eliminan las imágenes que ocupan más del 85 % del ancho y del alto de la página, como fondos o páginas escaneadas.
RF-IMG-04	Salvo que se desactive (`maskFaces`), se detectan rostros en las imágenes incrustadas con el modelo YuNet. La imagen se sustituye por una versión con los rostros ocultos, con un margen del 20 %, mediante un mosaico proporcional al tamaño del rostro o, con intensidad 100, un relleno sólido (`pixelationIntensity`). Los parámetros del detector son ajustables (`YunetOptions`). Si una imagen no puede analizarse, el documento termina con el estado `ERROR` .
RF-IMG-05	Se pueden ocultar rostros en imágenes individuales de forma síncrona (operación `redactImage`).
RF-IMG-06	Se pueden extraer las imágenes de un PDF, con filtros de tamaño mínimo y máximo (operación `extractImages`).

A.1.7. Ocultación y seudonimización

ID	Requisito
RF-OCU-01	Las detecciones se ocultan con cajas opacas negras. Las zonas fijas se cubren con cajas blancas.
RF-OCU-02	Cada caja se amplía con un margen calculado a partir de la altura del texto y del parámetro de grosor del tachado (`redactionThickness` , por defecto 2), para cubrir completamente los caracteres.
RF-OCU-03	Las cajas de los códigos QR se amplían con un margen fijo adicional.
RF-OCU-04	Las zonas fijas definidas por coordenadas (`fixedRedactionZones`) se cubren en todas las páginas.
RF-OCU-05	Opcionalmente (`usePseudonyms`), cada dato se sustituye por un seudónimo. El dato se cubre con una caja blanca y se escribe encima el seudónimo, con un tamaño de letra proporcional a la altura de la caja.
RF-OCU-06	Un mismo dato recibe el mismo seudónimo en todo el documento. El seudónimo asignado se devuelve en el detalle de cada detección (`pseudonym`).

A.1.8. Vista previa y revisión

ID	Requisito
RF-REV-01	En modo vista previa (`redactionPreview`), se dibujan contornos sin relleno: rojos para los elementos que se ocultarían y verdes para las exclusiones. Las imágenes que se eliminarían se marcan con un aspa roja.
RF-REV-02	Tras la revisión, las marcas confirmadas (`confirmedRedactions`) se aplican directamente en modo de reconstrucción (`reconstructionMode`), sin repetir la extracción, el OCR ni la detección.
RF-REV-03	En la aplicación directa, las imágenes y rostros confirmados para su ocultación se cubren con cajas blancas.
RF-REV-04	Se verifica, mediante una huella del documento original (`sourceDocumentHash`), que las marcas confirmadas corresponden al mismo documento. Si no coinciden, el proceso finaliza con `HASH_MISMATCH_ERROR` .
RF-REV-05	Cada marca confirmada indica si debe aplicarse (`apply`).

A.1.9. Sanitización

ID	Requisito
RF-SAN-01	Tras aplicar la ocultación, cada página del documento se convierte en imagen en memoria, antes de guardar el resultado. El documento final no conserva capa de texto, capas ocultas ni metadatos del original.

A.1.10. Aplicación web

ID	Requisito
RF-WEB-01	Bandeja de trabajo para subir documentos, consultar su estado y acceder a las acciones disponibles.
RF-WEB-02	Pantalla de revisión para aceptar o descartar cada detección antes de generar el documento final.
RF-WEB-03	Comparador sincronizado del documento original y el resultado.
RF-WEB-04	Reprocesamiento de documentos con ajustes específicos y plantillas reutilizables.
RF-WEB-05	Preferencias de procesamiento por usuario y plantilla por defecto.
RF-WEB-06	Panel de administración con gestión de usuarios y roles, y con controles globales: sitio activo, registro, recuperación de contraseña y captcha.

A.1.11. Servicio web

ID	Requisito
RF-WS-01	Servicio SOAP 1.1, con estilo *document/literal* y espacio de nombres `http://sensia.github.io/`. Ofrece las operaciones `submitBatch`, `getStatus`, `redactImage` y `extractImages`.
RF-WS-02	Procesamiento asíncrono de lotes. Cada documento lleva un identificador asignado por el sistema cliente (`clientId`) y recibe un identificador de seguimiento (`serverTicketId`), que se consulta con `getStatus`.
RF-WS-03	Estados tipificados del procesamiento: `QUEUED`, `PROCESSING`, `COMPLETED`, `NOT_FOUND`, `AUTH_ERROR`, `INVALID_DOCUMENT_ERROR`, `HASH_MISMATCH_ERROR` y `ERROR`.
RF-WS-04	Resultado de cada operación (`SUCCESS`, `WARNING` o `ERROR`), con mensaje y causa.
RF-WS-05	Detalle de cada detección: tipo, subtipo, acción aplicada, descripción, texto, página, coordenadas, huella del documento, indicador de aplicación y seudónimo.
RF-WS-06	Errores de petición devueltos como SOAP Fault (`ServiceFault`), con código y mensaje.
RF-WS-07	Autenticación de todas las operaciones mediante un token en la cabecera SOAP `AuthHeader`.
RF-WS-08	Publicación del contrato WSDL por el propio servicio, en la dirección del servicio con el sufijo `?wsdl`.

A.2. Requisitos no funcionales

A.2.1. Arquitectura y despliegue

ID	Requisito
RNF-ARQ-01	Servicios independientes: aplicación web (Angular), backend de aplicación (Node.js), motor de procesamiento (Java) y reconocimiento de entidades (Python).
RNF-ARQ-02	Despliegue en contenedores Docker orquestados con Docker Compose. Los contenedores se conectan por una red interna dedicada, y un proxy inverso actúa como único punto de entrada.
RNF-ARQ-03	Persistencia de la base de datos y de los logs en volúmenes de Docker.
RNF-ARQ-04	Versiones base: Node.js 24, Java 21, Angular 21, Python 3.12, OpenCV 4.13.0, Jetty 12 y NGINX 1.30.
RNF-ARQ-05	Hardware de referencia: servidor `amd64` con 4 vCPU y 8 GB de RAM, con Linux (probado en Ubuntu 26.04). Se recomiendan 16 GB con la observabilidad activa.
RNF-ARQ-06	Instalación en un directorio dedicado (`/opt/sensia`), administrado por un grupo específico del sistema.

A.2.2. Seguridad

ID	Requisito
RNF-SEG-01	Comunicaciones externas cifradas con TLS 1.2 o 1.3 a través de un proxy inverso (puerto 443). El HTTP sin cifrar solo redirige la aplicación web; la API y el servicio SOAP lo rechazan.
RNF-SEG-02	Autenticación mutua (mTLS) opcional para los clientes del servicio web, en un puerto propio (8443). El certificado lo valida el motor de extremo a extremo frente a una lista de huellas autorizadas, y la petición requiere además el token.
RNF-SEG-03	Cifrado y autenticación mutua, activos por defecto, en la comunicación interna gRPC con el servicio de reconocimiento de entidades.
RNF-SEG-04	Autenticación de las peticiones SOAP mediante un token en la cabecera `AuthHeader` .
RNF-SEG-05	Sesiones de la aplicación web basadas en tokens firmados (JWT).
RNF-SEG-06	Protección de los formularios de acceso y registro frente a accesos automatizados mediante ALTCHA.
RNF-SEG-07	Generación aleatoria de contraseñas, tokens y claves durante la instalación.

RNF-SEG-08	Infraestructura de clave pública propia, con una CA interna y una CA de clientes. Admite el certificado autofirmado con la CA interna, el certificado del organismo o un certificado de Let's Encrypt, con almacenes de claves PKCS#12 para el motor.
RNF-SEG-09	Análisis de vulnerabilidades de las dependencias y de las imágenes de los contenedores con Trivy antes de cada versión. Las vulnerabilidades altas o críticas con corrección disponible se resuelven antes de publicarla.
RNF-SEG-10	Envío de correo mediante SMTP con autenticación básica, OAuth2 o relay del organismo.
RNF-SEG-11	Contenedores sin privilegios de root (salvo el proxy), con la opción `no-new-privileges`, sin capacidades del kernel innecesarias y con límites de memoria.
RNF-SEG-12	Secretos en un fichero `.env` con permisos restringidos. Las claves de las CA no se montan en ningún contenedor, y cada servicio accede solo a sus propios certificados.
RNF-SEG-13	Herramientas de observabilidad no publicadas al exterior. El recolector de logs accede a Docker a través de un proxy de solo lectura.

A.2.3. Rendimiento

ID	Requisito
RNF-REN-01	Procesamiento asíncrono de lotes, sin mantener abierta la conexión del cliente.
RNF-REN-02	Servidor del motor con entre 10 y 200 hilos y un máximo de 8 peticiones SOAP procesadas simultáneamente. Las peticiones que esperan más de 30 segundos en cola se rechazan con HTTP 503.
RNF-REN-03	Aplicación directa de las marcas confirmadas, sin repetir la detección (RF-REV-02).
RNF-REN-04	Finalización anticipada de la búsqueda de códigos al alcanzar el límite por página.
RNF-REN-05	Tamaño máximo configurable. Por defecto, 50 MB por documento y 70 MB por petición, incluida la codificación Base64.

A.2.4. Disponibilidad

ID	Requisito
RNF-DIS-01	Reinicio automático de los servicios ante fallos (política `unless-stopped`).
RNF-DIS-02	Cierre de las conexiones inactivas a los 30 segundos y liberación de los hilos inactivos a los 60 segundos.
RNF-DIS-03	Modo mantenimiento desde el panel de administración y modo estricto mediante configuración.

RNF-DIS-04	Tolerancia a documentos con métricas de fuente dañadas (RF-EXT-05).
RNF-DIS-05	Comprobación periódica del estado de cada servicio (*healthcheck*) y arranque ordenado: el motor espera a que el servicio de reconocimiento de entidades esté operativo.

A.2.5. Observabilidad y trazabilidad

ID	Requisito
RNF-OBS-01	Instrumentación de los servicios con OpenTelemetry y envío de trazas a Tempo (opcional).
RNF-OBS-02	Centralización de logs en Loki, mediante Alloy, y consulta en Grafana (opcional).
RNF-OBS-03	Registro de la actividad del motor con SLF4J y Logback.
RNF-OBS-04	Rotación de los logs de los contenedores: 10 MB por fichero y un máximo de 10 ficheros por contenedor.
RNF-OBS-05	Detalle de las detecciones de cada documento en la respuesta del servicio, utilizable como registro de auditoría.

A.2.6. Interoperabilidad

ID	Requisito
RNF-INT-01	Servicio web SOAP 1.1 (*document/literal*) con contrato WSDL publicado por el propio servicio e implementado con Apache CXF (JAX-WS). Se resume en el Anexo B y se describe en detalle en el *Manual del integrador*.
RNF-INT-02	Comunicación interna entre el motor y el reconocimiento de entidades mediante gRPC y Protocol Buffers.
RNF-INT-03	Comunicación entre la aplicación web y su backend mediante REST/JSON.

A.2.7. Protección de datos

ID	Requisito
RNF-DAT-01	Revisión humana de las detecciones antes de generar el documento final, en el modo con vista previa.
RNF-DAT-02	Sanitización del documento final (RF-SAN-01).

RNF-DAT-03	Procesamiento local de los documentos, sin envío a servicios externos.
RNF-DAT-04	Eliminación de los archivos temporales al finalizar su periodo de retención y en la parada controlada del motor.
RNF-DAT-05	Medidas de seguridad diseñadas teniendo en cuenta el Esquema Nacional de Seguridad. La conformidad con el ENS depende de cada implantación y debe verificarla la entidad correspondiente.
RNF-DAT-06	Registro de datos sensibles en los logs desactivado por defecto, con activación solo para diagnósticos puntuales y conservación máxima de 2 días.

A.2.8. Marco legal

ID	Requisito
RNF-LEG-01	Código propio distribuido bajo licencia EUPL v1.2.
RNF-LEG-02	Componentes de terceros con sus propias licencias, relacionados en `NOTICE.md`.

Anexo B. Especificación del servicio web SOAP

Este anexo resume los aspectos del servicio web SOAP necesarios para su administración y para atender las consultas de los sistemas integradores.

El contrato completo se describe en el *Manual del integrador*, que es el documento de referencia para el desarrollo de integraciones. Incluye las opciones de procesamiento, los tipos de datos, las enumeraciones, los flujos de uso, la gestión de errores y los ejemplos de clientes.

B.1. Datos generales

Aspecto	Valor
Protocolo	SOAP 1.1, estilo *document/literal*
Espacio de nombres	`http://sensia.github.io/`
Servicio y puerto WSDL	`SensiaService` / `SensiaServicePort` (enlace `SensiaServiceSOAPBinding`)
Dirección, vía con token	`https://<dominio>/services/SensiaService`
Dirección, vía con certificado de cliente	`https://<dominio>:8443/services/SensiaService`
Contrato WSDL	Dirección del servicio con el sufijo `?wsdl`
Cabecera HTTP `Content-Type`	`text/xml; charset=utf-8`
Tamaño máximo de la petición	70 MB, incluida la codificación Base64 (apartado 5.2)

El servicio solo se atiende por HTTPS; las peticiones por HTTP se rechazan con el código 403. El WSDL publicado por el servicio anuncia la dirección HTTPS de la instalación.

El fichero `service.wsdl` del código fuente contiene una dirección de desarrollo (`http://localhost:8080/SensiaService`) que no debe utilizarse.

B.2. Autenticación

Todas las operaciones requieren la cabecera SOAP `AuthHeader`, con el token de la instalación (`SOAP_API_TOKEN`). El elemento `token` se escribe sin prefijo de espacio de nombres:

```
<soapenv:Header>
  <sen:AuthHeader xmlns:sen="http://sensia.github.io/">
    <token>VALOR_DE_SOAP_API_TOKEN</token>
    <cliente>IDENTIFICADOR_DEL_USUARIO_O_APLICACION</cliente>
  </sen:AuthHeader>
</soapenv:Header>
```

Las peticiones sin token o con un token no válido se rechazan con un SOAP Fault con el código `AUTH_ERROR`.

El token es común a todos los sistemas integradores de la instalación, por lo que su renovación les afecta a todos (apartado 6.3). En la vía con certificado de cliente se exigen tanto el certificado autorizado como el token (apartado 6.2).

cliente es obligatorio (1 a 128 caracteres). Cada cliente puede tener como máximo 2 documentos en proceso, y un lote (submitBatch) admite como máximo 2 documentos. Si se supera, la respuesta lleva cause = SATURADO y se puede reintentar más tarde.

B.3. Operaciones

Operación	Modo	Descripción	`SOAPAction`
`submitBatch`	Asíncrona	Envía un lote de documentos PDF y devuelve un identificador de seguimiento por documento.	`http://sensia.github.io/SensiaService/submitBatch`
`getStatus`	Consulta	Devuelve el estado de un documento y, al completarse, el resultado y el detalle de las detecciones.	`http://sensia.github.io/SensiaService/getStatus`
`redactImage`	Síncrona	Oculta los rostros de una imagen JPG o PNG.	`http://sensia.github.io/SensiaService/redactImage`
`extractImages`	Síncrona	Extrae las imágenes incrustadas en un PDF, sin anonimizarlas.	`http://sensia.github.io/SensiaService/extractImages`

B.4. Aspectos relevantes para la administración

Aspecto	Comportamiento
Peticiones simultáneas	Máximo de 8, común a todos los integradores y a la aplicación web. Las peticiones que esperan más de 30 segundos en cola reciben HTTP 503 (apartado 5.2).

Documentos en cola	Máximo de 12, común a todos los integradores y a la aplicación web. Un lote que no cabe en las plazas libres se rechaza completo con `ERROR` («Servicio saturado»).
Tamaño de las peticiones	70 MB por petición, equivalentes a unos 50 MB de contenido original (apartado 5.2).
Conservación de resultados	Cada resultado solo puede recuperarse una vez: tras una consulta con estado `COMPLETED`, se elimina del servidor. Los resultados no recogidos se eliminan al finalizar el periodo de retención.
Opciones no indicadas	Se aplican los valores configurados en el motor (apartado 5.3).
Reconocimiento de entidades	Si se solicita y el servicio `backend-ner` no está disponible, el documento termina con el estado `ERROR` (apartado 5.7).

B.5. Comprobación del servicio

La siguiente consulta permite comprobar que el servicio está operativo y que el token es válido, sin enviar ningún documento. Utiliza un identificador de seguimiento inexistente:

```
DOMINIO=sensia.tuorganismo.es
TOKEN=$(grep '^SOAP_API_TOKEN=' /opt/sensia/sensia.git/.env | cut -d"'" -f2)

curl -s -H 'Content-Type: text/xml; charset=utf-8' \
  -H 'SOAPAction: "http://sensia.github.io/SensiaService/getStatus"' \
  --data-binary @- "https://$DOMINIO/services/SensiaService" <<EOF | grep -oE
'<(processState|errorCode)>[^<]*'
<soapenv:Envelope xmlns:soapenv="http://schemas.xmlsoap.org/soap/envelope/"
xmlns:sen="http://sensia.github.io/">
  <soapenv:Header><sen:AuthHeader><token>$TOKEN</token><cliente>comprobacion</cliente>
</sen:AuthHeader></soapenv:Header>
  <soapenv:Body><sen:getStatusRequest><ticketId>comprobacion</ticketId></sen:getStatusRequest>
</soapenv:Body>
</soapenv:Envelope>
EOF
unset TOKEN
```

Con certificado autofirmado, añada `--cacert /opt/sensia/pki/ca/ca_interna.crt`. Para comprobar la vía con certificado de cliente, utilice el puerto 8443 y añada `--cert` y `--key` (apartado 4.6).

Resultado	Interpretación
`<processState>NOT_FOUND`	El servicio está operativo y el token es válido.
`<errorCode>AUTH_ERROR`	El token no es válido. Revise `SOAP_API_TOKEN` en el `.env`.
Sin salida o error de conexión	Revise `docker compose ps`, los logs de `backend-ws` y la exposición de puertos (apartado 6.1).

B.6. Información para los sistemas integradores

Para el desarrollo de una integración, facilite a cada sistema integrador:

- el *Manual del integrador*;
- la dirección del servicio y los límites de la instalación;
- el token, por un canal seguro;
- en su caso, el certificado de cliente y su contraseña, por canales distintos.