

Universidad Mariano Galvez de Guatemala

Ingeniería en sistemas de Información

Arquitectura de computadoras II

Ing. Alex Garcia

Proyecto Final

Julio Romeo Fuentes Velasquez 0900-08-558

Marlon André López Amado 0900-12-12964

José Alejandro Rivas Palencia 0900-20-776

Miguel Roberto Pineda Ruano 0900-15-15258

Guatemala 16 de Octubre 2026

Índice

Indice.....	2
1. Resumen ejecutivo.....	3
2. Contexto y problema.....	4
3. Objetivos.....	4
3.1 Objetivo general.....	4
3.2 Objetivos específicos.....	5
4. Inventario tecnológico y roles.....	5
5. Arquitectura global implementada.....	7
5.1 Plano de control y plano de datos.....	8
6. Arquitectura de red y direccionamiento.....	8
6.1 Topología validada.....	8
6.2 Puertos y protocolos.....	9
6.3 Incidente de clonación: duplicación de MAC.....	10
6.4 Incidente de comunicación Master → Master dentro de Docker.....	10
7. Flujo de sincronización ODS: MySQL → n8n → SQLite.....	11
7.1 Configuración del flujo de n8n.....	11
7.2 Directorios operativos.....	12
7.3 Campos de ODS replicados.....	12
8. Biblioteca documental independiente de ODS.....	13
9. Flujo del agente de IA.....	14
9.1 Herramientas.....	14
9.2 Evolución de la API.....	15
9.3 Modelo y rendimiento.....	15
10. Interfaz web y seguridad de interacción.....	16
11. Balanceo, disponibilidad y tolerancia a fallos.....	17
11.1 Balanceo implementado.....	17
11.2 Caída controlada y recuperación.....	18
11.3 Punto único de fallo (SPOF).....	19
12. Disponibilidad sin internet: límites exactos.....	19
13. Pruebas y evidencias disponibles.....	20
13.1 Comandos de evidencia reproducibles.....	21
14. Seguridad, integridad y buenas prácticas.....	22
15. Propuesta de topología final: tres mini PC.....	23
16. Métricas para informe final.....	24
17. Problemas encontrados y solución.....	25
18. Riesgos y limitaciones.....	26
19. Conclusiones.....	26

1. Resumen ejecutivo

EdgeAssist AI es un asistente de inteligencia artificial de ejecución local, diseñado para consultar órdenes de servicio (ODS), manuales técnicos y políticas o procedimientos internos en lenguaje natural, sin requerir servicios de inferencia en la nube. Se implementó utilizando un modelo de lenguaje Qwen cuantizado, ejecutado con llama.cpp, una API en FastAPI, bases SQLite de consulta local, un flujo de actualización desde MySQL mediante n8n y una interfaz web personalizada visualmente para SERVIMEX, pero independiente de la aplicación empresarial original.

Entre las funcionalidades implementadas se encuentran las siguientes:

- Consultar ODS existentes, su estado, falla y otros campos autorizados.
- Consultar manuales técnicos por modelo o tema, sin vincular necesariamente la búsqueda a una ODS.
- Consultar documentos internos (procedimientos, políticas) sin requerir un modelo de producto.
- Elegir entre herramientas consultar_ods y buscar_documentacion mediante selección automática del modelo (tool_choice=auto).
- Mostrar evidencia recuperada desde bases locales para verificar las respuestas.
- Distribuir las peticiones de inferencia entre dos nodos mediante HAProxy.
- Mantener las consultas durante una interrupción controlada de uno de los motores de inferencia, según la validación reportada por el operador.

El alcance de esta implementación comprende el balanceo de carga y la tolerancia a fallos en la capa de inferencia. Esto no equivale a disponer de alta disponibilidad integral de la aplicación. El Master, la API, HAProxy y la biblioteca local en el Master siguen constituyendo puntos únicos de fallo en la topología actual.

2. Contexto y problema

El entorno de servicio técnico utiliza una base MySQL de producción que incluye la tabla `ordenes_servicio`. Las búsquedas tradicionales requieren conocer el sistema o los identificadores. Asimismo, los manuales y documentos operativos se consultan por separado. Se plantea un asistente local que reduzca la fricción de búsqueda y no dependa de servicios externos de IA para responder.

El problema que aborda el proyecto consiste en facilitar consultas de datos operativos y documentación en lenguaje natural, utilizando recursos informáticos limitados y manteniendo la continuidad de la inferencia cuando uno de sus nodos deja de estar disponible.

El diseño se desarrolló considerando recursos de hardware modestos (equipos Celeron de aproximadamente 8 GB previstos para el despliegue final), inferencia local y cuantizada, almacenamiento independiente de MySQL de producción, comunicación LAN, y solución reproducible con software libre o de uso local.

3. Objetivos

3.1 Objetivo general

Diseñar e implementar un asistente Edge AI distribuido, capaz de consultar datos y documentación local en lenguaje natural, mediante inferencia cuantizada, sincronización de información, balanceo de carga y tolerancia a fallos parciales.

3.2 Objetivos específicos

1. Ejecutar un LLM compacto en hardware de bajos recursos con llama.cpp y cuantización de 4 bits.
2. Mantener una réplica SQLite local de campos seleccionados de `ordenes_servicio` sin escribir sobre MySQL de producción.
3. Construir una biblioteca de manuales, políticas y procedimientos mediante SQLite FTS5.

4. Exponer un agente FastAPI que seleccione herramientas con Qwen y acompañe las respuestas con evidencia.
5. Ofrecer una interfaz de chat web independiente, diseñada con la identidad gráfica de SERVIMEX.
6. Distribuir inferencias entre dos Workers y verificar la continuidad tras la caída de uno.
7. Documentar límites, métricas y migración a tres mini PC.

4. Inventario tecnológico y roles

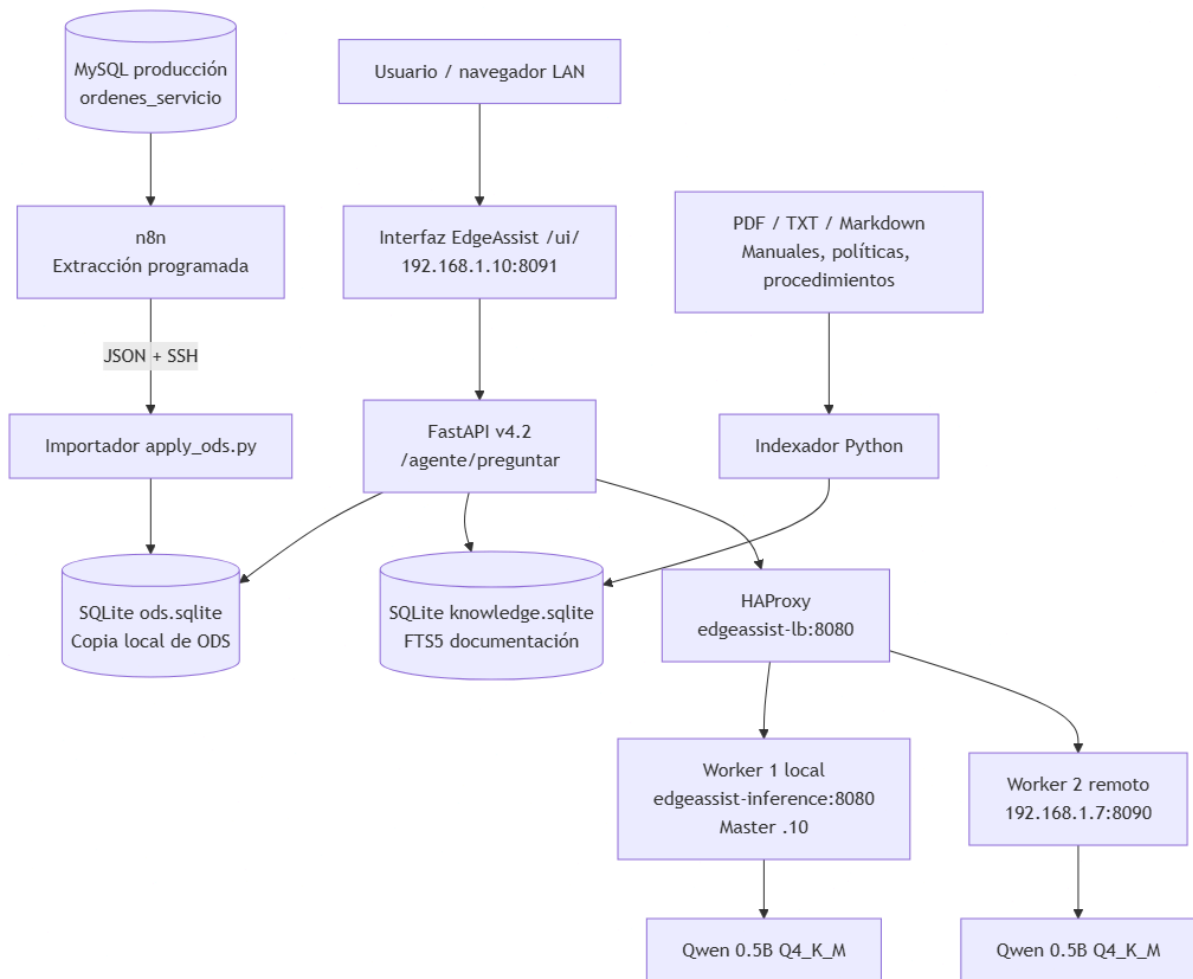
Elemento	Tecnología	Función	Situación
Fuente ODS	MySQL, tabla ordenes_servicio	Origen autoritativo	Existente
Orquestación ETL	n8n	Extracción, preparación y transporte de datos	Configurado; sincronización probada
Transporte ETL	SSH	Copia del lote JSON y ejecución de importador	Incluido en el flujo n8n
Réplica ODS	SQLite	Consultas locales de órdenes	152 filas observadas en prueba
Biblioteca	SQLite + FTS5	Índice de búsqueda de documentos	506 fragmentos observados
Importador documental	Python + extracción de texto de PDF	Indexación de PDF, TXT y Markdown	Utilizado para manual y política
IA	Qwen2.5-0.5B-Instruct, Q4_K_M	Comprensión y redacción	Funcional en laboratorio
Inferencia	llama.cpp server	API compatible con chat/completions	Funcional
Aplicación	FastAPI v4.2	Herramientas, validación, API y	Funcional

		evidencia	
UI	HTML/CSS/JavaScript estáticos	Chat independiente, apariencia SERVIMEX	Funcional según usuario
Proxy de inferencia	HAProxy en Docker	Round-robin y health checks	Ambos Workers UP y prueba funcional
Virtualización	Proxmox VE + LXC Ubuntu 22.04	Aislamiento en servidores físicos	Dos instancias restauradas y operativas
Contenedores	Docker Engine + Compose	Despliegue y redes internas	Funcional

Durante la fase inicial de laboratorio se utilizó un Dell OptiPlex con Intel i5-3470 y 12 GB de RAM, ejecutando Proxmox; el LXC de pruebas tenía límite aproximado de 6 GB de memoria y 2 vCPU. Más adelante se utilizaron dos servidores físicos con LXC replicados, IP 192.168.1.10 y 192.168.1.7. No se confirmó que esos dos hosts ya sean las mini PC Celeron definitivas, ni se registraron sus CPU/RAM reales.

La arquitectura prevista para la entrega contempla Master + 2 Workers en tres mini PC, mientras que la implementación validada utiliza Master + Worker 1 en .10 y Worker 2 en .7. Por tanto, la separación física del tercer equipo todavía no constituye un resultado validado.

5. Arquitectura global implementada



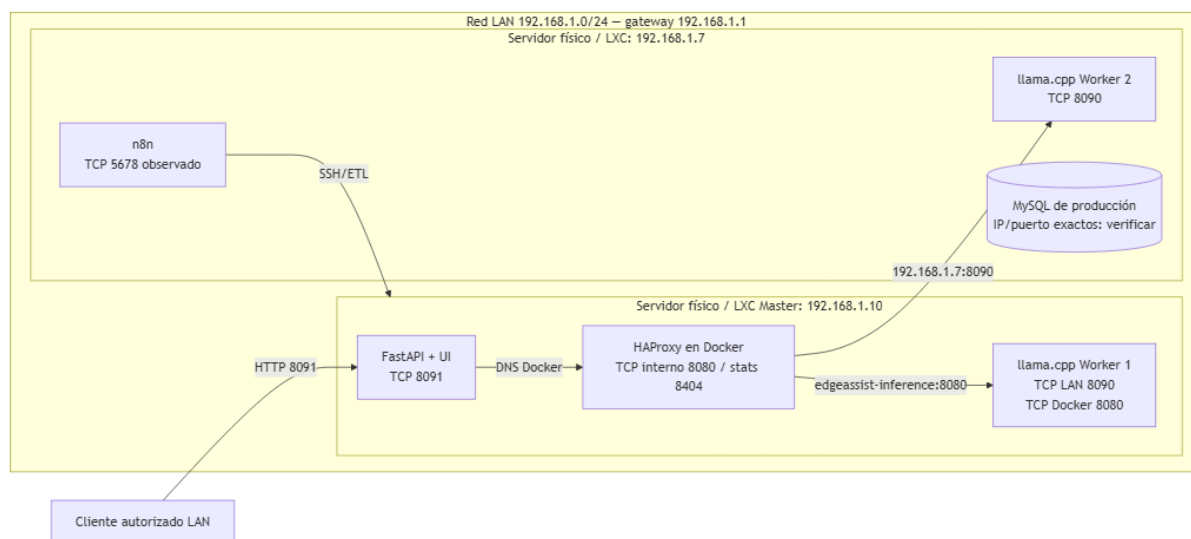
n8n tiene en su flujo exportado destinos SSH para dos Workers; el diagrama simplifica el destino lógico de la réplica. Durante el despliegue definitivo será necesario comprobar cuáles nodos conservan efectivamente ods.sqlite, pues la API centralizada en el Master es la que consulta la base en el prototipo actual. Los Workers de inferencia no necesitan la réplica para generar tokens si la consulta de herramientas ocurre en Fast API.

5.1 Plano de control y plano de datos

- Plano de control/aplicación: interfaz, FastAPI, configuración de herramientas, autenticación con X-API-Key y HAProxy.
- Plano de inferencia: dos instancias autónomas de Qwen atendiendo /v1/chat/completions.
- Plano de datos: MySQL de origen, exportaciones controladas, SQLite de ODS e índice documental FTS5.
- Plano de administración: Proxmox, Docker Compose, logs, respaldo y restauración de LXC.

6. Arquitectura de red y direccionamiento

6.1 Topología validada



En la topología validada, el servidor Master utiliza la dirección 192.168.1.10. Las direcciones de las instancias de Qwen en HAProxy son:

```

backend qwen_workers
    balance roundrobin
    option httpchk GET /health
    http-check expect status 200
    default-server inter 3s fall 2 rise 2
    server worker1 edgeassist-inference:8080 check
    server worker2 192.168.1.7:8090 check

```

La configuración completa de HAProxy utilizada durante el laboratorio también incluye un frontend HTTP en el puerto 8080 del contenedor, estadísticas en 8404 (dentro de la red Docker), timeouts largos para inferencia y chequeos cada 3 s, con umbrales fall 2 / rise 2.

6.2 Puertos y protocolos

Puerto	Protocolo	Extremo	Propósito	Exposición recomendada
8091	HTTP	Master .10 → FastAPI	UI /ui/, API /agente/preguntar	Solo LAN autorizada
8090	HTTP	LXC .10 y .7 → llama.cpp	Inferencia//health	Solo entre nodos autorizados
8080	HTTP	Red Docker → edgeassist-lb	Entrada interna del balanceador	No publicar en LAN
8080	HTTP	Red Docker → edgeassist-inference	Servidor llama.cpp dentro del contenedor	Solo red Docker
8404	HTTP	Red Docker → HAProxy	Estadísticas /stats	Administración privada
5678	HTTP	n8n observado en .7	Automatización	Restringir a administración
22	SSH	Destinos ETL	Transferencia/ ejecución del importador	Solo hosts autorizados

3306 u otro	TCP	MySQL de producción	Extracción ODS	Confirmar puerto real y ACL
--------------------	------------	----------------------------	-----------------------	------------------------------------

Red Docker: edgeassist-ai-net es una red local a cada motor Docker. Que dos equipos tengan una red con el mismo nombre no crea una red overlay entre ellos; el Worker remoto se accede por su IP LAN. No confundir IP internas 172.x.x.x con las direcciones de los nodos físicos.

6.3 Incidente de clonación: duplicación de MAC

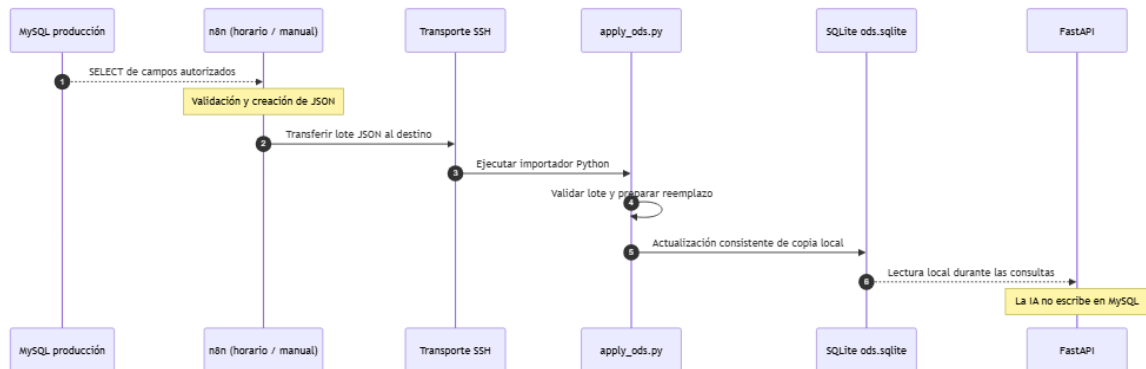
Al restaurar el backup del LXC en otro servidor, ambos contenedores conservaron la misma MAC 5a:e7:f5:9a:46:17, aunque tenían IP distintas (.7 y .10). Se observó ip neigh ... FAILED, Destination Host Unreachable y errores de conexión. El operador regeneró una MAC distinta para el LXC restaurado desde Proxmox. Después se confirmó HTTP 200 hacia /health en ambos destinos.

Este incidente evidenció la importancia de verificar que cada instancia virtualizada tenga identificadores únicos, especialmente direcciones MAC, direcciones IP y nombres de equipo, antes de diagnosticar fallas de conectividad como errores de aplicación.

6.4 Incidente de comunicación Master → Master dentro de Docker

Aunque el host .10 podía alcanzar 192.168.1.10:8090 y 192.168.1.7:8090, desde el contenedor FastAPI la conexión a la IP LAN del propio Master .10:8090 terminaba en timeout, mientras la conexión a .7:8090 daba HTTP 200. La solución aplicada fue direccionar el Worker local por DNS Docker (edgeassist-inference:8080) y el remoto por LAN (192.168.1.7:8090). Después, HAProxy mostró ambos Workers UP.

7. Flujo de sincronización ODS: MySQL → n8n → SQLite



7.1 Configuración del flujo de n8n

El archivo de exportación disponible se denomina `edgeassist-n8n-sincronizacion-rutas-corregidas.json` y contiene los siguientes componentes:

1. Activación manual y disparador programado.
2. Consulta MySQL de `ordenes_servicio`.
3. Nodo de código para validar y preparar el lote.
4. Conversión a JSON.
5. Transferencia por SSH a Worker 1 y Worker 2.
6. Comprobaciones por rama y ejecución remota de `apply_ods.py`.

Programación del archivo exportado: 0 6,12,18 *, es decir, a las 06:00, 12:00 y 18:00 en la zona horaria configurada por n8n. La continuidad de esta programación en producción requiere verificación sobre la instancia de n8n en ejecución.

7.2 Directorios operativos

```
/home/docker/edgeassist/
├─ data/
│   └─ ods.sqlite           # réplica local de ODS
├─ inbox/                  # lotes JSON recibidos por SSH
├─ bin/
│   └─ apply_ods.py         # importador del lote
├─ knowledge/
│   ├── catalogo.json
│   ├── documentos/
│   └─ knowledge.sqlite     # biblioteca FTS5
├─ ods-api/
│   ├── app.py              # FastAPI + agente
│   ├── compose.yaml
│   ├── .env                # secreto; no anexar al informe
│   └─ ui/                  # interfaz SERVIMEX
independiente
└─ haproxy/
    ├── haproxy.cfg
    └─ compose.yaml
```

7.3 Campos de ODS replicados

Se observaron 152 filas sincronizadas en la prueba y estos 21 campos de la tabla ODS:

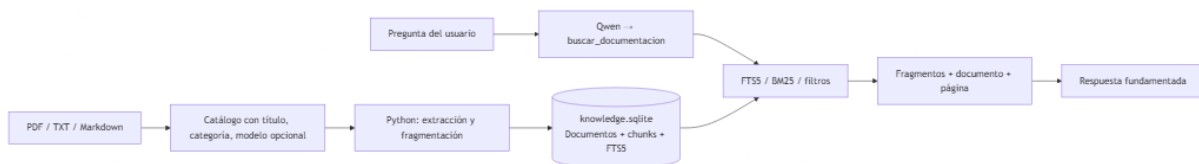
```
id, folio_interno, folio_consecutivo, folio_marca,
    marca, modelo, producto, status, sub_status,
estatus_agendamiento, falla, reparacion, tipo_servicio,
    tipo_garantia, tecnico_asignado, sucursal_id,
    fecha_programada, fecha_cita, fecha_reparacion,
    fecha_cierre, updated_at
```

La réplica SQLite representa el estado de los datos en el momento de la última sincronización; por tanto, no debe interpretarse como una consulta en tiempo real a MySQL.

El caso de prueba mostró sincronización a 2026-10-16T07:57:07.930187+00:00, con advertencia explícita sobre cambios posteriores. El número de filas no equivale al máximo ID; por ejemplo, la existencia de 152 filas no implica que la ODS con id=152 exista.

La solución distribuye las responsabilidades entre sus componentes: n8n actualiza los datos; FastAPI coordina las consultas; Qwen selecciona la herramienta y redacta las respuestas; y los módulos de acceso a SQLite realizan exclusivamente las lecturas autorizadas. La inferencia no modifica órdenes en MySQL.

8. Biblioteca documental independiente de ODS



Se registraron dos documentos en el catálogo de prueba:

- POL-001 — Política de garantías ficticia, categoria=politica, sin modelo de producto.
- MAN-SAM-001 — Manual Samsung autorizado incorporado por el operador, categoria=manual, modelo LH75BEDHVGFXZX.

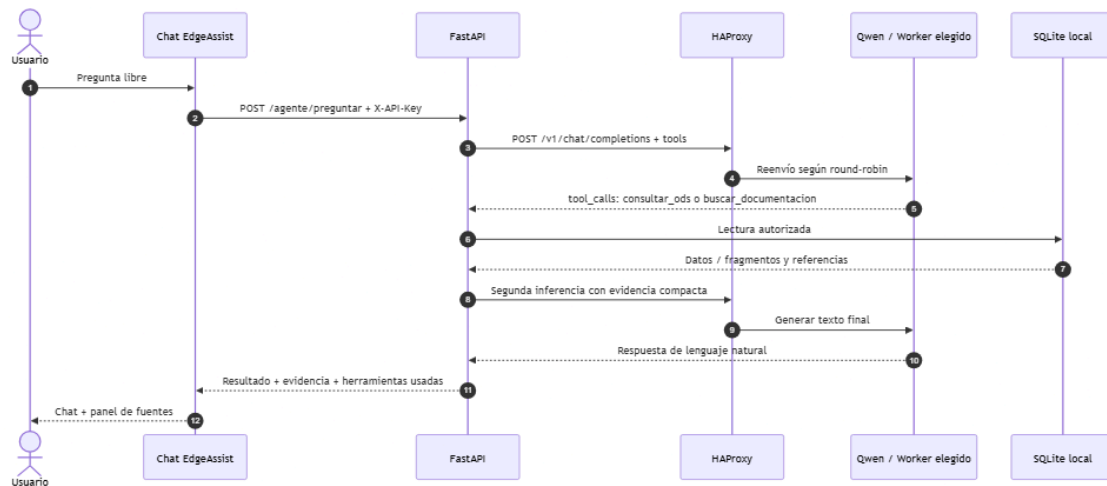
Se observaron 506 fragmentos indexados en conjunto. La búsqueda FTS5 directa sobre el manual produjo coincidencias con las palabras pantalla (175 fragmentos), imagen (71), negro (8) y diagnóstico (2); los conteos son del corpus en esa fecha y no implican relevancia clínica o técnica automática.

El modelo recibe fragmentos recuperados, no el PDF entero. Esto reduce memoria y permite citas por documento/página cuando el índice conserva ese dato. Es recuperación textual local, no entrenamiento del LLM ni una base de datos vectorial.

La recuperación documental presenta una limitación importante: los índices y las tablas de contenido pueden contener términos coincidentes con la consulta sin ofrecer instrucciones o

procedimientos útiles. El filtrado de relevancia y la comprobación de fuentes son necesarios. La política de ejemplo no constituye una política oficial de SERVIMEX.

9. Flujo del agente de IA



9.1 Herramientas

consultar_ods: busca una orden por identificador apropiado (id, folio_interno, etc.), devuelve estado, falla y demás campos permitidos. Se corrigió un error detectado: el modelo había enviado un ID numérico como si fuera un folio. La API incorpora validación para que ID interno 21 corresponda a id=21 y ODS-000021 al folio.

buscar_documentacion: consulta la biblioteca sin requerir ODS. Puede encontrar manuales de un modelo distinto al de cualquier orden existente y políticas sin modelo asociado. Se simplificaron/validaron argumentos después de observar confusiones de Qwen entre modelo, categoria y consulta.

9.2 Evolución de la API

- v3: primera selección de herramienta con Qwen; dificultad inicial de direccionamiento Docker y resolución de identificadores.
- v3.1: corrección de identificador interno frente a folio; prueba positiva ODS 21.
- v4: incorporación de biblioteca documental con FTS5.
- v4.1: mejoras de recuperación y normalización de categorías/modelos.
- v4.2: estabilización de la respuesta final, contexto compacto, rechazo de texto «tool_call» como salida final. Tres pruebas reportadas positivas: ODS, manual, política.

La versión 4.2 puede seleccionar herramientas y utilizar hasta dos en una misma respuesta. Sin embargo, todavía no se ha demostrado una planificación secuencial dinámica del tipo obtener ODS → descubrir modelo → decidir posteriormente qué manual consultar.

9.3 Modelo y rendimiento

Modelo utilizado: Qwen2.5-0.5B-Instruct, cuantizado Q4_K_M, servido por llama.cpp en Docker. La API de chat se consume por /v1/chat/completions; la capacidad de herramientas se habilitó en el servidor de inferencia con --jinja.

En el Dell de pruebas, el rendimiento observado pasó aproximadamente de 0,35 tokens/s a 23,54 tokens/s tras usar --load-mode none. Esta cifra es una observación del laboratorio original, no una medición de cada servidor físico actual ni del clúster. Deben anotarse comando, versión de llama.cpp, tamaño de contexto, CPU, temperatura y método de medición antes de usarla como comparación formal.

Se mencionó un contexto inicial --ctx-size 1024; se recomendó probar 4096 separadamente, pero no está confirmado el valor final. El cambio de tamaño de contexto no se considera implementado mientras no exista evidencia de su configuración.

10. Interfaz web y seguridad de interacción

EdgeAssist ofrece una interfaz HTML/CSS/JavaScript accesible mediante `http://192.168.1.10:8091/ui/` en la LAN. El estilo visual se adaptó a la aplicación SERVIMEX (menú azul, formularios y panel de consulta), pero no se integró funcionalmente al sistema SERVIMEX: no comparte sesión, backend ni autorización de usuarios empresariales.

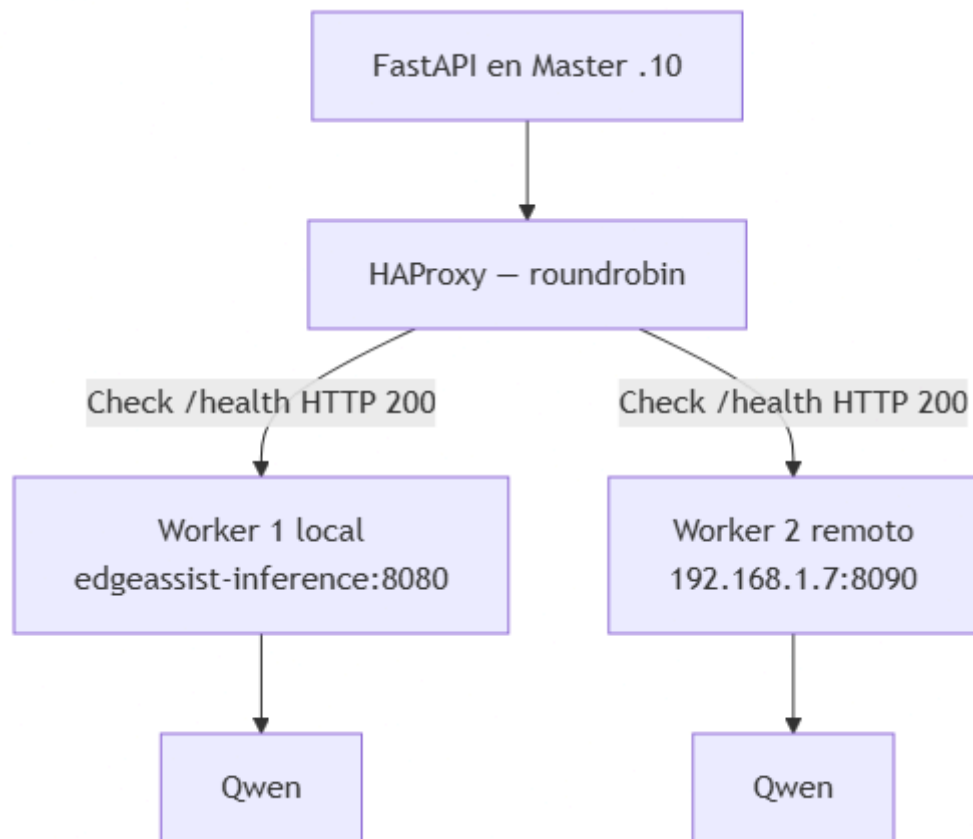
La interfaz se sirve como archivos estáticos desde FastAPI y utiliza el mismo origen para llamar a `/agente/preguntar`; se prescindió de Nginx y de un contenedor web adicional.

Autenticación observada: X-API-Key, variable `ODS_API_KEY`. Durante el laboratorio se recibió No autorizado porque la clave no estaba cargada en la terminal al ejecutar curl; tras cargar `.env`, la consulta respondió. Docker Compose carga variables para el contenedor, pero no exporta automáticamente las mismas variables al shell interactivo.

Limitaciones y controles recomendados: HTTP sin TLS expone tráfico y API Key ante observadores de la LAN; restringir red/usuarios, habilitar HTTPS antes del uso empresarial, implementar permisos por rol/sucursal, proteger `.env`, no publicar las estadísticas de HAProxy, registrar accesos con cuidado y no anexar órdenes sensibles completas en el informe público. Los documentos empresariales también pueden contener información restringida.

11. Balanceo, disponibilidad y tolerancia a fallos

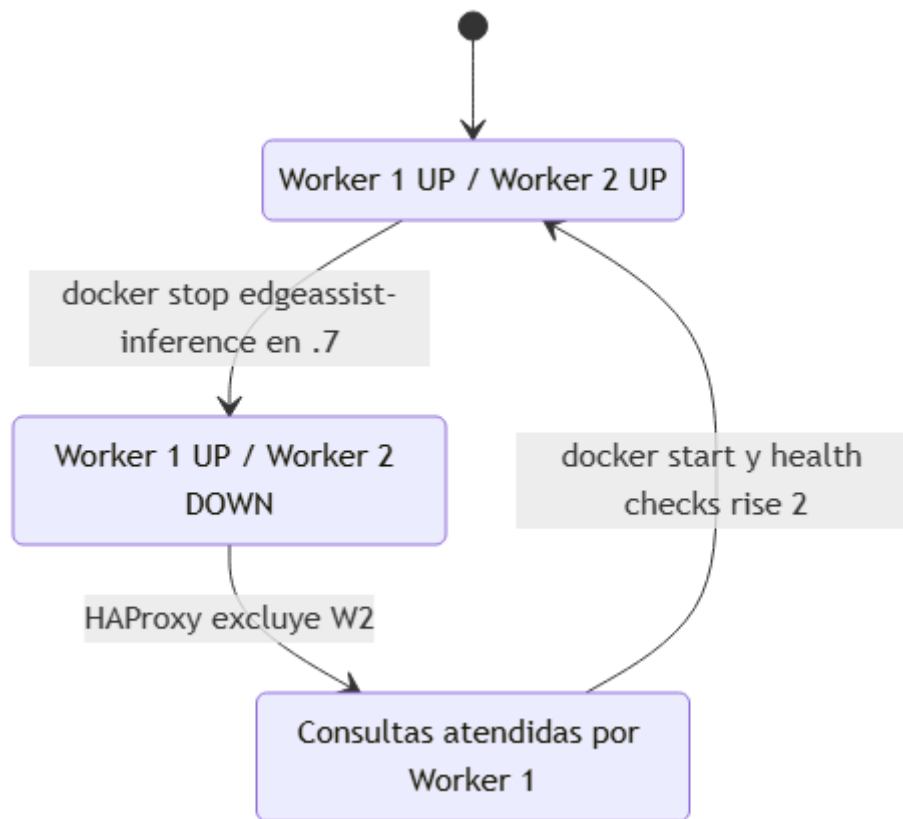
11.1 Balanceo implementado



En la validación inicial, HAProxy mostró worker1=UP y worker2=DOWN, pero la causa fue el acceso a la IP LAN del Worker local desde la red de Docker. Cambiar el destino local a edgeassist-inference:8080 llevó a ambos UP. El operador ejecutó pruebas de consulta y confirmó que el balanceo funcionaba.

Desde el punto de vista metodológico, el estado UP indica que HAProxy considera saludable al servidor de destino. Para demostrar el reparto efectivo de carga, es necesario comparar el contador stot de cada Worker antes y después de ejecutar varias consultas. Durante el desarrollo se reportó que el balanceo funcionó, pero no se conservó un registro numérico final de distribución. La documentación de la distribución por nodo debe complementarse con una captura o exportación CSV de las estadísticas reales.

11.2 Caída controlada y recuperación



El operador confirmó que la prueba de caída/recuperación funcionó. Este hecho respalda la continuidad de servicio ante pérdida de un Worker de inferencia durante las pruebas realizadas. No implica que una solicitud que ya estuviera en ejecución durante la caída se recupere sin error ni garantiza recuperación frente a todas las fallas posibles.

Prueba simétrica pendiente de validación: detener Worker 1 local y asegurar continuidad con Worker 2 remoto, sin detener FastAPI/HAProxy del Master. La arquitectura lo permite conceptualmente, pero falta consignar prueba y resultados concretos.

11.3 Punto único de fallo (SPOF)

Elemento	Redundancia actual	Consecuencia de caída
Inferencia Qwen	Dos instancias	Se tolera la caída de una, si la otra sigue sana
HAProxy en Master	No	Interrumpe selección de Workers
FastAPI en Master	No	Interrumpe agente y UI
Almacenamiento SQLite consultado por Master	No demostrado	Puede afectar respuestas basadas en datos
LXC / host Master	No	Interrumpe el servicio central
Switch/gateway LAN	No documentada	Posible interrupción de comunicaciones
Fuente MySQL	Externa al plano de inferencia	Se conservan lecturas de la última réplica válida si sigue disponible

12. Disponibilidad sin internet: límites exactos

La inferencia Qwen, FastAPI, las bases SQLite y los documentos pueden funcionar dentro de la LAN sin servicios externos de IA, una vez que modelos, imágenes Docker, paquetes y documentos ya están instalados.

13. Pruebas y evidencias disponibles

ID	Prueba	Evidencia observada	Resultado
T01	Replicar ODS MySQL → SQLite	152 filas, metadatos de sincronización	Positiva
T02	Consultar ODS 21	ODS-000021, falla=NO DA IMAGEN, modelo LH75BEDHVGFXZ X	Positiva
T03	Selección consultar_ods por Qwen	Herramienta registrada en JSON de respuesta	Positiva
T04	Consulta de política sin ODS	POL-001 obtenido desde FTS5	Positiva tras v4.2
T05	Consulta de manual sin ODS	MAN-SAM-001, modelo LH75BEDHVGFXZ X	Positiva tras v4.2
T06	Biblioteca documental	2 documentos catalogados y 506 fragmentos	Positiva
T07	Conectividad entre LXC	HTTP 200 a /health en .7 y .10	Positiva tras MAC única
T08	HAProxy: salud	Ambos Workers UP	Positiva
T09	Balanceo de inferencia	Confirmación funcional del operador	Positiva; anexar contadores
T10	Caída controlada de Worker	Confirmación funcional del operador	Positiva; anexar capturas/logs
T11	Prueba simétrica Worker 1	Sin salida conservada	Pendiente de evidencia
T12	Prueba formal sin WAN	Sin salida conservada	Pendiente

T13	Master en tercera mini PC	No ejecutado	Pendiente
T14	Rendimiento comparativo en hardware final	Sin datos medidos	Pendiente

13.1 Comandos de evidencia reproducibles

API y autenticación (sin imprimir la clave):

```
cd /home/docker/edgeassist/ods-api
set -a; source .env; set +a
curl -sS --max-time 180 http://127.0.0.1:8091/agente/preguntar \
  -H "X-API-Key: $ODS_API_KEY" \
  -H 'Content-Type: application/json' \
  -d '{"pregunta": "Consulta la orden de servicio con ID interno 21
e indica la falla registrada."}' \
  | python3 -m json.tool
```

Estadísticas de HAProxy por servidor:

```
docker exec edgeassist-ods-api python3 -c '
import urllib.request, csv, io
raw=urllib.request.urlopen("http://edgeassist-lb:8404/stats;csv",t
imeout=10).read().decode()
for row in csv.DictReader(io.StringIO(raw)):
    if row.get("svname") in ("worker1","worker2"):
        print(row["svname"], "estado:", row["status"], "sesiones
acumuladas:", row["stot"])
'
```

Prueba de caída controlada:

```
# SOLO en el equipo Worker 2 (192.168.1.7)
docker stop edgeassist-inference
```

```
# En Master (.10), después de dos health checks, verificar DOWN en estadísticas;  
# ejecutar una consulta de prueba desde UI o API.
```

```
# Recuperar en Worker 2 (.7)  
docker start edgeassist-inference
```

```
# Confirmar regreso a UP en HAProxy.
```

Como medida de seguridad operativa, las pruebas deben evitar la detención simultánea de los nodos y la interrupción de un LXC que aloje servicios adicionales, como n8n, SQL Server o Home Assistant.

14. Seguridad, integridad y buenas prácticas

- Consultar MySQL con permisos de lectura limitados; nunca escribir al origen desde el agente.
- Replicar solamente columnas autorizadas; evitar PII y datos sensibles no necesarios.
- Mantener el origen y fecha de la última sincronización en respuestas.
- Comprobar consistencia de importaciones antes de reemplazar la última SQLite válida.
- Mantener ods.sqlite y knowledge.sqlite montados en modo solo lectura desde FastAPI.
- Proteger documentos internos y restringir acceso al índice.

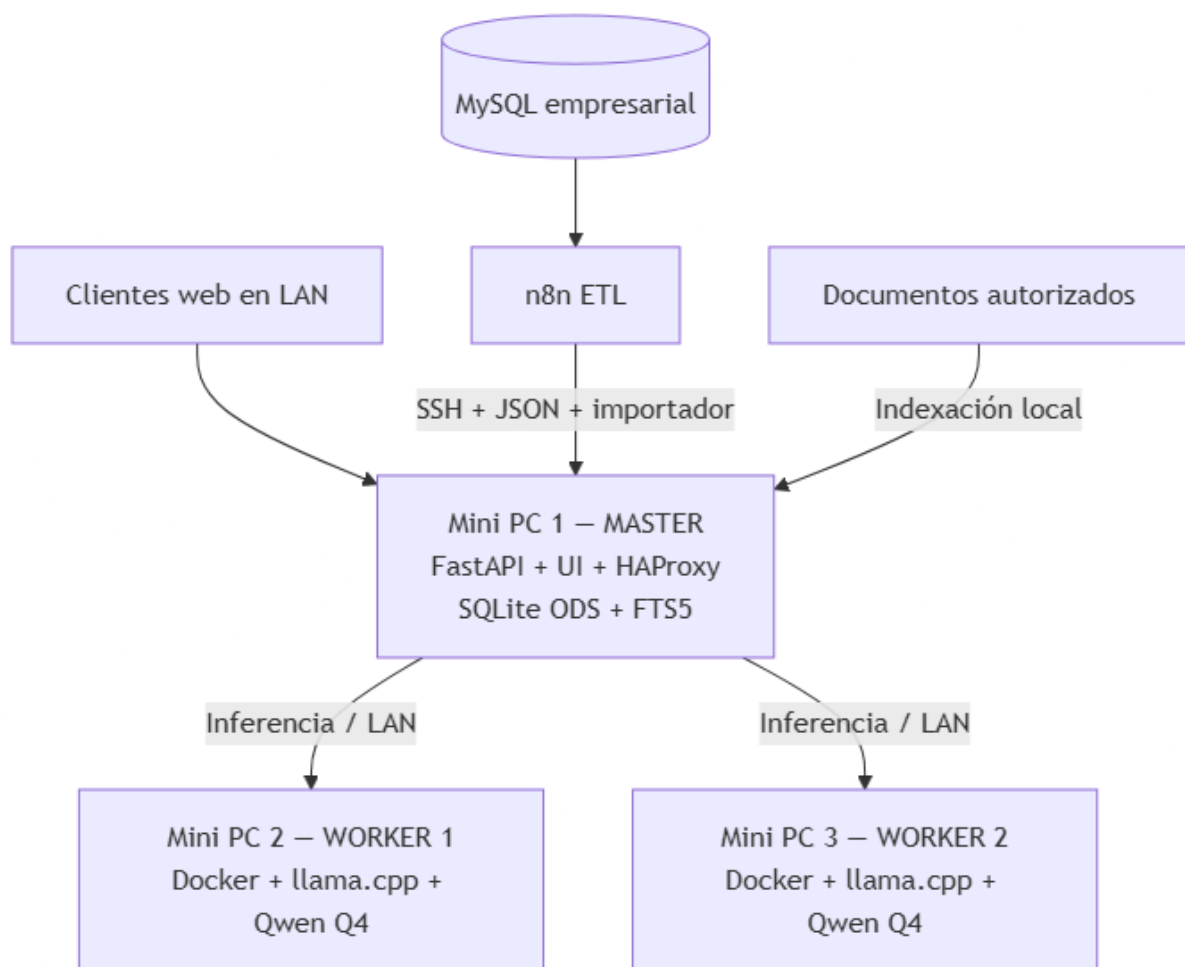
Servicios

- No publicar HAProxy 8404 ni las API internas fuera de las redes autorizadas.
- Guardar ODS_API_KEY en .env protegido y no en código cliente, capturas ni Git.
- Aplicar HTTPS y autenticación por usuario para uso empresarial; la API Key actual es una protección inicial, no control de acceso granular.
- Aplicar límites de consulta, registros mínimos, actualizaciones y copias de respaldo.
- Restringir SSH del ETL a claves/usuarios de propósito específico.
- Validar entradas de herramientas y parametrizar SQL; nunca ejecutar SQL arbitrario generado por el LLM.

Disponibilidad

- Establecer health checks independientes de la generación de texto.
- Medir latencia/tasa de error con uno y dos Workers.
- Documentar restauración de configuración de HAProxy y volúmenes.
- Resguardar el Master; la redundancia actual es parcial.

15. Propuesta de topología final: tres mini PC



Esta topología corresponde a la propuesta de despliegue final y todavía no se ha validado en tres mini PC independientes. La estrategia de transición es:

1. Respalidar los LXC y verificar compatibilidad de arquitectura CPU, espacio y memoria.
2. Instalar Master en la tercera mini PC, trasladando FastAPI, UI, SQLite y HAProxy.
3. Mantener los dos Workers existentes solo con llama.cpp/Qwen y sus puertos privados.
4. Cambiar las IP de destino en haproxy.cfg y LLM_URL=http://edgeassist-lb:8080 en FastAPI.
5. Definir ubicación final de n8n, MySQL, importador y catálogo documental.
6. Probar sincronización, fallos y recuperación, LAN sin WAN, reinicios y rendimiento en equipos definitivos.
7. Registrar especificaciones reales de CPU, RAM, discos SSD/HDD, consumo energético y temperatura.

La separación del Master y los Workers en equipos físicos distintos puede reducir la competencia por recursos de CPU y memoria durante la inferencia. No transforma automáticamente al Master en redundante.

16. Métricas para informe final

Se recomienda incorporar una tabla experimental con al menos 10 consultas representativas por escenario.

Métrica	Método sugerido	Valor actual
Tiempo total de respuesta (p50, p95)	time_total de curl, por escenario	Por medir
Tokens/s por Worker	Métricas de llama.cpp bajo misma carga	~23,54 tokens/s solo prueba Dell anterior
Disponibilidad durante caída	Consultas exitosas / intentos	Confirmación cualitativa; cuantificar
Tiempo de detección de fallo	Diferencia entre stop y estado DOWN	Aproximadamente 6 s según configuración; medir
Tiempo de recuperación	Diferencia entre start y estado UP	Por medir
Distribución de carga	HAProxy stot por Worker	Funcional; anexas valores

CPU/RAM	docker stats, Proxmox	Por medir
Consumo de energía	Medidor de pared	Por medir
Antigüedad de ODS	synced_at frente a consulta	Último dato observado: 2026-10-16 07:57 UTC
Calidad de recuperación documental	Preguntas con respuesta y fuente correctas	Tres pruebas funcionales; ampliar muestra

El algoritmo round-robin distribuye solicitudes HTTP de inferencia, pero no necesariamente preguntas completas: una única interacción del usuario puede originar varias llamadas al modelo. El balanceador no garantiza igualdad de carga computacional o tiempo de respuesta entre Workers.

17. Problemas encontrados y solución

Problema	Diagnóstico	Acción correctiva
Inferencia lenta inicialmente	~0,35 tokens/s en Dell	Parámetro --load-mode none; ~23,54 tokens/s observado
Error de herramienta ODS	id=21 tratado como folio 21	Validación/normalización de identificador
Manual y política sin resultados	Qwen confundía modelo/categoría/consulta	Mejoras de esquema y recuperación en v4.1
Error IA no disponible / marcador <tool_call>	Respuesta final o contexto inadecuados	Contexto compacto y filtrado de salida en v4.2
No autorizado	API Key sin exportar al shell	Cargar .env para pruebas CLI
LXC clonados sin comunicación	MAC duplicada, ARP FAILED	Generar MAC única desde Proxmox
HAProxy Worker local DOWN	Timeout hacia IP LAN del propio Master desde Docker	DNS Docker edgeassist-inference:8080
Balanceador sin reparto inicialmente	Worker marcado DOWN	Dos backends UP; pruebas de reparto y caída

18. Riesgos y limitaciones

1. SPOF central: Master, FastAPI y HAProxy no tienen redundancia.
2. Modelo pequeño: Qwen 0.5B puede confundir parámetros, fechas o consultas complejas; exige validación de herramientas y referencias.
3. Datos desfasados: MySQL y SQLite no se sincronizan continuamente, sino por lotes según programación.
4. Dependencia de LAN: la inferencia no requiere internet, pero sí disponibilidad de enlaces y red local.
5. Acceso HTTP: cifrado y permisos empresariales aún pendientes.
6. Recuperación léxica: FTS5 depende de palabras del documento y puede priorizar índices o términos genéricos.
7. Pruebas limitadas: falta batería sistemática de carga, falla del Worker 1, caída del Master, desconexión WAN y captura de métricas en hardware final.
8. Topología no definitiva: actualmente dos servidores; objetivo académico tres mini PC.
9. Copias clonadas: las instalaciones replicadas contienen servicios y configuraciones heredados; depurar roles al migrar.

19. Conclusiones

El desarrollo logró integrar inferencia local cuantizada, orquestación de datos MySQL→SQLite mediante n8n, recuperación documental FTS5 y un agente con selección automática de herramientas. La solución permite consultas de ODS y documentación sin obligar a relacionar manuales o políticas con una orden. La interfaz web facilita su uso fuera de la consola y, aunque reproduce el estilo de SERVIMEX, opera como una aplicación independiente.

La topología de laboratorio demostró que dos motores Qwen pueden permanecer accesibles detrás de HAProxy y que el sistema puede continuar respondiendo durante la indisponibilidad controlada de uno de ellos. El proyecto ofrece por tanto una prueba de concepto de Edge AI tolerante a fallos parciales.

Para concluir el cumplimiento integral del escenario académico es necesario efectuar el despliegue en las tres mini PC, separar Master y Workers, completar las mediciones y formalizar la seguridad de red y los límites de disponibilidad. Los resultados actuales no deben presentarse como disponibilidad total ni alta disponibilidad de toda la aplicación.

URL VIDEO

<https://drive.google.com/drive/folders/1PA9q00S7ivZ3PPGShfCIALNmodTEEcU7>

