Manual de operación - Agente de comunidad Abbante

Versión: 1.4

Fecha: 22 de julio de 2026

Audiencia: operadores de comunidad, administradores globales, comunicación, ventas y soporte

1. Propósito

El agente de comunidad ayuda a dar a conocer Abbante en X y a responder personas en Telegram, WhatsApp y Facebook Messenger. Trabaja con información aprobada, conserva una bitácora y entrega a una persona los asuntos que no debe resolver automáticamente.

El agente no consulta ni publica ubicaciones, documentos, placas, contenedores, cotizaciones ni datos de clientes en redes sociales.

2. Acceso y componentes

Entra en `https://abbante.online/administracion/login`. Existen dos perfiles autorizados:

- **Administrador global:** puede usar las seis pestañas y administrar toda la plataforma.
- **Operador de comunidad:** sólo ve **Atención > Conversaciones** y puede leer el historial, tomar o liberar una conversación y responder a la persona por el mismo canal.

El operador de comunidad no puede abrir resumen, publicaciones, prospectos, empresas, usuarios, licencias ni configuración. La restricción se aplica en el menú, en las páginas y en las funciones de la base de datos; escribir otra ruta manualmente no amplía sus permisos.

La pantalla tiene seis pestañas:

Pestaña	Objetivo
Resumen	Ver salud de canales y pendientes
Publicaciones	Crear, aprobar, rechazar y revisar contenido de X
Conversaciones	Leer contexto y tomar o devolver el control al robot
Prospectos	Revisar contacto, consentimiento, campaña y estado comercial
Conocimiento	Crear, aprobar, rechazar, retirar y consultar versiones de respuestas
Operación	Configurar cobertura, turnos, capacidad, asignación automática y SLA

El perfil **Operador de comunidad** recibe directamente la pestaña **Conversaciones**. Las otras pestañas sólo aparecen para un administrador global.

El indicador de cada canal usa estos colores:

- **Gris:** faltan credenciales.
- **Ámbar:** hay credenciales, pero el webhook o la suscripción requiere revisión.
- **Verde:** canal configurado y activo.

3. Publicaciones de X

El servicio genera borradores educativos para los siguientes temas: trazabilidad del contenedor, actores opcionales, subcontratación controlada, sostenibilidad y descarga diferida.

Todos los borradores nacen como **pendiente de aprobación**. Para publicar:

1. Abre **Publicaciones**.
2. Revisa que el texto sea correcto y tenga 280 caracteres o menos.
3. Confirma que no incluya datos de clientes ni afirmaciones no verificadas.
4. Revisa la fecha programada.
5. Pulsa **Aprobar**.
6. El servicio publica cuando llega la fecha y registra el identificador de X.

Rechaza el borrador si contiene tarifas no confirmadas, promesas absolutas, comparaciones sin evidencia, datos personales o información operacional.

Una aprobación repetida no genera dos publicaciones: la cola usa un reclamo atómico y conserva el resultado.

4. Telegram

En conversación privada el bot responde a los mensajes recibidos. En un grupo sólo responde cuando ocurre una de estas condiciones:

- la persona usa `/abbante`, `/ayuda` o `/start`;
- menciona al usuario del bot;
- responde directamente a un mensaje del bot.

El robot no copia ni procesa la conversación general del grupo. Si alguien publica teléfono o correo en el grupo, no repite el dato y dirige a mensaje privado.

El webhook se registra automáticamente al reiniciar el servicio cuando las variables de Telegram están completas.

5. WhatsApp

WhatsApp usa la Cloud API oficial de Meta y atiende conversaciones individuales iniciadas por la persona. El MVP no automatiza WhatsApp Web, grupos ni campañas salientes no solicitadas.

Cada webhook debe tener una firma válida. Una firma incorrecta produce HTTP 401 y el mensaje no se procesa.

Para captar un prospecto, el robot solicita nombre, empresa, medio de contacto y autorización expresa. Recibir un mensaje permite contestar esa conversación; no equivale por sí solo a consentimiento para seguimiento comercial posterior.

6. Facebook Messenger

Messenger atiende conversaciones privadas que una persona inicia desde la página de Facebook de Abbante. El callback productivo es:

`https://xaf.abbante.online/community-agent/webhooks/facebook`

El servicio valida el challenge de Meta y la firma `X-Hub-Signature-256`, ignora los ecos generados por la propia página y deduplica reintentos. La respuesta se envía con el token de la página; si el robot escala la conversación, el operador puede contestar desde la misma bandeja de AdminWeb.

La aplicación de Meta debe suscribirse a `messages`, `messaging_postbacks` y `messaging_referrals`. No se deben automatizar campañas salientes ni enviar mensajes fuera de las reglas vigentes de Messenger.

7. Base de conocimiento y escalamiento

El robot puede responder sobre:

- propósito de Abbante;
- flujo portuario;
- seguimiento y devolución del contenedor;
- viaje mínimo del chofer;
- actores opcionales;
- subcontratación y licencias;
- huella de carbono;
- descarga diferida;
- seguridad y privacidad;
- solicitud de demostración.

Si no encuentra una respuesta aprobada, informa que el equipo está trabajando para responderla y cambia la conversación a **handoff**. También escala quejas, cargos, emergencias, reclamos, daños, asuntos legales, solicitudes de baja e intentos de obtener instrucciones o secretos internos.

La fuente operativa está en Supabase y conserva clave, pregunta, respuesta, palabras de búsqueda, versión, vigencia, origen, creador, revisor y cantidad de usos. Los estados son:

- **Pendiente de aprobación:** aún no puede usarla el robot.
- **Aprobado y vigente:** puede participar como evidencia de una respuesta.
- **Rechazado:** requiere corrección.
- **Retirado:** deja de estar disponible sin borrar el historial.

Para agregar conocimiento desde un handoff, el operador abre la conversación, confirma la pregunta real y la respuesta que resolvió el caso, agrega palabras o frases de búsqueda y pulsa **Crear candidata**. Un administrador la revisa en **Conocimiento**. Aprobar una versión retira automáticamente la versión vigente anterior de la misma clave.

Cuando `OPENROUTER_API_KEY` está configurada, Llama puede interpretar una variante de la pregunta y seleccionar entre los artículos aprobados recuperados. La salida debe citar internamente identificadores de esa evidencia y el texto que se publica sigue siendo la respuesta aprobada, no texto libre del modelo. Si la evidencia no basta, el modelo falla o intenta citar una fuente no autorizada, el servicio usa el fallback aprobado o mantiene el handoff. Nunca usa conocimiento general del modelo para contestar.

8. Tomar una conversación

1. Abre **Conversaciones**.
2. Selecciona una fila en estado `handoff`.
3. Lee el historial completo y el motivo.
4. Pulsa **Tomar conversación** si todavía está en modo robot.
5. Escribe la respuesta en **Responder como operador** y envíala. La salida queda encolada, auditada y usa reintentos.
6. Confirma en el historial que el estado cambió de `pendiente` a `enviado`.
7. Devuelve el control al robot sólo cuando la base aprobada ya cubra las siguientes preguntas.

Mientras una conversación está en `handoff`, el robot registra entradas pero no responde automáticamente.

Asignación, turnos y SLA

Al crear un handoff, el servicio intenta asignarlo a la persona disponible con menor carga que cubra el canal, esté dentro de su turno y no haya alcanzado su capacidad. Si nadie está disponible, la conversación queda **Sin asignar** y el servicio vuelve a evaluarla periódicamente.

El administrador configura en **Operación**:

- canales atendidos por cada operador;
- días y horas del turno, incluida una jornada que cruza medianoche;
- capacidad simultánea;
- disponibilidad para asignación automática;
- SLA de primera respuesta y activación por canal.

Las conversaciones muestran responsable y vencimiento. Un SLA vencido se marca en rojo y genera una entrada de auditoría. El administrador puede reasignar una conversación. Un operador sólo puede atender las conversaciones que tiene asignadas o tomar una sin responsable dentro de sus canales.

Cuentas de atención

Un administrador global puede crear una cuenta individual para cada persona en **Personas > Usuarios y roles > Nuevo > Operador de comunidad**. La invitación y el cambio obligatorio de contraseña regresan a AdminWeb. Para editar nombre, correo, acceso o estado, selecciona al operador en la misma pantalla.

No compartas cuentas entre operadores. Para retirar acceso conserva el historial y usa **Suspender** o **Baja lógica**. La plantilla del rol elimina empresa, sindicato y cualquier otro permiso para mantener visible únicamente **Conversaciones**.

La cuenta inicial `operador.comunidad@abbante.mobile` se conserva como acceso de contingencia. Sólo sus credenciales están en el `.env` local ignorado por Git; las cuentas nuevas gestionan su contraseña mediante invitación, recuperación o contraseña temporal.

9. Prospectos y consentimiento

La bandeja muestra nombre, empresa, correo o teléfono, necesidad, campaña, puntaje y consentimiento.

Estados permitidos:

- **Nuevo:** capturado, sin seguimiento confirmado.
- **Contactado:** una persona inició el seguimiento.
- **Calificado:** existe necesidad y encaje comercial.
- **Convertido:** hay evidencia de conversión.
- **Descartado:** no procede o revocó consentimiento.

No cambies a **Convertido** sólo porque la persona pidió información. Una solicitud como “no me contacten” revoca el consentimiento y debe respetarse.

10. Variables de activación

Las variables se guardan únicamente en `/opt/abbante-community-agent/.env` del VPS. Nunca deben pegarse en chats, commits, SQL o manuales.

Telegram

- `TELEGRAM_BOT_TOKEN`
- `TELEGRAM_BOT_USERNAME`
- `TELEGRAM_WEBHOOK_SECRET`

WhatsApp Cloud API

- `WHATSAPP_ACCESS_TOKEN`
- `WHATSAPP_PHONE_NUMBER_ID`
- `WHATSAPP_VERIFY_TOKEN`
- `WHATSAPP_APP_SECRET`
- `WHATSAPP_WABA_ID`

Facebook Messenger

- `FACEBOOK_PAGE_ACCESS_TOKEN`
- `FACEBOOK_PAGE_ID` (Abbante: `1287664987752576`)
- `FACEBOOK_VERIFY_TOKEN`
- `FACEBOOK_APP_SECRET`
- `FACEBOOK_GRAPH_VERSION` (valor actual: `v25.0`)

La aplicación de Meta es **Abbante Community Agent** (`App ID 1012951521709303`). El callback de Messenger es:

> `https://xaf.abbante.online/community-agent/webhooks/facebook`

Las conversaciones de Messenger son individuales: cada respuesta se envía exclusivamente al PSID que originó el mensaje. El canal no se usa para campañas salientes ni permite que un usuario vea conversaciones de otros usuarios.

X API v2

- `X_USER_ACCESS_TOKEN`
- `X_USER_ID`

OpenRouter / Llama

- `OPENROUTER_API_KEY`
- `OPENROUTER_MODEL` (valor recomendado: `meta-llama/llama-3.3-70b-instruct`)
- `OPENROUTER_BASE_URL`
- `OPENROUTER_TIMEOUT_MS`

La llave de OpenRouter es opcional. Sin ella siguen funcionando las respuestas aprobadas deterministas, la base de conocimiento y el handoff.

Después de editar las variables:

```
cd /opt/abbante-community-agent
docker compose -f docker-compose.yml -f docker-compose.prod.yml up -d
curl -fsS https://xaf.abbante.online/community-agent/health
```

El resultado debe mostrar `true` únicamente en los canales cuyas variables estén completas.

11. Pruebas seguras

Antes de activar credenciales ejecuta desde el repositorio:

```
cd AgenteComunidad\CommunityAgent
npm test
cd ..\..
node AgenteComunidad/Simulator/community-agent-flow.mjs
```

El simulador valida salud, conocimiento, deduplicación, privacidad de grupo, firmas de WhatsApp y Facebook, respuesta de Messenger, handoff, asignación automática, Llama con evidencia y aprobación de X sin llamar a redes reales.

La verificación de Supabase incluye una prueba transaccional que revierte todos sus datos de QA:

```
cd supabase-api
node scripts/postdeploy-agente-comunidad-readonly.mjs
node scripts/qa-agente-comunidad-operacion-transaction.mjs
```

12. Incidentes

Síntoma	Acción
Canal gris	Completar las variables del canal y reiniciar
Canal ámbar	Revisar webhook, suscripción, permisos y logs
HTTP 401 en webhook	Corregir secreto o firma; no desactivar la validación
X no publica	Confirmar aprobación, fecha y permiso de escritura del token
Respuesta incorrecta	Tomar conversación, corregir conocimiento, probar y liberar
Llama no responde	Confirmar `ai.configured` en `/health`, revisar OpenRouter y conservar el fallback aprobado
Conversación sin asignar	Revisar canal, disponibilidad, turno y capacidad en **Operación**
SLA vencido	Asignar responsable, responder y revisar cobertura/capacidad
API temporalmente caída	Revisar cola; el servicio reintenta hasta cinco veces con backoff
Petición duplicada	No reenviar manualmente; la deduplicación evita dobles respuestas
Operador ve otra sección	Cerrar sesión y reportar el incidente; revisar `EsOperadorComunidad` y confirmar que los demás roles estén desactivados

Logs del servicio:

```
docker logs --tail 200 abbante-community-agent
```

13. Reversión

El servicio está aislado de XAF y de las aplicaciones móviles. Para desactivar toda salida basta con vaciar las variables sociales y recrear el contenedor. Las conversaciones, borradores y prospectos permanecen en Supabase.

Antes de cambiar Caddy conserva una copia del archivo. El despliegue inicial dejó un respaldo con el patrón:

`/opt/predictr/deploy/Caddyfile.bak-community-AAAAMMDDTHHMMSSZ`

No borres las tablas del agente para revertir el contenedor.