

Manual de Instalación y Operación — Grupo Banzai Veracruz

Asistente de ventas en español (chatbot RAG) para el inventario de autos seminuevos de Grupo Banzai Veracruz. Este manual cubre la instalación completa en un servidor (VPS), la carga del inventario, la publicación con dominio y SSL, la integración opcional con WhatsApp, y la operación diaria.

1. Requisitos previos

- **VPS** (servidor virtual) con Linux (Ubuntu 22.04 o similar), mínimo 2 vCPU y 2 GB de RAM. Recomendado 4 GB para OCR de imágenes.
- **Docker** y **Docker Compose** instalados:

```
curl -fsSL https://get.docker.com | sh
sudo usermod -aG docker $USER # cerrar sesión y volver a entrar
docker --version
docker compose version
```

- **Un dominio o subdominio** apuntando a la IP pública del VPS, por ejemplo `bot.grupobanzai.com` (registro DNS tipo A → IP del VPS).
 - Una **clave de OpenAI** (`OPENAI_API_KEY`) y/o **clave de Gemini** (`GEMINI_API_KEY`).
 - El archivo del inventario en PDF (lista de precios vigente).
-

2. Subir el código al VPS

Opción A — clonar desde Git (si el repositorio está disponible):

```
cd /opt
git clone <URL_DEL_REPOSITORIO> banzai-chatbot
cd banzai-chatbot
```

Opción B — copiar por SCP desde su computadora:

```
# Ejecutar en su computadora local (no en el VPS)
scp -r ./Erik usuario@IP_DEL_VPS:/opt/banzai-chatbot
```

Luego, en el VPS:

```
cd /opt/banzai-chatbot
```

3. Configurar el archivo `.env`

Copie la plantilla y edítela:

```
cp .env.example .env
nano .env
```

Valores importantes:

```
# Nombre de la marca (encabezado y título del sitio)
NEXT_PUBLIC_APP_NAME=Grupo Banzai Veracruz

# Credenciales del panel de administración (CÁMBIELAS)
ADMIN_EMAIL=admin@grupobanzai.com
ADMIN_PASSWORD=una_contraseña_larga_y_segura

# Proveedor de IA
OPENAI_API_KEY=sk-...
OPENAI_MODEL=gpt-5-nano

# Opcional: Gemini (si está presente, se usa primero)
GEMINI_API_KEY=
GEMINI_MODEL=gemini-flash-latest

# WhatsApp (opcional, ver sección 8)
WHATSAPP_VERIFY_TOKEN=
WHATSAPP_PHONE_NUMBER_ID=
WHATSAPP_ACCESS_TOKEN=
```

Importante: `NEXT_PUBLIC_APP_NAME` se incrusta al compilar la imagen. Si cambia el nombre, reconstruya con `docker compose up -d --build` para que el cambio aparezca en el navegador.

4. Levantar el sistema

```
docker compose up -d --build
```

Esto construye la imagen y arranca el contenedor `app` escuchando en el puerto **3000**. El servicio se reinicia solo si el servidor se reinicia (`restart: unless-stopped`).

Verifique que está vivo:

```
curl http://localhost:3000/api/health
# { "ok": true, "files": 0, "chunks": 0, ... }

curl http://localhost:3000/api/version
# { "version": "...", "model": "gpt-5-nano", "provider": "openai", ... }
```

5. Cargar el inventario inicial

Tiene dos formas de cargar la lista de precios:

A) Por script (recomendado para la primera carga):

1. Copie el PDF del inventario a la carpeta `uploads/` con el nombre `inventario.pdf` :

```
cp /ruta/a/PRECIOS_09_DE_MARZO_2026.pdf /opt/banzai-chatbot/uploads/inventario.pdf
```

El script también acepta un único PDF con cualquier nombre dentro de `uploads/` . Si hay **dos o más** PDFs, pedirá que renombre uno a `inventario.pdf` para evitar ambigüedad.

2. Ejecute la carga única:

```
docker compose exec app npm run seed
```

El script es **idempotente**: si ya hay archivos cargados, no hace nada.

B) Desde el panel de administración:

1. Abra <https://bot.grupobanzai.com/admin> e inicie sesión.
2. En la pestaña **Documentos**, suba el PDF (también acepta PDF, TXT, DOCX, CSV, XLSX, **PNG y JPG**).

Confirme que se cargó:

```
curl http://localhost:3000/api/health  
# files >= 1
```

6. Configurar Nginx como reverse proxy

Instale Nginx y cree el sitio:

```
sudo apt update && sudo apt install -y nginx  
sudo nano /etc/nginx/sites-available/banzai
```

Bloque `server` de ejemplo:

```
server {  
    listen 80;  
    server_name bot.grupobanzai.com;  
  
    client_max_body_size 25M; # permite subir PDFs/imágenes grandes  
  
    location / {  
        proxy_pass http://127.0.0.1:3000;  
        proxy_http_version 1.1;  
        proxy_set_header Upgrade $http_upgrade;  
        proxy_set_header Connection 'upgrade';  
        proxy_set_header Host $host;  
        proxy_set_header X-Real-IP $remote_addr;  
        proxy_set_header X-Forwarded-For $proxy_add_x_forwarded_for;
```

```

    proxy_set_header X-Forwarded-Proto $scheme;
    proxy_cache_bypass $http_upgrade;

    # Necesario para el streaming SSE del chat
    proxy_buffering off;
    proxy_read_timeout 300s;
}
}

```

Active el sitio y recargue:

```

sudo ln -s /etc/nginx/sites-available/banzai /etc/nginx/sites-enabled/banzai
sudo nginx -t
sudo systemctl reload nginx

```

7. SSL con Let's Encrypt (Certbot)

```

sudo apt install -y certbot python3-certbot-nginx
sudo certbot --nginx -d bot.grupobanzai.com

```

Siga las indicaciones (correo, aceptar términos, redirección HTTP→HTTPS). Certbot edita el bloque de Nginx y agrega el certificado automáticamente.

Renovación automática (Certbot ya instala un temporizador). Para probar:

```

sudo certbot renew --dry-run

```

8. Configurar WhatsApp Business Cloud API (opcional)

El sistema incluye un webhook en `/api/whatsapp/webhook` que usa el mismo motor de RAG/IA que el chat web.

1. Crear app en Meta for Developers

- Vaya a <https://developers.facebook.com/> → *Mis Apps* → *Crear app*.
- Tipo: **Empresa**. Agregue el producto **WhatsApp**.

2. Obtener el Phone Number ID

- En *WhatsApp* → *Configuración de la API*, copie el **Identificador del número de teléfono** (Phone Number ID). Póngalo en `WHATSAPP_PHONE_NUMBER_ID` dentro de `.env`.

3. Generar un Access Token permanente

- Cree un *Usuario del sistema* en *Business Settings* → *Usuarios del sistema*, asígnele la app y genere un token con los permisos `whatsapp_business_messaging` y `whatsapp_business_management`.
- Copie el token a `WHATSAPP_ACCESS_TOKEN`.

4. Configurar el Webhook

- Defina un token de verificación a su elección (ej. `banzai_verify_2026`) y póngalo en `WHATSAPP_VERIFY_TOKEN`.
- Reinicie el contenedor para aplicar las variables:

```
docker compose up -d
```

- En *WhatsApp* → *Configuración* → *Webhooks*, configure:
 - **Callback URL:** `https://bot.grupobanzai.com/api/whatsapp/webhook`
 - **Verify Token:** el mismo valor de `WHATSAPP_VERIFY_TOKEN`.
- Meta hará una petición `GET` de verificación; debe responder con éxito.

5. Suscribirse al evento `messages`

- En la misma pantalla de Webhooks, en el campo **WhatsApp Business Account**, suscríbase al campo **messages**.
- Envíe un mensaje de prueba al número de WhatsApp; el bot debe responder.

9. Actualizar el inventario cuando cambien los precios

Cuando reciba una nueva lista de precios:

1. Suba el nuevo PDF desde el **panel admin** (pestaña Documentos). Puede eliminar el archivo anterior desde la misma pestaña para que el chatbot solo use los precios vigentes.

O bien, por línea de comandos, reemplace el archivo y vuelva a indexar borrando la base para forzar una carga limpia (ver sección 12 para respaldos antes de hacerlo):

```
cp nueva_lista.pdf uploads/inventario.pdf
# Detener, limpiar la base y volver a sembrar:
docker compose exec app rm -f /app/db/app.db
docker compose restart app
docker compose exec app npm run seed
```

2. Verifique con una pregunta de prueba en el chat (por ejemplo, el precio de un modelo cuyo precio cambió).

10. Cambiar de `gpt-5-nano` a `gpt-5-mini` (mayor calidad)

El modelo es configurable por variable de entorno, **sin cambios de código**:

1. Edite `.env`:

```
OPENAI_MODEL=gpt-5-mini
```

2. Reinicie:

```
docker compose up -d
```

3. Confirme:

```
curl https://bot.grupobanzai.com/api/version
# "model": "gpt-5-mini"
```

`gpt-5-mini` da respuestas de mayor calidad a un costo algo mayor. `gpt-5-nano` es más económico y rápido. Puede alternar cuando lo desee.

11. Solución de problemas

- **Ver logs en vivo:**

```
docker compose logs -f app
```

- **Ubicación de la base de datos:** `db/app.db` (SQLite). Se conserva entre reinicios gracias al volumen montado.
- **Archivos cargados:** carpeta `uploads/`. Texto procesado: `parsed/`.
- **Reiniciar el contenedor:**

```
docker compose restart app
```

- **Reconstruir tras un cambio de código o de `NEXT_PUBLIC_APP_NAME`:**

```
docker compose up -d --build
```

- **Primera subida de imagen (OCR) tarda más:** la primera vez que se procesa un PNG/JPG, Tesseract descarga los modelos de idioma `spa+eng` (~25 MB). Es un retardo único; las siguientes imágenes son rápidas.
- **El bot responde "no tengo esa unidad":** revise que el inventario esté cargado (`/api/health` con `files >= 1`) y que el modelo preguntado exista en la lista vigente.
- **El chat no transmite (se queda "pensando"):** confirme en Nginx que `proxy_buffering off;` está presente (necesario para SSE).

12. Respaldos recomendados

Respalde periódicamente (por ejemplo, diariamente vía `cron`):

- `db/` — base de datos con inventario, conversaciones y leads.
- `uploads/` — archivos originales cargados.
- `.env` — credenciales y configuración (guárdelo en un lugar seguro, contiene claves de API).

Ejemplo de respaldo manual:

```
tar czf banzai-backup-$(date +%F).tar.gz db uploads .env
```

Para restaurar, detenga el contenedor, descomprima en la carpeta del proyecto y vuelva a levantar:

```
docker compose down
tar xzf banzai-backup-2026-03-09.tar.gz
docker compose up -d
```