Manual de uso (Editor, Desarrollador y Mantenedor)

Este documento resume cómo utilizar, desarrollar y operar el cotizador. Si buscas instalación y despliegue paso a paso, revisa `DEPLOYMENT.md`.

- URL pública típica del frontend: `https://cotizador.aysafi.com`
- URL pública típica del backend: `https://emqx.aysafi.com`

Editor (uso diario de la app)

Objetivo: crear, revisar, aceptar o rechazar cotizaciones y enviar documentos por correo.

Ingreso y UI general

- Abre el sitio y verás el listado de cotizaciones más recientes.
- Los tiempos aparecen en español corto (ej.: “hace 1 min”, “ayer”, “en 3 h”).
- Cuando una cotización requiere atención, verás una etiqueta azul “Revisar”.
- La lista es compacta: nombre de cliente truncado, columnas alineadas y fechas normalizadas.

Crear y editar

- Completa los campos del formulario y guarda. El listado se actualiza en tiempo real.
- Si adjuntas logos, respeta el límite de tamaño (la app restringe a 5 MB). Los archivos quedan en `outputs/` del servidor.

Actualización en tiempo real

- La app usa eventos del servidor (SSE). Cuando hay cambios, el renglón parpadea sutilmente para indicar actualización.
- Si pierdes conexión brevemente (reinicios del servidor), se reconecta sola; no necesitas refrescar la página.

Aceptar o rechazar cotizaciones

- Aceptar: el botón “Aceptar” aparece cuando se cumplen las condiciones (p. ej., anticipo requerido en CLP según la lógica de negocio vigente).
- Rechazar: al rechazar, se solicita motivo opcional. Esto marca la cotización con “Revisar” para seguimiento posterior.
- Copiar: al usar “Copiar” como base de una nueva, la bandera “Revisar” se limpia.

Correos y PDFs

- Al guardar/actualizar o al aceptar/rechazar, la app puede enviar correos con HTML enriquecido.
- Los PDF incluyen marcas de agua condicionales según el estado (borrador, aceptada, rechazada) y el logo si existe.
- Si un correo falla, consulta al Mantenedor (ver sección “SMTP y correo”).

Página de diagnóstico (/admin/config)

- Página informativa que muestra la configuración efectiva que usa el frontend.
- No modifica nada; sirve para confirmar que el frontend apunta al backend correcto.
- El acceso administrativo requiere token cuando el backend así lo exige (ver Mantenedor → Seguridad).

Desarrollador (trabajo en local y contribución)

Objetivo: montar el entorno local, entender la estructura y agregar cambios de forma segura.

Estructura del repositorio (resumen)

- `backend/`: Express (API, SSE, emails, PDFs, uploads). `server.js` inicia HTTP/HTTPS.
- `frontend/`: React + Vite. Carga configuración en runtime desde `/config.js`.
- `outputs/`: JSON, QR y PDFs generados. En producción puede moverse fuera del repo.
- `release/`: paquetes para subir a cPanel (generados por script).

Instalación y desarrollo

1. Instala dependencias en la raíz y frontend.

```
npm install
cd frontend
npm install
```

1. Arranca todo en dev (backend + frontend):

```
npm run dev
```

- El frontend corre en `http://localhost:5173` y se proxea al backend `http://localhost:5000` para `/api`, `/outputs` y `/config.js`.

Configuración en runtime (sin rebuild)

- El backend sirve `/config.js` con `window.__APP_CONFIG__ = { API_BASE, FRONTEND_URL }`.
- El frontend debe obtener URLs vía utilidades: `apiUrl('/ruta')` y `eventsUrl()`.
- Importante: `config.js` es script clásico; NO uses `export`. Ejemplo válido:

```
window.__APP_CONFIG__ = {
  API_BASE: 'https://emqx.aysafi.com',
  FRONTEND_URL: 'https://cotizador.aysafi.com'
};
```

Scripts útiles (raíz)

- `npm run dev`: backend + frontend en paralelo con recarga.
- `npm run backend`: solo backend con nodemon.
- `npm run frontend:dev`: solo frontend Vite.
- `npm run frontend:build`: build de producción (Vite).
- `npm run build:checked`: lint y build de frontend.
- `npm run test`: corre pruebas backend y frontend.
- `npm run package:cpanel`: prepara carpeta `release/cpanel-*/` con `frontend/dist`, `frontend/.htaccess` y `frontend/config.js.template`.

Pruebas y calidad

- Lint: `npm run lint` (JS/JSX).
- Tests backend: `npm run test:backend` (Node test runner).
- Tests frontend: `npm run test:frontend` (Vitest). Si ves warnings de React `act(...)`, revísalos pero no suelen bloquear.

Convenciones

- Ramas: usa `develop` para integrar y `main` para producción (ver `BRANCH_FLOW_QUICKSTART.md`).
- UI de tiempos: usa el formateador relativo ya incluido (español corto) para consistencia.
- Eventos en tiempo real: suscríbete vía `EventSource(eventsUrl())` con backoff (ya implementado).
- Al consumir la API: no codifiques URLs absolutas; usa `apiUrl()`.

Mantenedor (operación y soporte)

Objetivo: desplegar, mantener y resolver incidencias.

Despliegue

- Guía completa: `DEPLOYMENT.md`.
- Modos soportados:
  - Monolítico en cPanel (Node.js/Passenger) sirviendo también la SPA.
  - Separado: frontend estático en cPanel y backend en VPS con Nginx (recomendado) o HTTPS directo en Node.

Variables de entorno (backend)

- Esenciales: `JWT_SECRET`, `FRONTEND_URL`, `PUBLIC_API_BASE` (vacío si monolítico).
- Admin opcional: `ADMIN_PASSWORD` (protege diagnósticos y permite `POST /api/admin/login`).
- SMTP: `SMTP_HOST`, `SMTP_PORT`, `SMTP_USER`, `SMTP_PASS`.
- Logs HTTP: `MORGAN_FORMAT`.
- Salidas: `OUTPUTS_DIR` (si prefieres fuera del repo).
- HTTPS directo (opcional): `HTTPS=true`, `HTTPS_PORT=8443`, `TLS_CERT_FILE`, `TLS_KEY_FILE`, `TLS_CA_FILE`.

HTTPS y dominios

- Evita “Mixed Content”: si la página es `https://`, el backend también debe ser `https://`.
- Recomendado: Nginx en 443 con certificado y proxy a Node. Para pruebas, puedes usar HTTPS directo en Node.

Diagnóstico y seguridad

- `/admin/config`: visualiza la config efectiva del frontend y prueba `/api/config` del backend.
- Si `ADMIN_PASSWORD` está definido, obtén token con `POST /api/admin/login` y úsalo para endpoints/admin según corresponda.
- SSE: el backend expone `/api/events` (Content-Type: `text/event-stream`). Si hay proxy, desactiva buffering y amplía timeouts.

SMTP y correo

- Verificación: `npm run verify:smtp` y `npm run send:test-email`.
- Errores comunes: `ETIMEDOUT`, `Greeting never received` → revisa host/puerto/TLS y credenciales; valida puertos en el proveedor.

Archivos generados y backups

- PDFs/QR/JSON quedan en `outputs/` o en la ruta definida por `OUTPUTS_DIR`.
- Respáldalos periódicamente. Para regenerar PDFs: `npm run regenerate-pdfs` o uno puntual con `npm run regenerate-pdf`.

Problemas frecuentes (y soluciones)

- “Mixed Content bloqueado”: corrige `API_BASE` a `https://` y reintenta.
- “config.js Unexpected token 'export'”: reemplaza por script clásico con `window.__APP_CONFIG__`.
- “Conexión SSE inestable tras proxy”: en Nginx usa `proxy_buffering off` y `proxy_read_timeout`/`proxy_send_timeout` altos.
- “No llegan correos”: valida SMTP y puertos, revisa logs del backend (morgan y nodemailer).

Registro y monitoreo

- Revisa logs del proceso (systemd journal o consola de cPanel).
- Activa `MORGAN_FORMAT=combined` para trazabilidad en producción.

Referencias rápidas

- Despliegue detallado: `DEPLOYMENT.md`.
- Flujo de ramas: `BRANCH_FLOW_QUICKSTART.md`.
- Paquetes de release para cPanel: `npm run package:cpanel`.
- Diagnóstico de config: visita `/admin/config`.