Guía de despliegue

Esta aplicación tiene un backend Node.js/Express y un frontend React (Vite). Puedes desplegarla de dos formas:

- A) Monolítica en cPanel (backend y frontend juntos)
- B) Separada: frontend en cPanel (https://cotizador.aysafi.com) y backend en un VPS (https://emqx.aysafi.com)

A continuación están ambos flujos. Si buscas el escenario "separado", ve directo a la sección B.

A) Quickstart cPanel (monolítico: backend + frontend)

En cPanel usaremos Application Manager (Node.js/Passenger) para correr el backend y servir la SPA.

1. Obtener el paquete listo (release)

- En tu entorno local puedes generar el paquete con `npm run package:cpanel`. Se crea una carpeta `release/cpanel-<timestamp>/` con un ZIP listo.
- Alternativamente, usa el ZIP ya generado que encuentres en `release/` (por ejemplo: `release/cpanel-20250924-001439.zip`).

1. Subir y descomprimir en cPanel

- Crea una carpeta en tu home (ej. `cotizador`).
- Sube el ZIP y descomprímelo ahí. Debes ver `backend/`, `frontend/dist/`, `.env.example`, `package.json`, etc.

1. Crear aplicación Node.js en cPanel

- Application root: la carpeta donde descomprimiste (ej. `/home/usuario/cotizador`).
- Application URL: el subdominio elegido (ej. `https://cotizador.tudominio.cl`).
- Startup file: `backend/server.js`.
- Versión de Node: 20.x.

1. Variables mínimas (en Application Manager)

- `JWT_SECRET`: secreto largo y único.
- `FRONTEND_URL`: ej. `https://cotizador.tudominio.cl`.
- (Opcional) `PUBLIC_API_BASE`: déjalo vacío si frontend y backend comparten dominio.
- (Opcional) `OUTPUT_DIR`: por defecto usa `backend/outputs` (junto a `app.js`).
- (SMTP si enviarás correos) `SMTP_HOST`, `SMTP_PORT`, `SMTP_USER`, `SMTP_PASS`, `SMTP_FROM`.

1. Instalar dependencias (Terminal de la App)

- Ejecuta en la raíz de la app: `npm ci`.
- Luego: `cd frontend && npm ci && cd ..`.

1. Reiniciar y probar

- Reinicia la app desde Application Manager.
- Abre la URL pública y verifica:
  - La página carga sin errores.
  - Rutas profundas de la SPA como `/admin/login` funcionan.
  - `/outputs/*` sirve archivos si existen.
  - `/config.js` responde (runtime config) — opcional si usas config embebida.

1. Seguridad y notas

- Si sirves el frontend con Apache (docroot estático) en lugar de Passenger, usa el `.htaccess` de `frontend/.htaccess` incluido en el paquete para CSP y fallback SPA.
- En modo monolítico (Node/Passenger), los headers de `.htaccess` no aplican.

1. Permisos

- Asegura que `backend/outputs/` sea escribible por la app.
- No subas `.env` al repo; usa variables de entorno en cPanel.
- Rota `JWT_SECRET` y credenciales SMTP tras pruebas.

Alternativa (si no usas el ZIP de release):

- Sube el repo (sin `node_modules/`), construye el frontend con `npm run frontend:build` y continúa desde el paso 3.

SMTP troubleshooting (Nodemailer)

- `ETIMEDOUT`/`Greeting never received`: revisa host/puerto/seguridad; prueba `npm run verify:smtp` y `npm run send:test-email`.

B) Separado: frontend en cPanel y backend en VPS

Objetivo: servir el frontend estático en `https://cotizador.aysafi.com` (cPanel) y el backend en `https://emqx.aysafi.com` (VPS). El frontend llamará al backend vía `PUBLIC_API_BASE` usando `window.__APP_CONFIG__` cargado desde `config.js`.

1) Backend en VPS (https://emqx.aysafi.com)

Requisitos en el VPS (Ubuntu/Debian típico):

- DNS del dominio `emqx.aysafi.com` apuntando a la IP del VPS
- Node.js 20 y npm
- Nginx como reverse proxy con SSL (Let’s Encrypt)
- Acceso SSH y permisos para crear un servicio (systemd)

Pasos:

1. Clonar e instalar dependencias

```
sudo mkdir -p /opt/cotizador && sudo chown $USER:$USER /opt/cotizador
cd /opt/cotizador
git clone https://github.com/asdrubalfuentes/cotizador .
npm ci
```

1. Variables de entorno del backend (archivo `.env` en `/opt/cotizador`)

```
# URL públicas
FRONTEND_URL=https://cotizador.aysafi.com
PUBLIC_API_BASE=https://emqx.aysafi.com

# Seguridad
JWT_SECRET=pon-aqui-un-secreto-largo-unico
ADMIN_PASSWORD=elige-una-clave-admin

# SMTP (si enviarás correos)
SMTP_HOST=smtp.tu-proveedor.com
SMTP_PORT=587
SMTP_USER=usuario@tu-dominio.com
SMTP_PASS=tu-pass

# Otros (opcionales)
OUTPUT_DIR=/var/lib/cotizador/outputs
MORGAN_FORMAT=combined
```

1. Directorio de salidas (si usas ruta externa)

```
sudo mkdir -p /var/lib/cotizador/outputs
sudo chown -R $USER:$USER /var/lib/cotizador
```

1. Servicio systemd (opcional recomendado)

Archivo: `/etc/systemd/system/cotizador.service`

```
[Unit]
Description=Cotizador Backend
After=network.target

[Service]
Type=simple
WorkingDirectory=/opt/cotizador
Environment=NODE_ENV=production
ExecStart=/usr/bin/node backend/server.js
Restart=always
RestartSec=5
```

```
# User=cotizador   ; si creas un usuario de servicio

[Install]
WantedBy=multi-user.target
```

Activar e iniciar:

```
sudo systemctl daemon-reload
sudo systemctl enable cotizador --now
```

1. Nginx reverse proxy con SSL y SSE

Archivo: `/etc/nginx/sites-available/cotizador-backend`

```
server {
  listen 80;
  server_name emqx.aysafi.com;
  location /.well-known/acme-challenge/ { root /var/www/html; }
  location / { return 301 https://$host$request_uri; }
}

server {
  listen 443 ssl http2;
  server_name emqx.aysafi.com;

  ssl_certificate     /etc/letsencrypt/live/emqx.aysafi.com/fullchain.pem;
  ssl_certificate_key /etc/letsencrypt/live/emqx.aysafi.com/privkey.pem;

  client_max_body_size 10m; # subir logos (la app limita 5MB)

  location / {
    proxy_pass http://127.0.0.1:3000; # puerto donde corre Node
    proxy_http_version 1.1;
    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;

    # SSE (EventSource): sin buffer y con timeouts amplios
    proxy_read_timeout 3600s;
    proxy_send_timeout 3600s;
    proxy_buffering off;
  }
}
```

Activar sitio y recargar:

```
sudo ln -s /etc/nginx/sites-available/cotizador-backend /etc/nginx/sites-
enabled/
sudo nginx -t && sudo systemctl reload nginx
```

1. Comprobaciones rápidas del backend

- `curl https://emqx.aysafi.com/` → JSON `{ ok: true, ... }`
- `curl -I https://emqx.aysafi.com/api/events` → `Content-Type: text/event-stream`

Notas CORS/SSE:

- CORS: el backend usa `cors()` abierto; para restringir: `cors({ origin: 'https://cotizador.aysafi.com' })`.
- SSE: con `proxy_buffering off` y timeouts altos, EventSource funciona estable tras proxies.

HTTPS directo en Node (alternativa)

Si no deseas usar Nginx delante, el backend puede exponer HTTPS directamente (útil para pruebas o despliegues simples). Ya viene soportado en `backend/server.js`.

Requisitos:

- Certificados presentes en `/etc/ssl/emqx/` con estos nombres por defecto:
  - `emqx.crt` (cert)
  - `emqx_key.rsa` (key)
  - `emqx.ca-bundle.crt` (chain/ca)
- O bien, define rutas personalizadas mediante variables de entorno.

Pasos:

1. Coloca los archivos en `/etc/ssl/emqx/` y asegúrate de que el usuario que ejecuta Node pueda leerlos.
2. Variables en `.env` (ejemplo):

```
HTTPS=true
HTTPS_PORT=8443
TLS_CERT_FILE=/etc/ssl/emqx/emqx.crt
TLS_KEY_FILE=/etc/ssl/emqx/emqx_key.rsa
TLS_CA_FILE=/etc/ssl/emqx/emqx.ca-bundle.crt
```

1. Reinicia el servicio. Verás logs como:

```
[HTTPS] Cotizador backend running on port 8443
[HTTPS] cert: /etc/ssl/emqx/emqx.crt
[HTTPS] key : /etc/ssl/emqx/emqx_key.rsa
[HTTPS] ca  : /etc/ssl/emqx/emqx.ca-bundle.crt
```

1. Abre el firewall para el puerto 8443, o preferiblemente mantén Nginx en 443 y haz proxy a 8443.

Frontend (cuando usas HTTPS directo):

- En el `config.js` del frontend usa `https://emqx.aysafi.com:8443` como `API_BASE`, o configura Nginx en 443 para ocultar el puerto público y dejar `API_BASE=https://emqx.aysafi.com`.

2) Frontend estático en cPanel (https://cotizador.aysafi.com)

1. Build local y subida

- Ejecuta: `npm run frontend:build`
- En cPanel, crea el subdominio `cotizador.aysafi.com` apuntando a un docroot (ej. `/home/usuario/public_html/cotizador`).
- Sube todo el contenido de `frontend/dist/` al docroot.
  - Si usaste `npm run package:cpanel`, dentro de `release/cpanel-.../frontend/` encontrarás dos utilidades:
    - `config.js.template`: renómbralo a `config.js`, edita las URLs y súbelo al docroot.
    - `.htaccess`: archivo listo para SPA fallback en Apache/cPanel; súbelo al docroot si aún no existe.

1. Crear `/config.js` en el docroot (enganche al backend)

Contenido del archivo:

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

Nuestro `index.html` ya incluye `<script src="/config.js"></script>`. Para cambiar de backend en el futuro, solo edita este archivo (no requiere rebuild).

1. Habilitar SPA fallback con `.htaccess` (rutas profundas como /admin/config)

```
RewriteEngine On
RewriteBase /
RewriteCond %{REQUEST_FILENAME} !-f
RewriteCond %{REQUEST_FILENAME} !-d
RewriteRule . /index.html [L]
```

1. SSL del subdominio

- Activa AutoSSL/Let's Encrypt en cPanel para `cotizador.aysafi.com`.

1. Verificación end-to-end

- Abre `https://cotizador.aysafi.com/` y en la consola ejecuta `window.__APP_CONFIG__` (debe mostrar API_BASE/FRONTEND_URL correctos).
- En la UI, entra a "Config runtime" (/admin/config) para revisar `window.__APP_CONFIG__` y probar `/api/config`.
- Crea/edita registros para confirmar llamadas a `https://emqx.aysafi.com/*`.

C) VPS monolítico full stack con scripts (SSH)

Objetivo: desplegar backend y frontend juntos en un mismo VPS, sirviendo el frontend desde el propio Node/Express (sin Apache/cPanel). Se proveen dos scripts listos: uno en Node.js y otro en shell.

Qué resuelven los scripts:

- Empaquetan el repo local (excluyen `node_modules`, `.git`, releases y `outputs/`).
- Suben el paquete al VPS por `scp` y lo extraen en `deployPath` (p. ej. `/opt/cotizador`).
- Instalan dependencias (`npm ci`) y construyen el frontend (`frontend/build`).
- Escriben un `.env` remoto a partir de tu `deploy.vps.json`.
- Inician o reinician el proceso con `pm2` bajo el nombre `cotizador` y guardan el estado (`pm2 save`).

Requisitos previos:

- En tu equipo: `ssh`, `scp`, `tar`. En Windows 10/11, OpenSSH viene integrado; en PowerShell puedes usarlo.
- En el VPS: Node.js 20 y npm. El script instala `pm2` si no está.
- Usuario SSH con permisos para crear `deployPath` (p. ej. `/opt/cotizador`).
- (Opcional) Nginx en el VPS si quieres exponer en 443 con TLS y proxy al Node interno.

Archivos relevantes en este repo:

- `backend/scripts/deploy_vps_monolithic.js` (Node)
- `backend/scripts/deploy_vps_monolithic.sh` (Shell)
- `deploy.vps.example.json` (plantilla de configuración)

1) Crea tu configuración `deploy.vps.json`

Copia el ejemplo y edítalo:

```
{
  "host": "1.2.3.4",
  "port": 22,
  "username": "ubuntu",
  "privateKey": "C:/Users/tuusuario/.ssh/id_rsa",
  "deployPath": "/opt/cotizador",
  "backendPort": 5000,
  "domain": "emqx.tudominio.com",
  "frontendDomain": "cotizador.tudominio.com",
  "useNginx": false,
  "env": {
    "NODE_ENV": "production",
```

```
    "FRONTEND_URL": "https://cotizador.tudominio.com",
    "PUBLIC_API_BASE": "",
    "JWT_SECRET": "cambia-esto",
    "ADMIN_PASSWORD": "admin-pass",
    "MORGAN_FORMAT": "combined"

  }
}
```

Notas:

- En Windows, asegúrate de que `privateKey` apunte a tu clave privada (OpenSSH). También puedes omitirla si tu `ssh-agent`/config lo gestiona.
- `PUBLIC_API_BASE` vacío indica monolítico (frontend y backend mismo origen). Si decides poner Nginx delante en 443, puede seguir vacío.
- Para correos, agrega `SMTP_HOST`, `SMTP_PORT`, `SMTP_USER`, `SMTP_PASS` y opcional `SMTP_FROM` en `env`.
- Si quieres externalizar salidas, añade `OUTPUT_DIR` (p. ej. `/var/lib/cotizador/outputs`) y asegura permisos.

2) Ejecuta el despliegue

- Con Node (recomendado y multiplataforma): `npm run deploy:vps` (usa `deploy.vps.json`).
- Alternativa shell (Linux/macOS con `jq`): `bash backend/scripts/deploy_vps_monolithic.sh deploy.vps.json`.

Qué hace el script (resumen):

1. Crea un tarball temporal excluyendo directorios/patrones pesados.
2. Sube el tarball al VPS a `/tmp` y lo extrae en `deployPath`.
3. Ejecuta `npm ci` en la raíz, `npm ci && npm run build` en `frontend/`.
4. Genera el archivo `.env` remoto con los pares `clave=valor` de `env`.
5. Inicia o reinicia `pm2` para `backend/server.js` con nombre `cotizador` y hace `pm2 save`.

3) Verifica el servicio

- En el VPS: `pm2 status` debe mostrar `cotizador` online.
- Backend local: `curl http://localhost:<backendPort>/` (por defecto 5000) debe responder `{ ok: true }`.
- Desde Internet: si no configuras Nginx/TLS, el puerto 5000 no debe exponerse públicamente; usa Nginx para 443.

4) (Opcional) Publicar en 443 con Nginx y TLS

Puedes reutilizar la configuración de la sección B (Nginx reverse proxy con SSL y SSE). Cambia solo el `proxy_pass` al puerto interno que configuraste (5000 por defecto). Asegura:

- `proxy_buffering off` y timeouts altos para SSE en `/api/events`.
- Certificados válidos (Let's Encrypt) y redirección 80 → 443.

5) Re-deploys y rollback

- Para publicar cambios, vuelve a ejecutar el script; pm2 hará restart con la nueva versión.
- Si un despliegue falla a mitad de camino, vuelve a ejecutar. El script es idempotente en pasos claves.
- Mantén respaldos de tu `.env` y de `OUTPUT_DIR` si es externo a la carpeta del proyecto.

6) Problemas comunes

- "Node/npm no encontrados" en VPS: instala Node 20 (Nodesource o nvm) e inténtalo de nuevo.
- Permisos en `deployPath`: crea la carpeta con el usuario SSH o ajusta ownership (`chown`).
- `pm2: command not found`: el script intenta instalarlo globalmente con npm; revisa PATH si tu shell no lo ve.
- Frontend desactualizado: confirma que el script ejecuta `frontend:build` sin errores (revisa logs de la ejecución).
- `outputs/` sin permisos: si usas ruta externa en `OUTPUT_DIR`, crea la carpeta y ajusta permisos.

Variables de entorno (resumen y propósito)

- `PUBLIC_API_BASE`: Base pública del backend para el frontend.
  - Split: `https://emqx.aysafi.com`
  - Monolítico: vacío (mismo dominio)
- `FRONTEND_URL`: URL pública del frontend (correos/QR), ej. `https://cotizador.aysafi.com`.
- `JWT_SECRET`: firma del token admin; único y largo.
- `ADMIN_PASSWORD`: contraseña para obtener token admin (`POST /api/admin/login`).
- `SMTP_HOST`, `SMTP_PORT`, `SMTP_USER`, `SMTP_PASS`: correo saliente.
- `OUTPUT_DIR` (opcional, preferido): ruta de PDFs/QR/JSON/logos cuando no quieras usar `backend/outputs` interno.
- `OUTPUTS_DIR` (legacy): alias de `OUTPUT_DIR`.
- `MORGAN_FORMAT` (opcional): formato de logs HTTP.

Cómo "enganchar" frontend y backend en dominios distintos (versión simple)

1. Asegúrate de que el backend responda en `https://emqx.aysafi.com` y permita CORS.
2. En cPanel, crea/edita `/config.js` con `API_BASE = 'https://emqx.aysafi.com'` y `FRONTEND_URL = 'https://cotizador.aysafi.com'`.
3. Para cambios futuros de URL, edita solo `config.js` y recarga el navegador (no necesitas rebuild).
4. Verifica en `/admin/config` que la configuración runtime sea correcta.

¿Qué es /admin/config?

`/admin/config` es una página de diagnóstico del frontend que te ayuda a comprobar la configuración de ejecución (runtime) sin reconstruir la app.

Qué muestra y cómo usarla:

- Muestra el objeto `window.__APP_CONFIG__` cargado desde `/config.js` (por ejemplo, `API_BASE` y `FRONTEND_URL`).
- Intenta llamar a `GET /api/config` del backend para verificar lo que éste expone como configuración efectiva. Si definiste `ADMIN_PASSWORD`, esta ruta puede requerir un token admin (obtenible via `POST /api/admin/login`).
- Úsala después de editar `config.js` en cPanel o de cambiar variables de entorno del backend; te confirma si el frontend y el backend ven las URLs correctas.
- No cambia la config; solo la visualiza y enlaza a recursos clave para pruebas.

Troubleshooting: Mixed Content y config.js

- Error: “Mixed Content: The page was loaded over HTTPS, but requested an insecure endpoint http://…”.
  - Causa: `API_BASE` en `config.js` usa `http://` mientras la página está en `https://`.
  - Solución: usa un backend con HTTPS (reverse proxy Nginx con certificado) y apunta `API_BASE` a `https://tu-backend`. Evita puertos `:3000`/`:5000` en `http://` cuando la página es HTTPS.
  - Recuerda aplicar lo mismo para SSE (`/api/events`).
- Error: “config.js:1 Uncaught SyntaxError: Unexpected token 'export'”.
  - Causa: se subió por error un archivo con sintaxis de módulos (por ejemplo `export …`).
  - Solución: `config.js` debe ser un script clásico que solo asigne:

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

- Nota: el warning de Vite al construir “<script src="/config.js"> can’t be bundled without type="module"” es esperado. Ese archivo se carga en runtime para evitar rebuilds.

Verificación automatizada post-despliegue (recomendado)

Después de desplegar o cambiar configuración DNS/SSL, ejecuta las suites automatizadas desde tu máquina de desarrollo:

1. Prueba de frontend y backend en producción (puedes ajustar dominios si no usas los predeterminados):

```
$env:FRONTEND_URL_PROD="https://cotizador.aysafi.com";
$env:BACKEND_URL_PROD="https://emqx.aysafi.com:8443"; npm run
test:security:prod
```

1. Interpreta los resultados:

- “Backend no alcanzable … :8443”: el puerto 8443 no está expuesto o el servicio no está arriba. Si usas Nginx en 443 como proxy, puedes cambiar `BACKEND_URL_PROD` a `https://emqx.aysafi.com` (sin puerto) y reejecutar.
- “/admin/login responde 404”: falta SPA fallback en el hosting del frontend. Sube `.htaccess` con la regla de reescritura a `index.html`.
- “Frontend sin Content-Security-Policy”: añade la plantilla `.htaccess` incluida en el paquete cPanel para CSP y otros headers de seguridad.
- “Posible referencia http:// en asset”: inspecciona el asset, refuerza CSP y evita librerías que inyecten contenido inseguro.

1. Si ocultas 8443 detrás de 443

Cuando tu backend está detrás de Nginx en 443, ejecuta:

```
$env:BACKEND_URL_PROD="https://emqx.aysafi.com"; npm run test:security:prod
```

Mantén en el frontend `API_BASE` apuntando a `https://emqx.aysafi.com` (sin puerto) y confirma los timeouts/SSE en Nginx.