Cotizador (scaffold)

Proyecto scaffold para un cotizador con frontend en React (Vite) y backend en Node.js (Express). Persistencia en filesystem (`outputs/`).

Novedades:

- Configuración de runtime del frontend vía `/config.js` (no requiere rebuild para cambiar URLs base)
- Página de diagnóstico protegida: `/admin/config` para ver la configuración efectiva
- Script de empaquetado cPanel: `npm run package:cpanel`

Guías útiles:

- Flujo de ramas: ver `BRANCH_FLOW_QUICKSTART.md` para una guía rápida de trabajo con main/develop/feature/hotfix).
- Manual de uso por roles: ver `USAGE.md` (Editor, Desarrollador y Mantenedor).

Instalación y ejecución (backend)

1. Instala dependencias en la raíz

```
npm install
```

2. Instala dependencias del frontend y arranca en modo desarrollo

```
cd frontend
npm install
npm run dev
```

3. Arranca el backend (desde la raíz)

```
npm run backend
```

4. Desarrollo todo-en-uno (frontend + backend con recarga y morgan)

```
npm run dev
```

5. Variables de entorno (crear `.env` basado en `.env.example`)

```
PORT=5000
OUTPUTS_DIR=outputs
SMTP_HOST=
SMTP_PORT=
SMTP_USER=
SMTP_PASS=
MORGAN_FORMAT=dev

# Frontend/URLs públicas
FRONTEND_URL=https://cotizador.tudominio.cl
# Si el backend expone la API en otra URL/origen, define PUBLIC_API_BASE.
# Déjalo vacío si frontend y backend comparten dominio.
PUBLIC_API_BASE=
```

6. Notas de despliegue y configuración en runtime

- El backend sirve `/config.js` con `window.__APP_CONFIG__` (API_BASE y FRONTEND_URL) a partir de variables de entorno.
- El frontend carga `/config.js` antes de inicializar y todas las llamadas usan `apiUrl()`/`eventsUrl()`; al cambiar `PUBLIC_API_BASE` o `FRONTEND_URL` solo necesitas reiniciar la app, no reconstruir.
- Diagnóstico: inicia sesión admin y abre `/admin/config` para ver lo que ve el frontend y lo que expone el backend en `/api/config`.

7. Lint antes de construir

Para asegurar calidad, ejecuta lint antes del build:

```
npm run lint ; npm run frontend:build
```

Puedes integrar un script combinado si lo prefieres:

```
{
  "scripts": {
    "build:checked": "npm run lint && npm run frontend:build"
  }
}
```

8. Pruebas locales y en la nube

Dispones de tres suites de pruebas:

- Backend local (unit/smoke):

```
npm run test:backend
```

- Frontend (Vitest):

```
npm run test:frontend
```

- Seguridad HTTPS local (recorre rutas y verifica mixed content):

```
npm run test:security
```

- Seguridad E2E en producción (frontend y backend públicos):

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

Notas e interpretación:

- Si el backend 8443 no está alcanzable, las pruebas de backend se marcarán como SKIP y no fallarán la suite; revisa Nginx/firewall/servicio.
- Si `/admin/login` devuelve 404 en el frontend productivo, es indicio de que falta el fallback SPA del servidor estático; corrige `.htaccess`.
- Si no existe `/config.js` en el frontend, es válido (config embebida). El test sólo lanza advertencia.
- Si hay advertencia de referencias `http://` en assets, revisa el bundle o CSP. En producción se recomienda CSP para mitigar inyecciones.