Anexo Técnico – Optimización Frontend React (Flujo GET)

Elaborado por: High Cloud Tec – Oscar Iván Ocampo

Cliente: Coris Argentina

Fecha: 4 de noviembre de 2025

Documento confidencial

1. Contexto

Durante las pruebas del flujo de cotización (HAR), el tiempo de respuesta del GET Comparador alcanzó 13.8 segundos, lo que representa el 68% del tiempo total de la cotización. Este valor está fuera del SLA objetivo (≤8s) y muestra que el cuello de botella principal está en la capa frontend / renderizado visual, no en el backend.

El análisis de recursos indica un alto peso en archivos bundle.js (>1.5 MB), múltiples hojas CSS y JS sin minificar, carga sincrónica de imágenes grandes y scripts externos de tracking y chat.

2. Diagnóstico técnico del flujo GET

Causa probable	Impacto	Solución técnica recomendada
Componentes React con renderizado completo del comparador	Aumenta el TTI y bloquea la interactividad	Implementar lazy loading de secciones no visibles (React.lazy / Suspense)
Bundle único sin división de módulos	Aumenta el tiempo de descarga inicial	Aplicar code splitting con Webpack o Vite (splitChunks, dynamic import)
Carga de imágenes sin optimización	Eleva LCP y ralentiza el render	Comprimir imágenes a WebP/AVIF y usar loading='lazy'
Scripts externos bloqueantes (Chatbot, Tag Manager, Analytics)	Retrasa el render principal	Agregar defer o async en scripts no esenciales
Sin caching ni headers eficientes	Repetición de descargas en cada request	Añadir Cache-Control (3600s) y validación ETag

Sin CDN activo para estáticos	Aumenta latencia global	Implementar CDN (CloudFront / Cloudflare) con reglas para /static/, /img/, /js/

3. Acciones prioritarias para el desarrollador

Acción	Descripción	Prioridad
Optimizar carga de componentes React	Aplicar lazy loading, React.memo y Suspense en secciones del comparador.	Alta
Reducir y minificar bundle	Dividir el paquete principal JS y CSS, y minificar en build.	Alta
Comprimir imágenes y banners	Implementar pipeline de compresión (WebP/AVIF) en servidor o CI/CD.	Media
Revisar scripts externos	Posponer carga de tracking/chatbot hasta que el DOM esté listo.	Media
Implementar CDN y cache estático	Servir JS/CSS desde CloudFront con cache dinámico y headers optimizados.	Media

4. Métricas objetivo tras optimización

Métrica	Estado actual	Meta post-optimización
GET Comparador	13.8 s	≤ 6.0 s
LCP (Largest Contentful Paint)	6.8 s	≤ 3.0 s
Speed Index	12.2 s	≤ 6.0 s
Peso total página	3.2 MB	≤ 1.5 MB

5. Validación sugerida

Una vez aplicadas las correcciones:
1. Ejecutar nueva auditoría con Lighthouse (modo Performance).
2. Validar tiempos en Chrome DevTools → Network (GET).

3. Repetir test HAR para registrar evidencia técnica.
4. Entregar resultado comparativo frente a este informe (Coris v4).