GUÍA NODESTUDIO

Checklist para que tu app no te funda la tarjeta

Qué revisar antes de publicar una app vibe codeada para no llevarte sorpresas en la factura.

✓ 15 puntos + 1 prompt para Claude Code

1 Checklist antes de publicar

Marcá cada punto. Si alguno queda sin tildar, ahí hay una posible bomba de costos.

CÓDIGO E IA

- [ ] **Llamadas innecesarias a APIs**
  Nada que llame a una API en cada render, tecla o scroll. Debounce, sin llamadas duplicadas, sin polling cada pocos segundos.
- [ ] **Loops que consumen IA sin límite**
  Agentes, retries y cadenas de prompts con tope de iteraciones, `max_tokens` y timeout.
- [ ] **Rate limits**
  Límite por IP y por endpoint, sobre todo en los que llaman a IA, envían emails o SMS.
- [ ] **Límites por usuario**
  Cuota diaria/mensual de requests y tokens por cuenta. Plan gratis con tope duro.
- [ ] **Caching**
  Cachear respuestas repetidas y datos que casi no cambian. Usar prompt caching si el proveedor lo ofrece.
- [ ] **Jobs, cron y workers infinitos**
  Frecuencia razonable, condición de salida, reintentos con backoff y máximo. Nada que se re-encole solo.
- [ ] **Serverless functions**
  Timeout y concurrencia máxima configurados. Ninguna función que se dispare a sí misma en cadena.
- [ ] **Logs excesivos**
  Nada de loguear payloads, imágenes ni respuestas completas de IA. Nivel `info` en producción y retención corta.

1 Checklist antes de publicar (cont.)

INFRA Y CUENTAS

- [ ] **Costos de base de datos**
  Sin queries N+1, con índices y paginación. Ojo con listeners en tiempo real y planes que cobran por lectura.
- [ ] **Bandwidth**
  Imágenes comprimidas, CDN, paginación. Sin descargar archivos pesados en cada visita.
- [ ] **Almacenamiento**
  Límite de tamaño en uploads, borrar temporales, reglas de expiración.
- [ ] **Servicios de terceros**
  Lista de cada API paga con su precio por unidad. API keys solo en el servidor, nunca en el frontend.
- [ ] **Alertas de gasto**
  Budget alerts activas en el cloud, el hosting y cada proveedor de IA (al 50%, 80% y 100%).
- [ ] **Cuotas y límites de uso**
  Spend limit duro en los paneles de los proveedores. Autoscaling con máximo.

ESTIMACIÓN DE COSTOS

Mensual	100 usuarios	1.000	10.000	100.000
IA / tokens				
Base de datos				
Hosting / serverless				
Storage + bandwidth				
Terceros				
Total				

```
costo = usuarios × acciones/día × 30 × costo por acción
```

2 Que Claude Code lo audite por vos

Abrí Claude Code en la carpeta del proyecto y pegá esto:

PROMPT PARA PEGAR EN CLAUDE CODE

```
Actuá como auditor de costos. Revisá TODO este proyecto (código, configs, infra, cron,
workflows, dependencias y variables de entorno de ejemplo) y buscá "bombas de costos":
cualquier cosa que pueda generar gasto inesperado cuando haya usuarios reales.

Revisá específicamente:
1. Llamadas a APIs innecesarias o duplicadas (renders, useEffect, polling, sin debounce).
2. Loops, agentes o reintentos que llamen a IA sin tope de iteraciones, max_tokens o
timeout.
3. Endpoints sin rate limit y sin límite de uso por usuario.
4. Falta de caching en respuestas repetidas o datos que casi no cambian.
5. Jobs, cron, workers o colas que puedan correr infinitamente o re-encolarse solos.
6. Consultas a base de datos ineficientes (N+1, sin índices, sin paginación, listeners en
tiempo real innecesarios).
7. Bandwidth y almacenamiento (uploads sin límite, imágenes sin optimizar, archivos que
nunca se borran).
8. Serverless functions sin timeout, con concurrencia ilimitada o que se invocan a sí
mismas.
9. Logs excesivos o que guardan payloads y respuestas completas.
10. Servicios de terceros pagos y API keys expuestas en el frontend.
11. Alertas de gasto, cuotas y límites duros que falten configurar.

Después:
- Listá cada hallazgo con archivo y línea, por qué puede costar plata, severidad (crítica /
alta / media / baja) y el peor escenario.
- Armá una tabla de costo mensual estimado con 100, 1.000, 10.000 y 100.000 usuarios, con
los supuestos que usaste (acciones por usuario, tokens por request, precios). Si no sabés
un precio, marcalo como supuesto; no lo inventes.
- Proponé una optimización concreta para cada hallazgo.
- Aplicá solo las optimizaciones seguras, las que no cambian lo que ve el usuario: topes,
timeouts, caching, debounce y rate limits con valores razonables. Para cualquier cambio que
afecte funcionalidad, datos, precios o infraestructura, pedime confirmación antes.
- No borres datos ni toques configuración de producción.
- Al final, resumí qué cambiaste y qué me queda configurar a mano (por ejemplo, alertas de
gasto en el panel de cada proveedor).
```



NodeStudio

Automatizamos procesos de negocio con IA y sistemas a medida.

nodestudioai.com