Prepara tu app para escalar · 3 cosas que le dices a Claude antes de lanzar (+ load testing)

CATEGORÍA · CÓDIGO Y DESARROLLO

Tu app funciona con 10 usuarios, se arrastra con 100 y se cae con mil: eso es el scaling cliff, y las apps hechas con IA le pegan durísimo porque el código está optimizado para que funcione, no para que funcione a escala (esa query que con 100 filas tarda 50 ms, con cien mil se va a 30 s).

QUÉ VAS A ENCONTRAR

FICHA DEL RECURSO

Qué es, para qué sirve y cómo se implementa

FUNCIÓN

Tu app funciona con 10 usuarios, se arrastra con 100 y se cae con mil: eso es el scaling cliff, y las apps hechas con IA le pegan durísimo porque el código está optimizado para que funcione, no para que funcione a escala (esa query que con 100 filas tarda 50 ms, con cien mil se va a 30 s).

PARA QUÉ SIRVE

Tu app funciona con 10 usuarios, se arrastra con 100 y se cae con mil: eso es el scaling cliff, y las apps hechas con IA le pegan durísimo porque el código está optimizado para que funcione, no para que funcione a escala (esa query que con 100 filas tarda 50 ms, con cien mil se va a 30 s). Esta te da los prompts exactos —para Claude Code o Codex— de las 3 cosas que arreglan eso en tu propio código sin que toques una línea: indexing, caching y async. Más el paso que casi nadie hace, el load testing: simular mil usuarios y encontrar los cliffs antes que tus usuarios. Cierra con un checklist de qué revisar antes de lanzar y FAQ.

DÓNDE APLICARLO

Nivel Intermedio. Temas: escalar, scaling-cliff.

CÓMO IMPLEMENTARLO

1. Confirma requisitos: plan de Claude, herramientas y accesos que la tarea necesita.
2. Prepara el material (archivos, cuentas, datos) antes de empezar.
3. Ejecuta el método sobre un caso real pequeño, no sobre el proyecto completo.
4. Verifica el resultado contra lo que esperabas antes de repetirlo o automatizarlo.
5. Cuando funcione dos veces seguidas, conviértelo en rutina o skill.

EL RECURSO EN DETALLE

Qué resuelve y cómo funciona

Tu app funciona con 10 usuarios, se arrastra con 100 y se cae con mil: eso es el scaling cliff, y las apps hechas con IA le pegan durísimo porque el código está optimizado para que funcione, no para que funcione a escala (esa query que con 100 filas tarda 50 ms, con cien mil se va a 30 s).

Esta te da los prompts exactos —para Claude Code o Codex— de las 3 cosas que arreglan eso en tu propio código sin que toques una línea: indexing, caching y async. Más el paso que casi nadie hace, el load testing: simular mil usuarios y encontrar los cliffs antes que tus usuarios. Cierra con un checklist de qué revisar antes de lanzar y FAQ.

ANTES DE DARLO POR HECHO

Comprobación y errores frecuentes

1. Aplica el método sobre un caso pequeño y compara el resultado con lo que esperabas.
2. Si el resultado no sale, revisa primero los requisitos: plan, versión del modelo y accesos.
3. Anota lo que tuviste que corregir a mano: eso es lo que hay que añadir a tus instrucciones permanentes.
4. Cuando funcione dos veces seguidas, automatízalo como rutina o skill.

DATOS DEL RECURSO

Tipo: Guía paso a paso
Categoría: Código y desarrollo
Nivel: Intermedio
Temas: escalar, scaling-cliff

El ecosistema de IA cambia rápido: confirma en el repositorio que el proyecto sigue activo y que los comandos no han cambiado desde la fecha de esta ficha.

SIGUIENTE PASO

Instálalo hoy, en pequeño.

Elige la ruta 1, pégale el enlace a Claude Code y pruébalo en un caso real chico. Un recurso instalado y probado vale más que diez guardados para después.

MÁS GUÍAS GRATIS EN 377CREATIVE.COM/RECURSOS