02

GUÍA PRÁCTICA

aGo lab

Versión 1.0

Las 5 fases de la deuda técnica empresarial

Diagnóstico para gerentes, CTOs y dueños de PyME. Síntomas verificables, decisión por fase y costo de no actuar.

AUTOR
Sixto Valdés

FECHA
Mayo 2026

WEB
ago.cl

§ EN 30 SEGUNDOS

En qué fase está tu empresa.

La deuda técnica no es binaria. Es un continuo en cinco fases, y la decisión correcta cambia en cada una. Tratar fase 2 como fase 5 lleva a reescribir cuando bastaba refactorizar. Tratar fase 4 como fase 2 lleva a parchar lo que ya no admite parche.

Este pliego es para diagnosticar antes de invertir. Cinco fases con síntomas verificables, decisión correcta por fase y costo estimado de no actuar.

El framework se apoya en Cunningham (1992) y en datos de Stripe y Harris Poll (2018) y Dalal et al. (2020).

§ TRES DATOS

33 %

tiempo dev gastado en deuda técnica (Stripe, 2018)

20-40 %

del valor de assets TI es deuda (McKinsey, 2020)

1992

año en que Ward Cunningham acuñó el término

§ CÓMO USAR ESTA GUÍA

Cinco minutos: lee el mapa de fases en la página 3 y reconoce los síntomas de tu empresa en la página 4.

Treinta minutos: lee todo en orden. Llévalo a tu próxima conversación con tu equipo de TI o tu directorio.

Para llevárselo a tu jefe o socios: imprime las páginas 3, 4 y 5. Son las que sostienen la conversación.

§ DIAGNÓSTICO

Las cinco fases.

El deterioro es secuencial. Una empresa rara vez salta fases hacia abajo. Pero también rara vez sube de fase sin decisión deliberada de inversión.

01

Sano

Stack moderno, tests, deploys automáticos. Estimaciones se cumplen.

02

Tensión silenciosa

Parches localizados. Estimaciones se desvían 50-100 %.

03

Dependencia operativa

Una o dos personas son irreemplazables.

04

Frágil

Cada cambio rompe algo. Deploys los viernes.

05

Crítico

Equipo original se fue. Reconstruir es la única salida.

Tiempo típico entre fases sin disciplina: de 1 a 2: 12-24 meses. De 2 a 3: 6-12 meses. De 3 a 4: 3-6 meses. De 4 a 5: puede ser semanas tras un evento gatillante.

> *"La deuda técnica es un problema de cadencia, no de código."*

§ RECONOCIMIENTO

Lo que se dice vs. lo que pasa.

Cada fase tiene una versión oficial (lo que dice el equipo en una reunión) y una versión real (lo que ocurre cuando se mira con honestidad). Si reconoces más de dos filas, ahí está tu fase.

	LO QUE SE DICE	LO QUE ESTÁ PASANDO
01	**Sano.** Onboarding de dev nuevo en 2 semanas. Cambios estimados se cumplen ±20 %.	Riesgo real: caer a fase 2 en 12-24 meses sin disciplina de mantenimiento.
02	**Tensión.** «Tenemos un par de cosas que mejorar.»	El equipo dice «déjame ver» antes de prometer. Dependencias rezagadas que nadie actualiza.
03	**Dependencia.** «Tenemos un equipo sólido.»	Las vacaciones del dev senior se vuelven críticas. Hay un Excel maestro que nadie quiere tocar.
04	**Frágil.** «Todo lo importante está en producción.»	Hay miedo real de tocar ciertos módulos. Código fuente no coincide del todo con producción.
05	**Crítico.** «Estamos evaluando opciones.»	Consultores miran, dicen «esto hay que migrar», y se van porque no quieren tocarlo.

§ ERROR COMÚN

Muchas empresas en fase 2 contratan más desarrolladores creyendo que el problema es de capacidad. No lo es.

Sumar gente a un sistema con deuda no resuelta acelera el deterioro: cada nuevo dev agrega su forma de parchar. Brooks lo dijo en 1975 y sigue siendo cierto.

§ DECISIÓN

Qué corresponde hacer en cada fase.

La acción correcta depende de la fase. La misma decisión aplicada en la fase equivocada es contraproducente.

01 **Fase 1, Sano**

Mantener disciplina. 10-15 % del tiempo en deuda preventiva. Documentación viva. Actualizar dependencias trimestralmente.

02 **Fase 2, Tensión**

Bloquear 20 % del esfuerzo en deuda. Priorizar áreas problemáticas identificadas. No contratar más sin antes documentar.

03 **Fase 3, Dependencia**

Auditoría externa. Distribuir conocimiento por pair programming. Empezar plan modular de modernización.

04 **Fase 4, Frágil**

Migración por módulos con sistema viejo en paralelo. Congelar features no críticas. Aceptar 6-12 meses sin ROI directo.

05 **Fase 5, Crítico**

Reconstrucción planificada. 12-24 meses. Decisión de continuidad del negocio, no técnica.

§ COSTO DE NO ACTUAR

En fase 1-2: sumar 12-24 meses al deterioro. Más caro recuperar después.
En fase 3: perder al dev clave (renuncia o licencia) equivale a colapso operativo.
En fase 4-5: cada mes adicional duplica el costo de salida y el riesgo de continuidad.

§ ERRORES CAROS

Tres errores que cuestan años.

§ LOS TRES MÁS COMUNES

01 Tratar fase 2 como fase 5. Reescribir cuando bastaba refactorizar. Dos años perdidos.

02 Tratar fase 4 como fase 2. Parchar lo que ya no admite parche.

03 Contratar en fase 3 sin documentar primero. Acelera la llegada a fase 4.

§ CUÁNDO PEDIR AYUDA

a No sabes en qué fase está tu empresa.

b Equipo interno minimiza lo que tú sospechas.

c Te piden presupuesto para reescribir todo.

d Un dev clave anunció que se va.

e Un incidente reciente expuso fragilidad.

§ CONVERSACIÓN TIPO CON TU EQUIPO DE TI

Tres preguntas que te dicen la fase sin jerga:

1. «Si necesitamos publicar este cambio mañana, ¿lo podemos hacer con confianza?» La respuesta y el lenguaje corporal dicen más que el plan.

2. «Si renunciara hoy [el dev clave], ¿cuánto tardaría alguien más en sostener esto?» Si la respuesta es «varias semanas», estás en fase 3 o peor.

3. «¿Cuándo fue el último deploy un viernes y por qué?» Si responden «nunca lo hacemos los viernes», fase 4 confirmada.

§ PARA PROFUNDIZAR

Recursos y casos.

§ LECTURA RECOMENDADA

Martin Fowler, Technical Debt Quadrant

Stripe, The Developer Coefficient (2018)

McKinsey, Tech Debt: Reclaiming Tech Equity (2020)

§ CASOS aGo PÚBLICOS

Bioaudita: arquitectura modular, mantenimiento en fase 1 sostenida

Sign DataNubi: aislamiento arquitectónico de módulos críticos

TCultura: cadencia de mantenimiento sobre stack en producción

§ VEINTE MINUTOS SIN COMPROMISO

¿Quieres una segunda opinión sobre la fase en la que está tu empresa?

No vendemos reescrituras. Diagnosticamos primero, recomendamos según fase, y decidimos contigo el camino que cuesta menos.

hola@ago.cl • +56 9 7744 7331 • ago.cl

§ BIBLIOGRAFÍA

Cunningham, W. (1992). The WyCash Portfolio Management System. *Addendum to the Proceedings on Object-Oriented Programming Systems, Languages, and Applications*, 29-30. https://doi.org/10.1145/157709.157715

Dalal, V., Krishnakanthan, K., Münstermann, B., & Patenge, R. (2020). *Tech debt: Reclaiming tech equity* (inf. téc.). McKinsey Digital. https://www.mckinsey.com/capabilities/mckinsey-digital/our-insights/tech-debt-reclaiming-tech-equity

Stripe & Harris Poll. (2018). *The Developer Coefficient: Software engineering efficiency and its $3 trillion impact on global GDP* (inf. téc.). Stripe. https://stripe.com/files/reports/the-developer-coefficient.pdf

§ aGo lab

Sistemas que cumplen desde la arquitectura.

Construimos software a medida con cumplimiento Ley 21.719, GDPR y ARCO+ integrado desde el modelo de datos, no como capa agregada después. Operamos desde Chillán, Concepción y Santiago para clientes en Chile, Latinoamérica y proyectos internacionales.

WEB
ago.cl

LINKEDIN
linkedin.com/company/ago-lab

CORREO
hola@ago.cl

INSTAGRAM
@agolab.cc

WHATSAPP
+56 9 7744 7331

GITHUB
github.com/sixtovaldese

§ VEINTE MINUTOS, SIN COMPROMISO

Si tu equipo está evaluando construir o reformar un sistema, conversemos.

Lleva este documento a tu próxima reunión de TI o legal y nos contactas si quieres profundizar.

Versión 1.0 • Mayo 2026 • Compilado en LaTeX