Lo que la universidad no te enseña sobre el mundo real en Tecnologías de la Información

Edición 2026

Christian Elías Cruz González | 25 de febrero 2026

Lo que la universidad no te enseña (pero vas a necesitar)

Comunicación técnica clara

- Explicar problemas sin sonar arrogante
- Escribir tickets bien definidos
- Documentar decisiones
- Saber decir “no” con argumentos
- Si no sabes explicar tu solución, para la empresa no existe.

Trabajar en equipo multidisciplinario

En la universidad casi todo es individual.

En la industria:

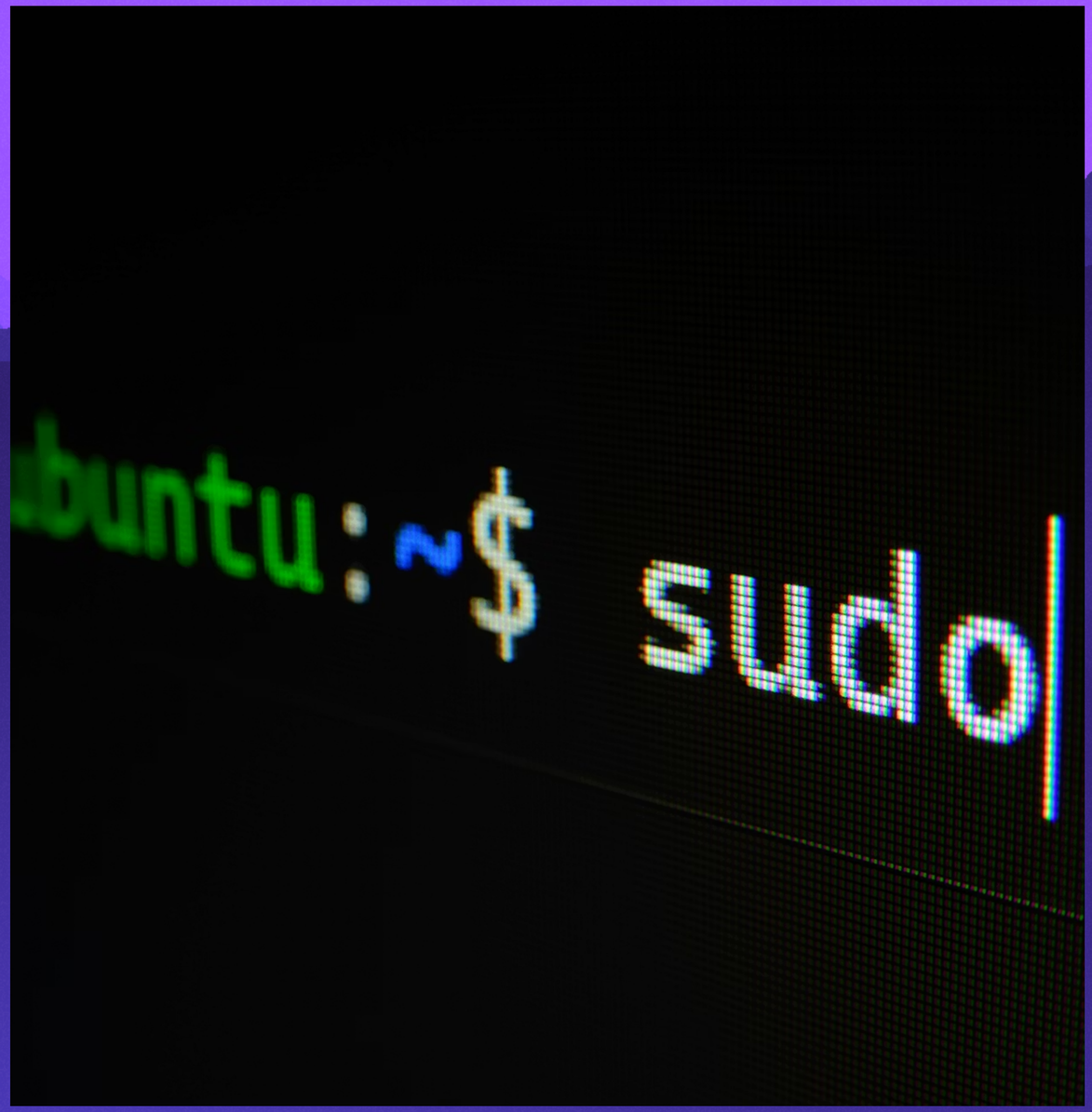
- Hay QA
- Hay negocio
- Hay infraestructura
- Hay seguridad
- Hay producto

Manejo de ambigüedad

- “Es como lo que hace X empresa, pero mejor”
- “Eso ya lo hacía el sistema anterior”
- “Eso es darle clic a un botón”
- “Es solo agregar una columna”
- “Me salió un error raro”

Saber usar la terminal

- Automatización
- Conexión a servidores
- Administración remota
- Scripts
- Debugging real
- Contenedores
- Cloud



Saber usar un editor de código

- Redes → archivos de configuración
- Infraestructura → YAML, Terraform
- Bases de datos → scripts SQL versionados
- Seguridad → scripts de análisis
- DevOps → pipelines CI/CD
- IoT → firmware

No todos van a ser desarrolladores. Pero todos van a escribir código en algún momento.

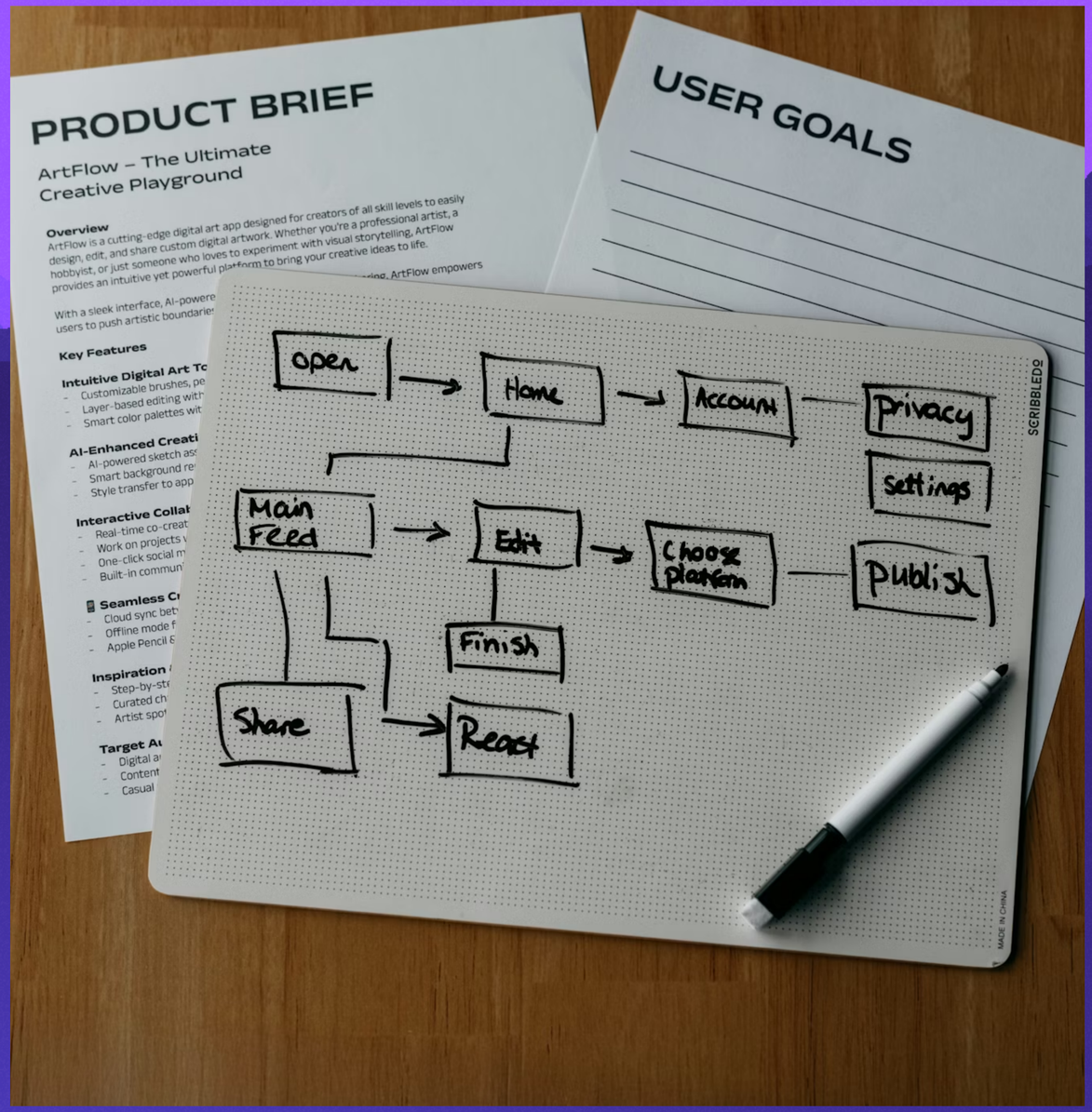
Git

Si no sabes Git, no estás listo para trabajar en equipo.

- Trabajo colaborativo
- Historial de cambios
- Revisión de código
- Trazabilidad
- Recuperación ante errores
- Control de versiones en infraestructura y script

Markdown

- Documentación vive en repositorios
- Los README son críticos
- Las decisiones arquitectónicas se escriben
- Las APIs se documentan



Escribir bien en inglés técnico

- Leer documentación
- Escribir tickets
- Entender foros
- Comunicarte en equipos globales

Saber usar un editor de código

- Redes → archivos de configuración
- Infraestructura → YAML, Terraform
- Bases de datos → scripts SQL versionados
- Seguridad → scripts de análisis
- DevOps → pipelines CI/CD
- IoT → firmware

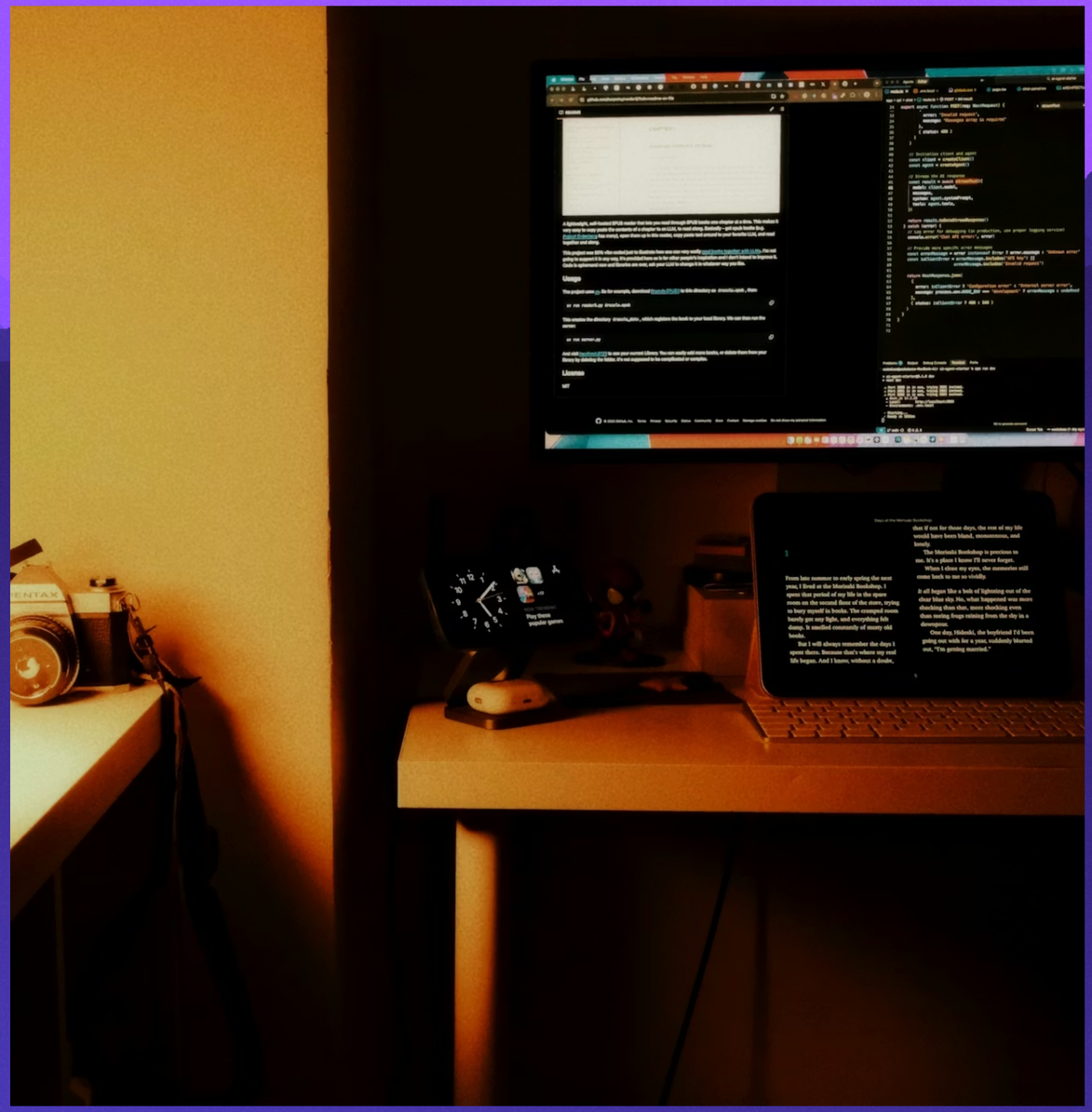
No todos van a ser desarrolladores. Pero todos van a escribir código en algún momento.

Debugging serio

- Leer logs
- Interpretar stack traces
- Reproducir errores
- Aislar problemas
- Formular hipótesis

Leer código ajeno

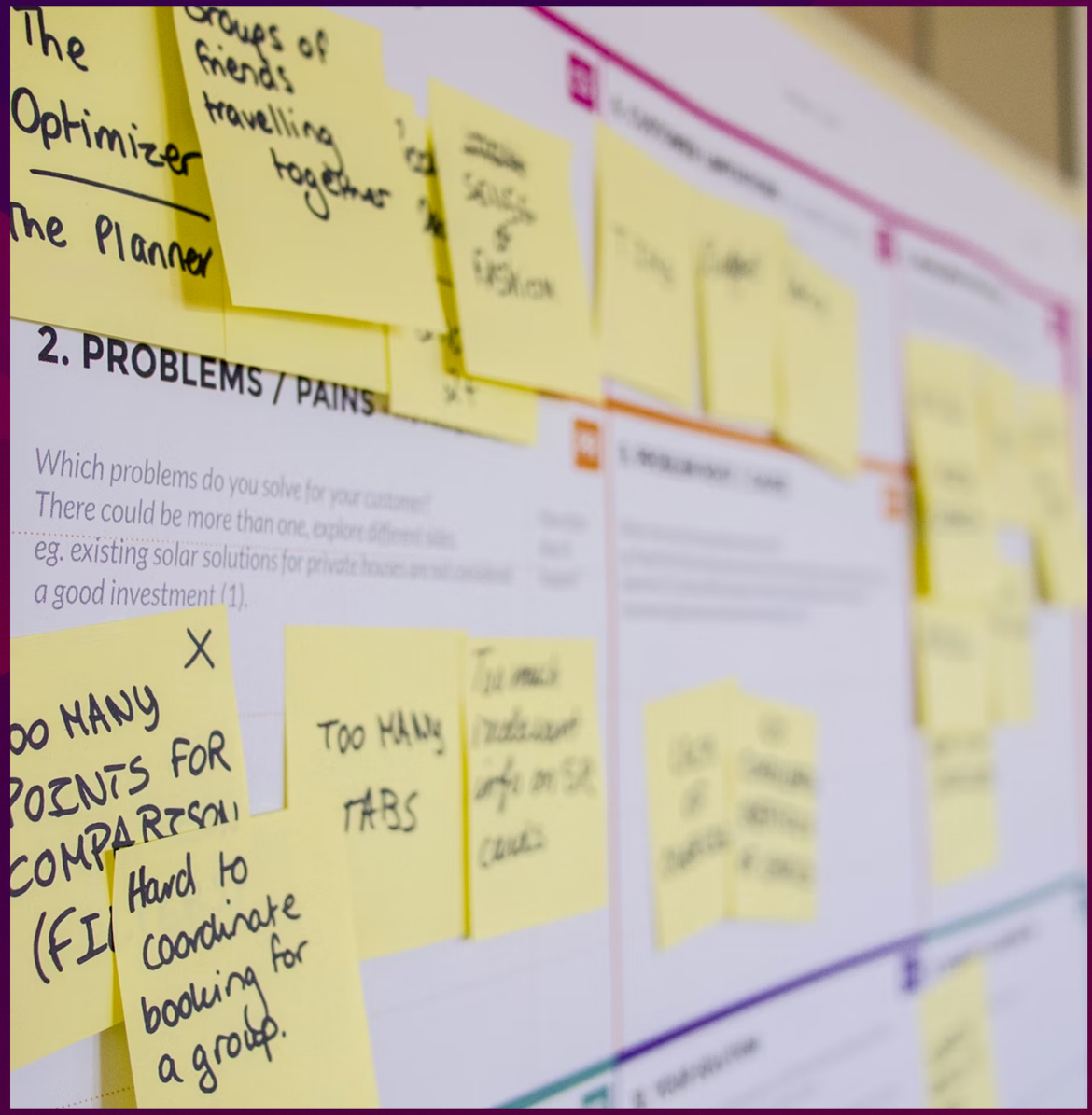
- El 80% del tiempo lees código que no escribiste.
- Muchas veces es legacy.
- Muchas veces está mal documentado.



Cómo pensar como Ingeniero y no como ejecutor técnico

Resolver problemas, no completar tickets

- En la universidad te dicen exactamente qué hacer.
- En la industria te dicen el problema y a veces ni siquiera está bien definido.
- Mentalidad de ingeniero: Pregunta el contexto, entiende restricciones, evalúa impacto y propone alternativas.



Sostenibilidad

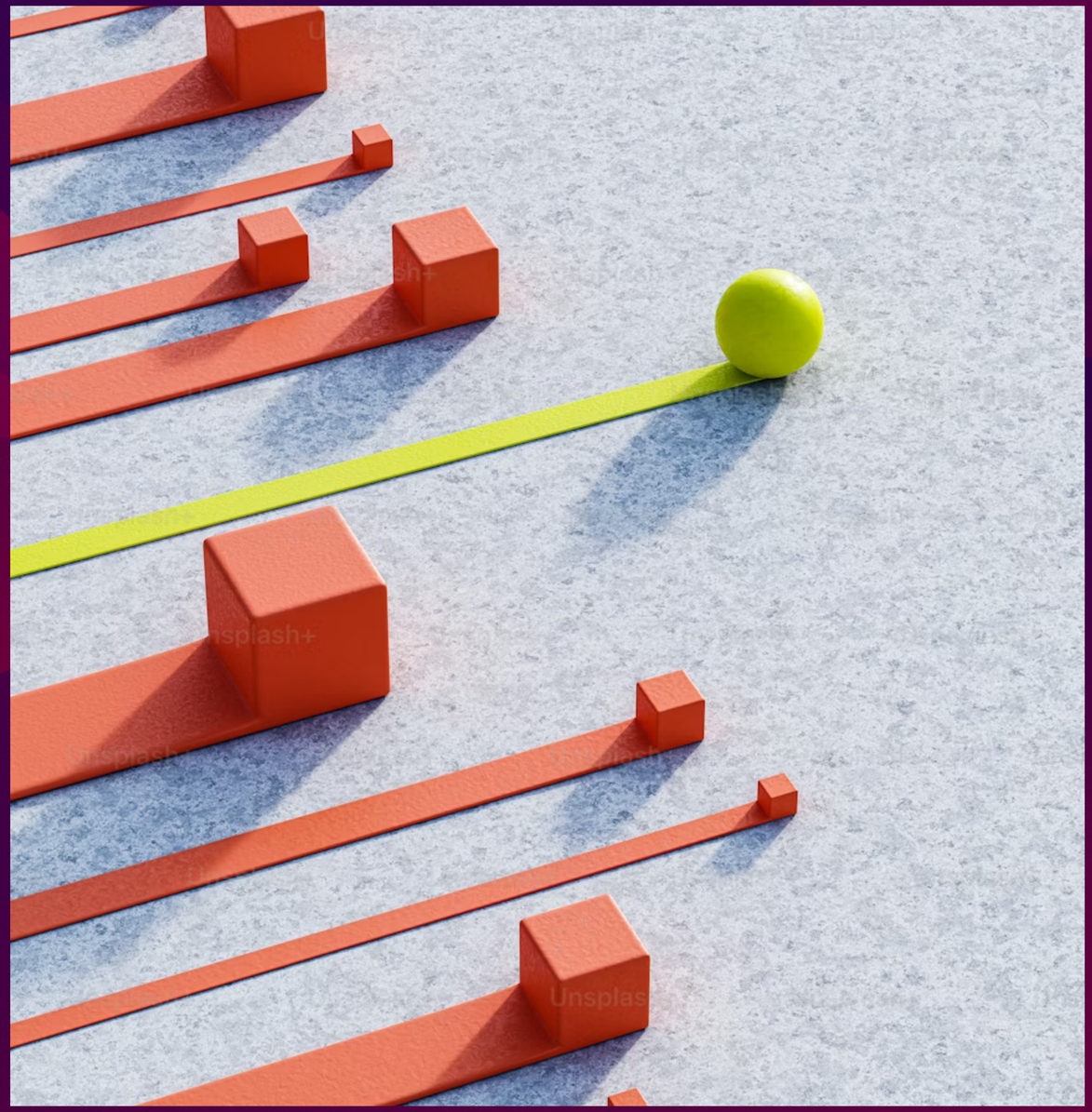
- Código entendible
- Scripts claros
- Infraestructura documentada
- Configuraciones reproducibles
- Nada “mágico”
- Lo difícil no es hacer que funcione. Lo difícil es que siga funcionando dentro de dos años.

Escalabilidad

- ¿Qué pasa si hay 10x usuarios?
- ¿Qué pasa si la empresa duplica sucursales?
- ¿Qué pasa si los datos crecen exponencialmente?
- Diseñar para crecer es más barato que rediseñar cuando ya explotó.

Impacto

- Cada hora de caída cuesta dinero.
- Cada error en producción afecta reputación.
- Cada mala decisión técnica genera deuda técnica.
- Cada optimización puede ahorrar miles.
- No trabajamos con código, redes o servidores. Trabajamos con impacto.



Tomar decisiones

- No existe solución perfecta.
- Toda decisión tiene costo.
- Hay que elegir conscientemente.
- Ingeniería es el arte de elegir bien bajo restricciones.
- El mercado no necesita más personas que sepan usar herramientas. Necesita personas que sepan tomar decisiones técnicas con criterio.

Cómo trabajar para empresas internacionales desde casa

Tu competencia ya no es local

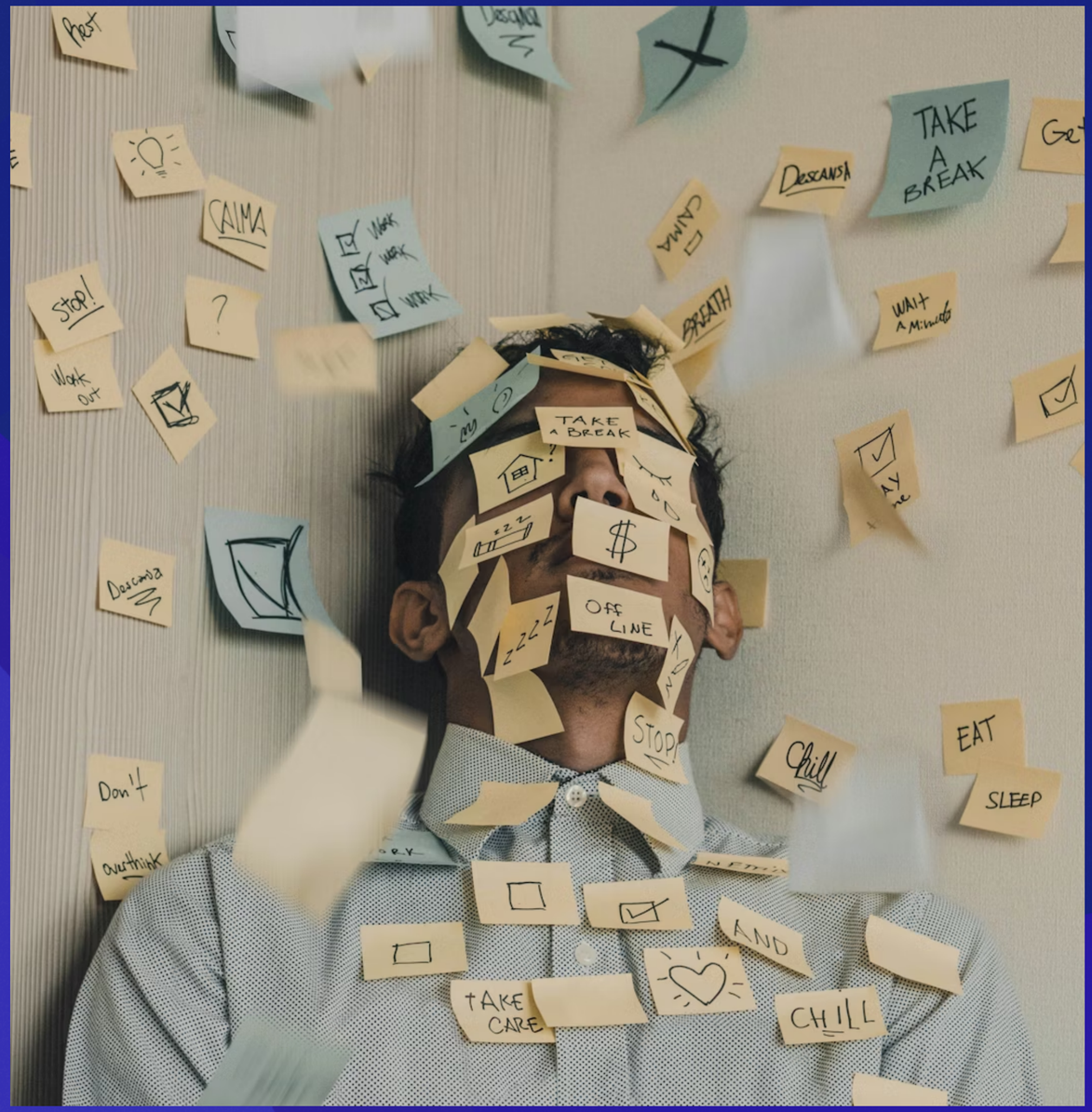
- Compites con gente de otros países.
- Hay diferencias culturales.
- Hay zonas horarias distintas.
- Hay estándares más altos.

Comunicación escrita impecable

- Escribir tickets claros.
- Explicar decisiones.
- Justificar cambios.
- Documentar problemas.
- En remoto, lo que escribes es tu reputación.

Gestión del tiempo y disciplina

- No procrastinar.
- Saber priorizar.
- Manejar zonas horarias.
- Separar vida personal y profesional.



Entender estándares internacionales

- Code reviews estrictos.
- Procesos formales.
- Seguridad más rigurosa.
- Auditorías.
- Cultura de feedback.

Profesionalismo digital

- LinkedIn profesional.
- GitHub con proyectos reales.
- CV claro.
- Portafolio técnico.
- Presencia coherente.

El mundo real no necesita técnicos.
Necesita Ingenieros.