

Refactoriza Tu Escritura

Principios de ingeniería para una comunicación que funciona.

Son las 3 AM. Se cayó producción.

A smartphone is shown with a red critical alert box overlaid on a README.md file. The alert box contains the text: "CRITICAL ALERT: Production Down - [Service-XYZ] 3:01 AM. Immediate action required. Check logs." with a red "ACKNOWLEDGE" button below it. The README.md file shows configuration instructions:

README.md - Configuración

```
1 # Configuración  
2  
3 La configuración debe ser actualizada antes del despliegue.
```

Three thought bubbles appear from the right side of the slide, corresponding to the numbered steps in the README:

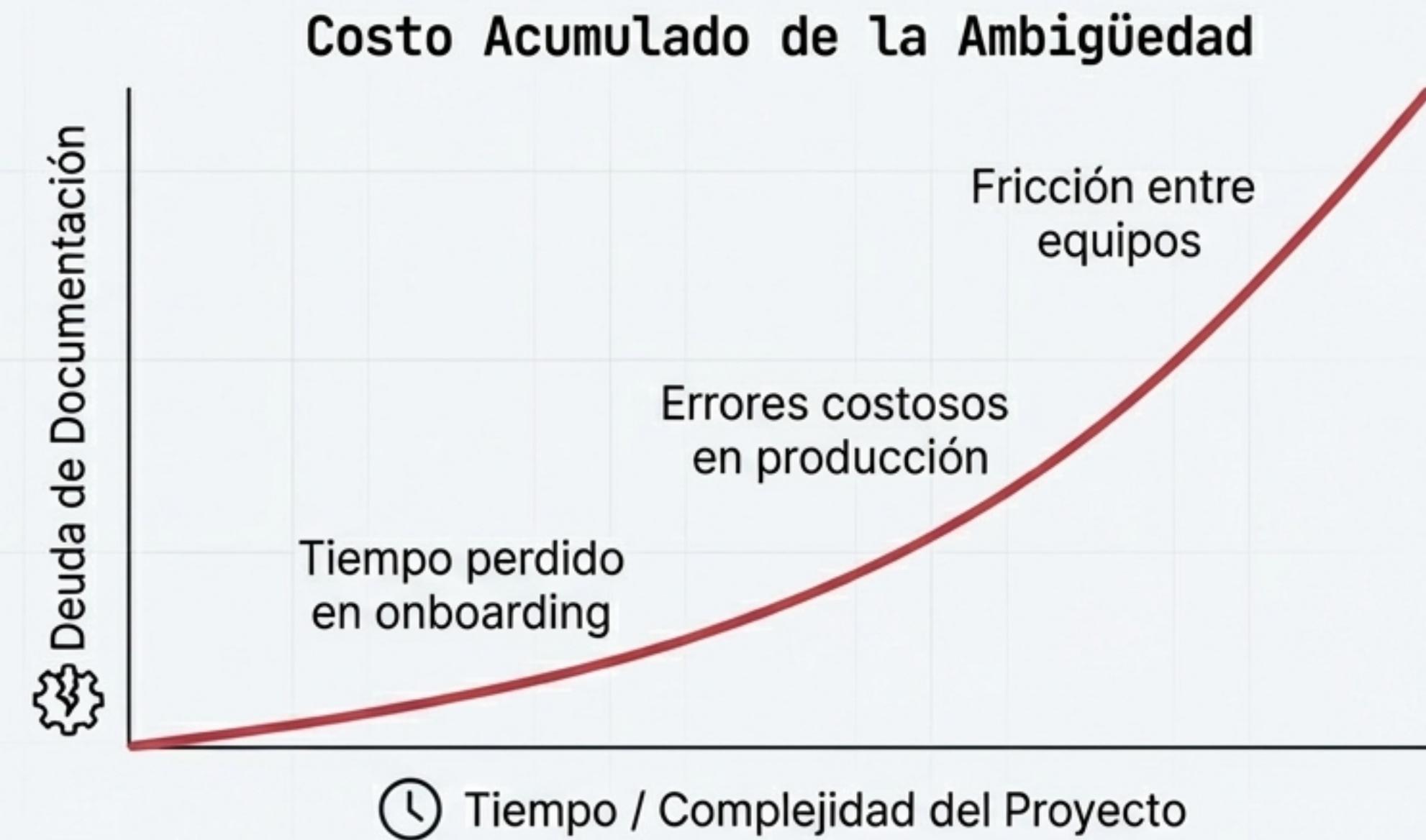
- ¿Por quién?
- ¿Debo hacerlo yo manualmente?
- ¿Lo hace el pipeline de CI/CD automáticamente?

La ambigüedad no es un problema de estilo.
Es un defecto crítico (bug).

El Verdadero Culpable: La Deuda de Documentación

Así como el código de baja calidad, la documentación poco clara acumula un "interés" que pagamos con:

- ⌚ **Tiempo perdido** en onboarding.
- 💡 **Errores costosos** en producción.
- ⇄ **Fricción** entre equipos.



Nuestra misión: Tratar la escritura con el mismo rigor que el código.
Vamos a identificar los *code smells* y a refactorizar.

Primer Refactor: Eliminar la Voz Pasiva

El **code smell** de la ambigüedad.

La voz pasiva oculta al actor, creando incertidumbre.

‘Los datos fueron procesados.’

¿Por quién? ¿Por qué?

‘El backend procesa los datos.’ ✓

Beneficios del Refactor

Precisión

Se define explícitamente el actor del sistema.

Concisión

Oraciones 20-30% más cortas en promedio.

Accionabilidad

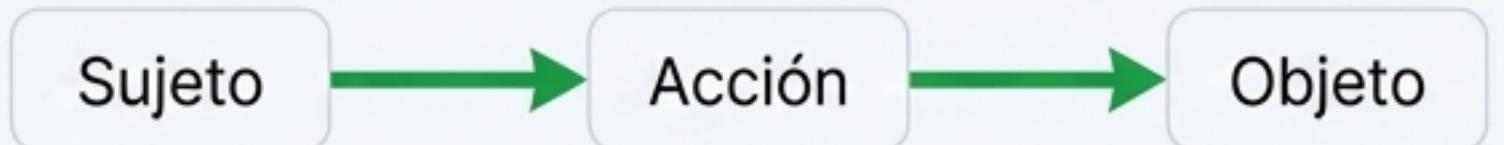
El lector sabe qué hacer o qué esperar.

El Benchmark de Rendimiento Cognitivo

Voz Activa (Eficiente)

El servidor lanza una excepción.

Complejidad: $O(1)$ - Acceso directo.



Flujo Mental: Sujeto → Acción → Objeto. El cerebro procesa la información de forma lineal.

Voz Pasiva (Ineficiente)

Una excepción es lanzada por el servidor.

Complejidad: $O(n^2)$ - Búsqueda en contexto.



Flujo Mental: El cerebro espera, busca al actor y reconstruye la escena. Gastas ciclos de CPU del lector inútilmente.

Algoritmo de Refactorización

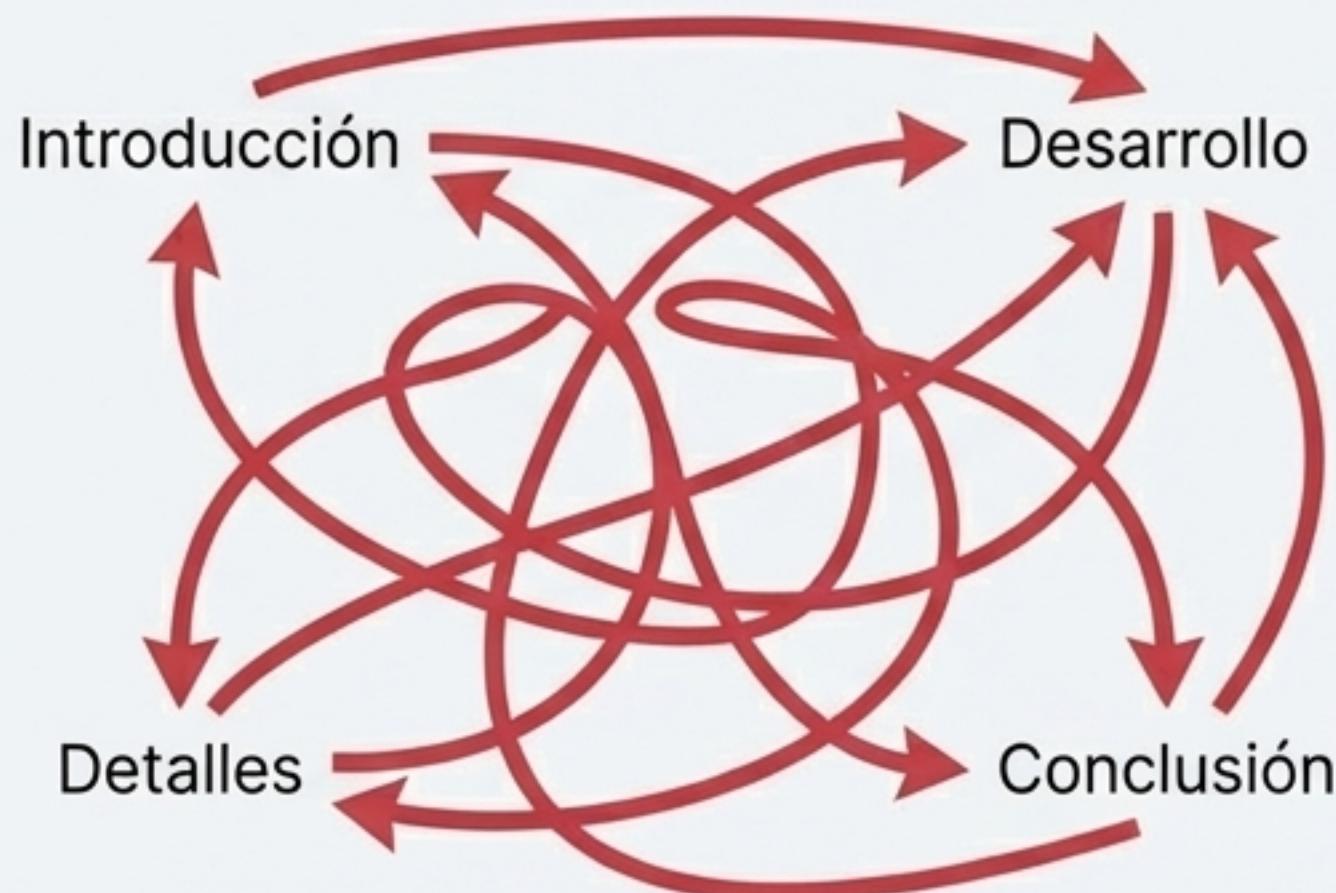
Pasos:

- 1. Detectar:** Busca el patrón ser + participio ("fue enviado", "es hecho").
- 2. Identificar Agente:**
Pregúntate: "¿Quién o qué realiza la acción?".
- 3. Reubicar:** Mueve el Agente al inicio de la oración.
- 4. Limpiar:** Elimina los verbos auxiliares ("ser", "estar").

- "Una excepción es lanzada por el servidor si el token es inválido." (12 palabras)
- + "El servidor lanza una excepción si el token es inválido." (9 palabras, 25% más corto)
- "El botón 'Deploy' deberá ser presionado cuando los tests hayan pasado."
- + "Presione el botón 'Deploy' una vez que pasen los tests."

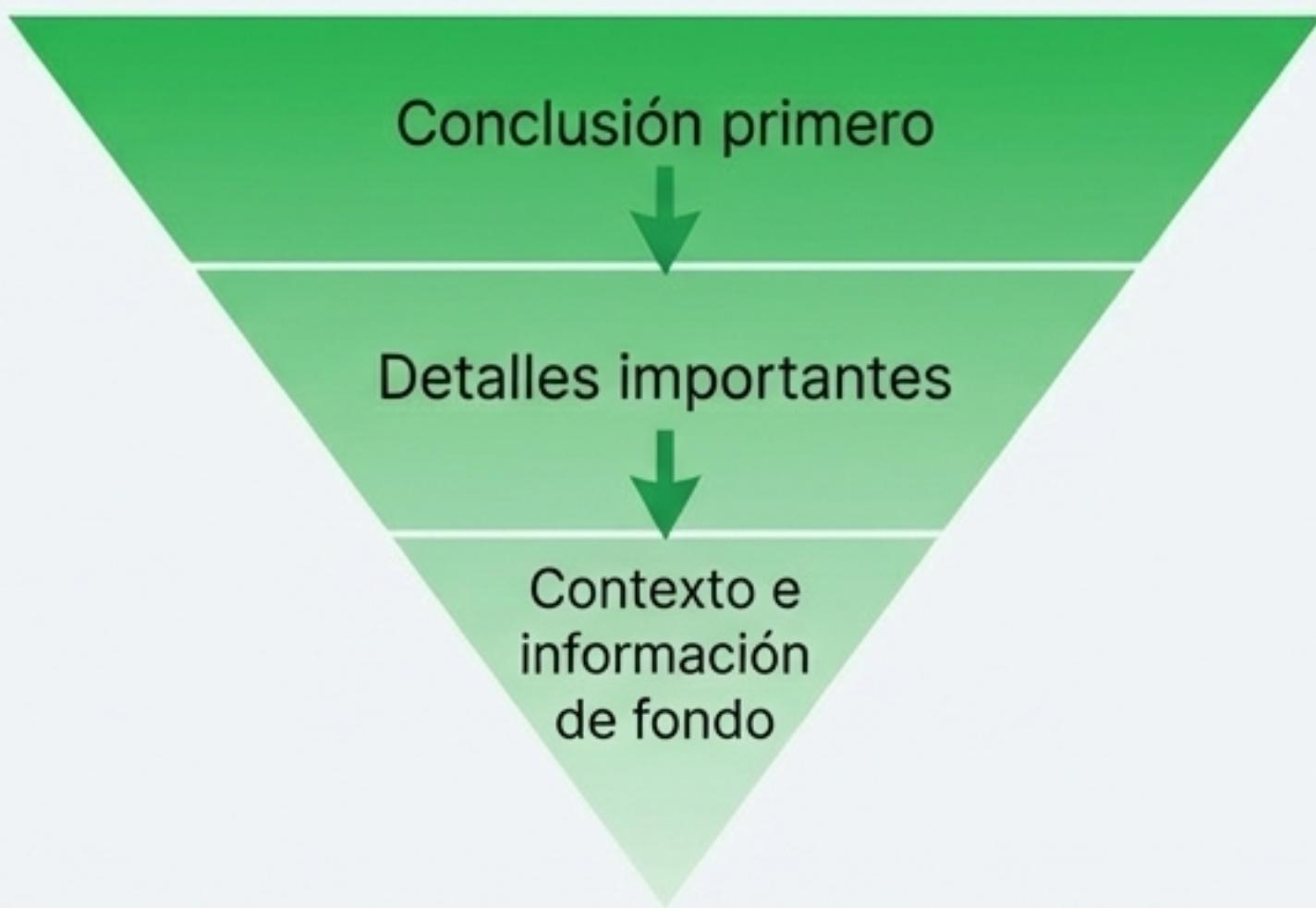
Segundo Refactor: Aplicar el Patrón de Pirámide Invertida

El Anti-Patrón (Cronológico / Código Espagueti)



"Funciona para novelas de misterio. Para un ingeniero a las 3 AM, es un desastre."

El Patrón Correcto (Pirámide Invertida)



"Dile al lector qué pasó antes de explicarle cómo pasó."

Caso de Uso: Refactor de un Reporte de Estatus

Antes

Ayer revisamos los logs y vimos errores. Luego reiniciamos el servidor. Pareció funcionar, pero luego falló de nuevo. Investigamos la DB y vimos bloqueos. Al final, liberamos los bloqueos y el sistema subió.

Después

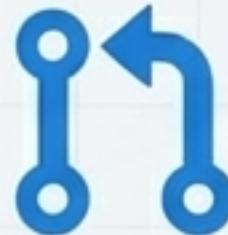
El sistema está estable ahora.

La causa raíz fueron bloqueos en la base de datos, que ya liberamos manualmente.

Ayer detectamos errores, un reinicio falló, y luego encontramos y resolvimos los bloqueos.

Regla de Oro

Si el usuario deja de leer después de la primera frase, ¿se llevó la información vital?



Pull Request Challenge: Refactoriza este Postmortem

Legacy Code

A las 14:00, una caída en la latencia fue detectada por el equipo de SRE. Una investigación fue iniciada. Se descubrió que una configuración errónea había sido aplicada al balanceador de carga. La configuración fue revertida y el servicio fue restaurado a las 14:15.

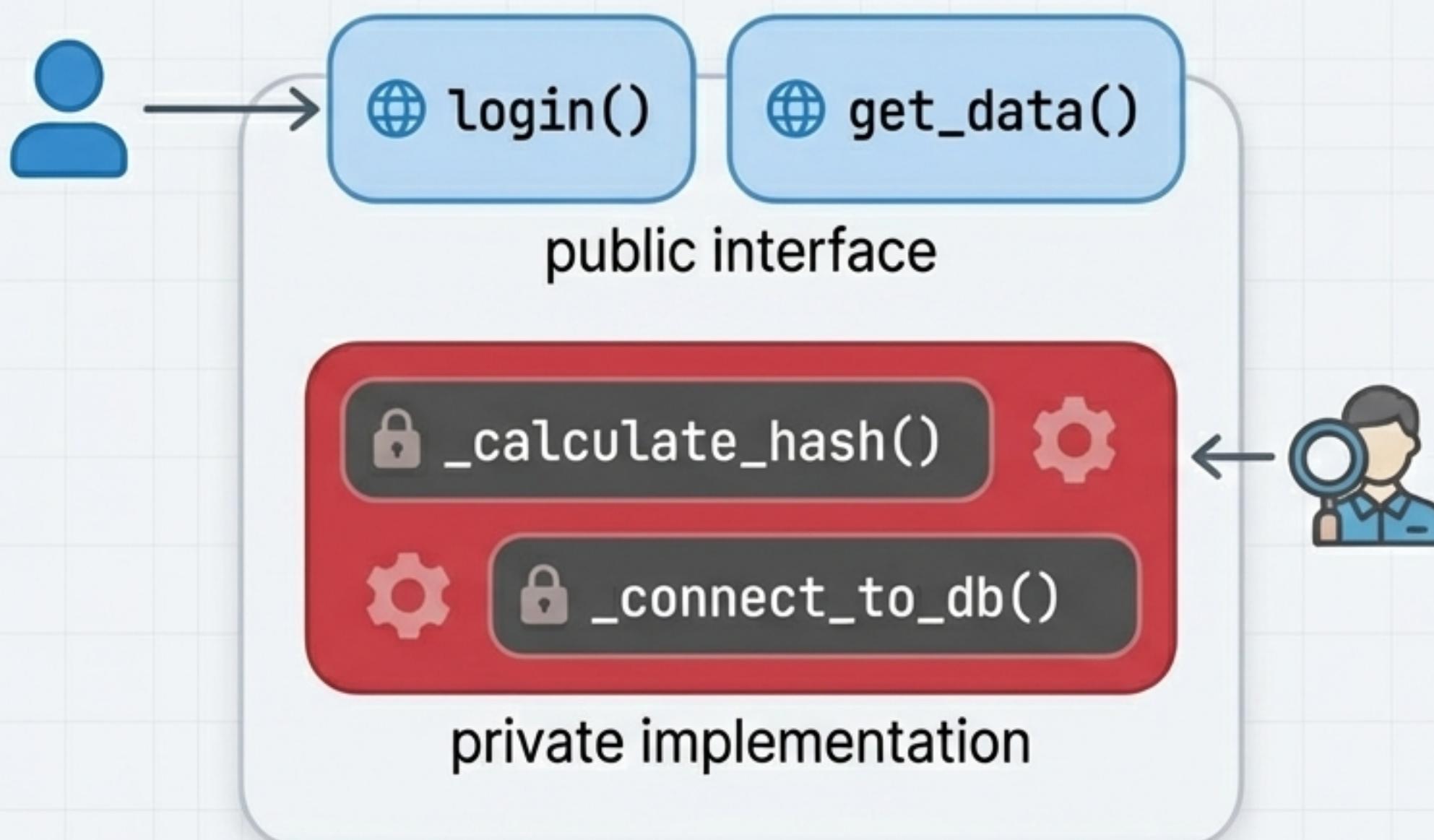
Tu Tarea: Refactoriza este texto usando Voz Activa y Pirámide Invertida para clarificar la cronología y responsabilidad.

› Solución Sugerida (revelable)

Restauramos el servicio a las 14:15 tras un incidente de latencia. ****Causa Raíz**:** Una configuración errónea en el balanceador de carga. ***Cronología*:** A las 14:00, el equipo de SRE detectó la caída, investigó, revirtió la configuración y restauró el servicio.

Tercer Refactor: Diseñar tu API de Audiencia

El error #1 en la documentación no es la gramática, es la **falta de empatía**. Escribirle a todos de la misma manera es como tener un solo método `public` que expone toda la lógica interna.



La Analogía Central

- **Documentación de Usuario** = `public interface`. Contrato, cómo se usa, qué esperar. (Caja Negra).
- **Documentación de Mantenedor** = `private implementation`. Lógica de negocio, por qué se hizo así, trade-offs. (Caja Blanca).

La Matriz de Audiencia: `public` vs. `private`

| Dimensión | USUARIO (API Pública) | MANTENEDOR (Implementación Privada) |
|-------------|--|--|
| Objetivo | Resolver un problema rápido. | Entender, modificar y arreglar el sistema. |
| Perspectiva | Caja Negra (Black Box) | Caja Blanca (White Box) |
| Pregunta | ¿Cómo uso X? | ¿Por qué X funciona así? |
| Artefacto | README.md, Guías de Inicio, API Reference | CONTRIBUTING.md, Architecture Docs, Comentarios de código |

Ejemplo: El Endpoint de Login

Versión Usuario (Swagger / API Docs)

POST /login

Autentica un usuario y devuelve un JWT.

Input:

{user, pass}

Output:

200 OK { token }

Error:

401 Invalid Credentials

(Nota: Al usuario no le importa si usas bcrypt o argon2)

Versión Mantenedor (Wiki Interna / Comentario de Código)

```
1 // Módulo de Autenticación
   // Usamos bcrypt con un work factor
   // de 12.
   // WARNING: No aumentar a >14, rompe
   // el SLA de latencia (<500ms).
   // La validación ocurre *antes* de
   // la conexión a DB para mitigar DoS.
```

Test de Empatía: El Camaleón Técnico

El Incidente: La base de datos principal se cayó por falta de espacio en disco.

Tu Misión: Escribe un mensaje de 1-2 líneas para cada audiencia.



Usuario Final
(en la app)

"[Placeholder for user's answer]"

"Estamos experimentando problemas de conexión. El servicio volverá pronto." (Caja Negra)



CTO / Manager
(en Slack)

"[Placeholder for user's answer]"

"Incidente crítico en DB principal. Causa: Disco lleno. ETA de solución: 15 mins." (Impacto y Tiempo)

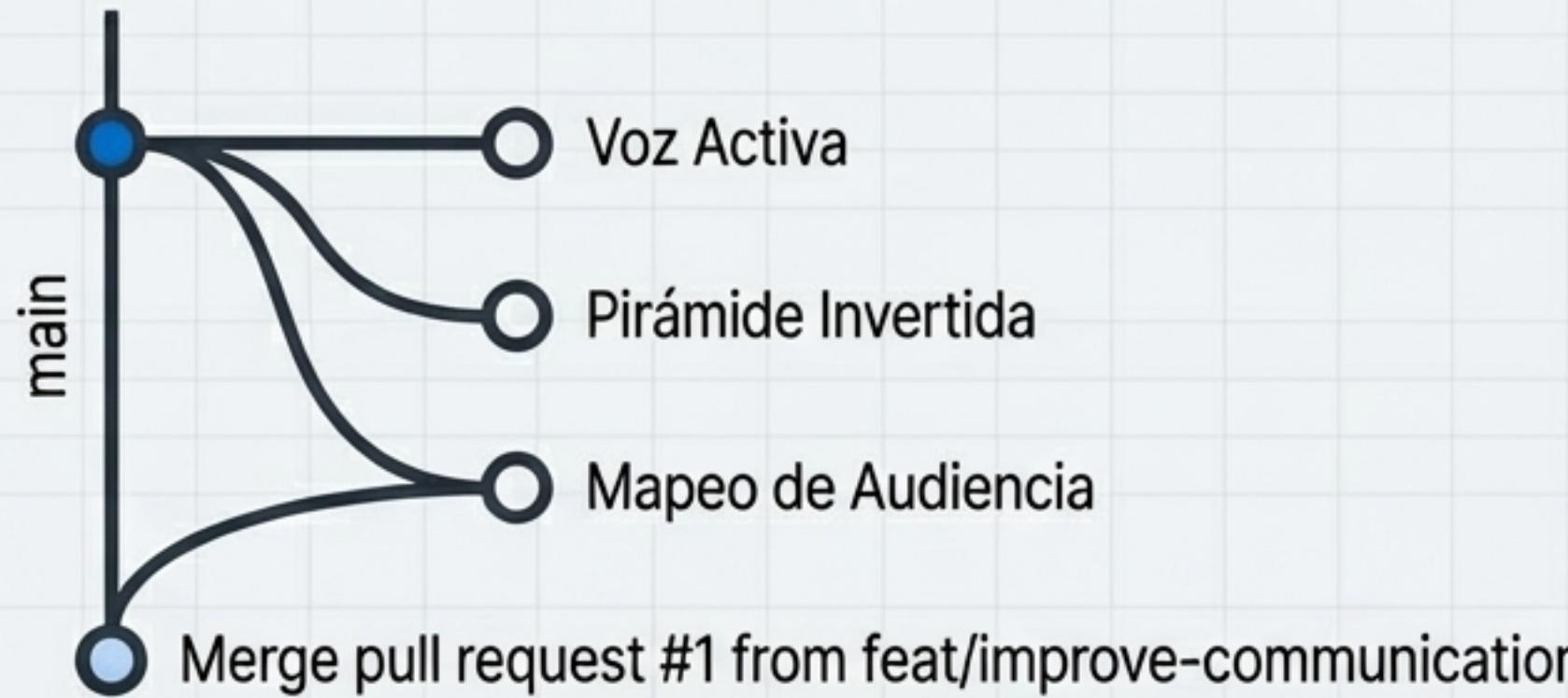


Equipo de DevOps
(en PagerDuty/canal de incidentes)

"[Placeholder for user's answer]"

"PostgreSQL master disk_usage al 100% en /var/lib. Necesitamos purgar logs o extender el volumen EBS." (Caja Blanca, Accionable)

git commit -m "feat: improve communication skills"



Resumen del Toolkit

- **Voz Activa:** Refactoriza a nivel de línea para eliminar la ambigüedad.
- **Pirámide Invertida:** Aplica un patrón de arquitectura para entregar valor al instante.
- **Mapeo de Audiencia:** Diseña una API de información clara, separando la interfaz de la implementación.

“ El Cambio de Mentalidad

> "Trata tu documentación como código. Exígele claridad, eficiencia y un buen diseño."

Llamada a la Acción

Abre el último README que escribiste. Encuentra una oración pasiva, un "lead" enterrado o un detalle de implementación que puedas refactorizar ahora mismo.

**“El código no está terminado
hasta que la documentación
está commiteada.”**