Iván García García
Guillermo Villacorta Moro
Ana Díez Díaz

Ingeniería del Software en Google

Ley de Hyrum, mito del genio y grandes jefes

La ley de Hyrum establece que cuando tienes un número grande y suficiente de consumidores de una API, todos los comportamientos de la API (incluso aquellos no definidos como parte del contrato público) llegarán a ser dependencia de alguien de forma eventual.

A menudo imaginamos al "programador genio" que hace todo solo, pero la realidad es que los grandes sistemas de software no funcionan así. Los grandes jefes se preocupan por el qué, mientras que los jefes tradicionales se preocupan por el cómo.

Desafíos de Escala

Los servicios que ofrece Google ya están inventados, pero nunca se habían ofrecido en tal magnitud. Por ello, sufren de problemas de **escala**, no de tipo. Buscan escalar y mejorar los servicios, teniendo **costes** de escalado sublineal respecto a la base del código para no tener que invertir cada vez más recursos en solo mantenerla.

El desafío no es solo organizar el código, sino también usar mejor los recursos sin tener que reconstruir todo desde cero. Con el crecimiento de la IA, la industria busca formas de hacer que los sistemas sean más **eficientes**. Esto permite ofrecer mejores servicios a más personas sin gastar tantos recursos.

Indexar

Al indexar el código permiten una mayor **legibilidad** de este, facilitando el uso de herramientas de mantenimiento para analizar y mejorar el software a gran escala, asó como, facilitar el trabajo de los desarrolladores.

Guías de Estilo

Están disponibles públicamente. Tienen reglas que se enfocan más en que el **código sea fácil de leer**, no solo de escribir., un ejemplo es que, aunque auto en C++ puede ser útil, prefieren escribir los tipos de manera explícita cuando es posible.

Tienen varias herramientas como analizadores de pila o verificadores que ayudan a los ingenieros a seguir las guías de estilo de forma más fácil. En lugar de centrarse solo en detalles de formato, como la identación, usan **herramientas automáticas** para aplicar estas reglas y verificar que el código cumpla con ellas.

Testing

Así como el código, los tests también deben ser **claros** y comprensibles. Las pruebas deben ser lo más **pequeñas** posibles y probar **una funcionalidad** cada una. Las pruebas simples, que cubren los escenarios comunes, ayudan a entender cómo un usuario podría usar el sistema. Además, son una excelente manera de experimentar con las API y detectar problemas temprano.

Flaky Tests

Son pruebas que dan resultados **inconsistentes**, pudiendo fallar, a veces, sin haber cambiado nada en el sistema o en la prueba misma. Son caras en Google porque se deben ejecutar varias veces para confirmar si realmente son inestables. Dado que Google realiza miles de pruebas, fallos poco frecuentes se vuelven comunes, lo que hace que estas pruebas sean aún más problemáticas.

Test Double

Se utilizan dobles de pruebas como **sustitutos de sistemas complejos**, como bases de datos, que permite probar el software sin necesidad de interactuar con estos sistemas reales. En Google, aunque son útiles para pruebas rápidas, pueden dar una falsa sensación de seguridad si no reflejan con precisión el entorno de producción.

Análisis estático

El análisis estático **examina el código fuente** sin ejecutarlo, lo que permite aprender sobre el comportamiento del programa. En Google, utilizan tanto análisis estático como dinámico, pero el primero resulta ser muy efectivo para entender el código y **detectar errores pronto** en el ciclo de desarrollo.

Tricorder

Google creó esta plataforma de análisis estático escalable para **detectar errores** y **patrones propensos a fallos**. Permite a los equipos de diferentes lenguajes desarrollar **herramientas de análisis** y aplicarlas automáticamente durante la revisión del código. Además, también sugiere mejoras y **corrige errores** directamente en el código durante el desarrollo.

Cambios de código a gran escala

En Google es un cambio tan grande que **no se puede hacer de una sola vez**, abarcando miles de archivos. Se hace para **mejorar la eficiencia** o la experiencia del desarrollador en toda la base de código.

Estos cambios se intentan **automatizar** todo lo posible y se dividen en bloques pequeños de decenas de archivos para facilitar su impementación y su validación.

Google creó la plataforma de croudsourcing **Busy Beavers** para gestionar los cambios no automatizables. En esta se suben los cambios que hay que implementar y los desarrolladores van eligiendo cuál de ellos hacer en cada momento.