Ingeniería de software en Google

Lecciones sobre programación aprendidas a lo largo del tiempo

Marcombo

Organizadas por Titus Winters,
Tom Manshreck y Hyrum Wright

Ingeniería de software en Google

Lecciones sobre programación aprendidas a lo largo del tiempo

Organizadas por Titus Winters,
Tom Manshreck y Hyrum Wright

Ingeniería de software en Google

Lecciones sobre programación aprendidas a lo largo del tiempo

Organizadas por Titus Winters,
Tom Manshreck y Hyrum Wright

Edición original publicada en inglés por O'Reilly con el título *Software Engineering at Google*, ISBN 978-1-492-08279-8 © 2020 Google, LLC.

Título de la edición en español:

Ingeniería de software en Google

Primera edición en español, 2022

www.marcombo.com

Diseño de portada: Karen Montgomery
Ilustración: Rebecca Demarest
Traducción: Francisco Martínez Carreño
Corrección: Mónica Muñoz Marinero

ISBN: 978-84-267-3487-7

Producción del ePub: booqlab

Contenidos

Parte II. Cultura

2. Cómo trabajar bien en equipo

3. Compartir conocimientos

8. Guías de estilo y normas

- ¿Por qué tenemos normas?
- Creación de normas
  - Principios rectores
  - Guía de estilo
- Cambio de las normas
  - El proceso
  - Árbitros de estilo
  - Excepciones
- Orientación
- Aplicación de las normas
  - Comprobadores de errores
  - Formateadores de código
- Conclusión
- Resumen

9. Revisión del código

- Flujo de revisión del código
- Cómo funciona la revisión de código en Google
- Beneficios de la revisión de código
  - Corrección de código
  - Comprensión de código
  - Coherencia del código
  - Beneficios psicológicos y culturales
  - Compartir conocimientos
- Mejores prácticas de la revisión de código
  - Sea cortés y profesional
  - Escriba cambios pequeños
  - Escriba descripciones de los cambios que tengan calidad
  - Mantenga al mínimo el número de revisores
  - Automatizar donde sea posible

21. Gestión de dependencias

22. Cambios a gran escala

24. Entrega continua

25. La computación como servicio

Prólogo

Siempre me han fascinado infinitamente los detalles de cómo hace Google las cosas.

He preguntado a mis amigos de Google para conseguir información sobre cómo funcionan realmente las cosas dentro de la empresa.

¿Cómo gestionan el depósito del código monolítico y masivo sin caerse?

¿Cómo colaboran con éxito decenas de miles de ingenieros en miles de proyectos?

¿Cómo mantienen la calidad de sus sistemas?

Trabajar con antiguos empleados de Google solo ha aumentado mi curiosidad. Si alguna vez ha trabajado con un exingeniero de Google (o «Xoogler», como se lo llama a veces), sin duda habrá escuchado la frase «en Google nosotros…». Salir de Google a otras empresas parece ser una experiencia impactante, al menos desde el lado de la ingeniería. Por lo que este forastero puede decir, los sistemas y procesos para escribir código en Google deben estar entre los mejores del mundo, dada la escala de la empresa y la frecuencia con la que la gente canta sus alabanzas.

En la ingeniería de *software* en Google, un conjunto de Googlers (y algunos Xooglers) nos brindan un plan extenso para muchas de las prácticas, herramientas e incluso elementos culturales que subyacen en la ingeniería de *software* en Google. Es fácil centrarse en las increíbles herramientas que Google ha creado para ayudar a escribir código, y este libro ofrece muchos detalles sobre esas herramientas. Pero también va más allá de la simple descripción de las herramientas, para ofrecernos la filosofía y los procesos que siguen los equipos de Google. Estos se pueden adaptar a

una variedad de circunstancias, tenga o no la escala y las herramientas. Para mi deleite, hay varios capítulos en los que se profundiza en varios aspectos de las pruebas automatizadas, un tema que continúa encontrando demasiada resistencia en nuestra industria.

Lo mejor de la tecnología es que nunca hay una sola forma de hacer algo. En cambio, existen una serie de contrapartidas que todos debemos tener en cuenta, dependiendo de las circunstancias de nuestro equipo y la situación. ¿Qué podemos obtener de forma económica del código abierto? ¿Qué puede construir nuestro equipo? ¿Qué tiene sentido apoyar para nuestra escala? Cuando interrogaba a mis amigos Googlers, quería que me hablaran del mundo en el extremo de la escala: rico en recursos, tanto en talento como en dinero, con grandes exigencias en el *software* que se crea.

Esta información anecdótica me dio ideas sobre algunas opciones que, de otro modo, no habría considerado.

Con este libro, hemos escrito esas opciones para que todos las lean. Por supuesto, Google es una empresa única y sería una tontería suponer que la forma correcta de administrar su organización de ingeniería de *software* es copiar con precisión su fórmula. Aplicado de manera práctica, este libro le dará ideas sobre cómo se pueden hacer las cosas y mucha información que puede utilizar para reforzar sus argumentos para adoptar las mejores prácticas como las pruebas, la compartición de conocimientos y la creación de equipos colaborativos.

Es posible que nunca tenga usted que construir algo parecido a Google, y que ni siquiera desee recurrir en su organización a las mismas técnicas que ellos aplican. Pero, si no está familiarizado con las prácticas que ha desarrollado Google, se está perdiendo una perspectiva sobre la ingeniería de *software* que proviene de decenas de miles de ingenieros que han trabajado en colaboración en el *software* durante más de dos décadas. Ese conocimiento es demasiado valioso para ignorarlo.

—Camille Fournier

Autora, *The Manager's Path*

Prefacio

Este libro se titula *Ingeniería de software en Google*. ¿Qué entendemos exactamente por «ingeniería de *software*»? ¿Qué distingue a la «ingeniería de *software*» de la «programación» o la «informática»? ¿Y por qué Google tendría una perspectiva única que añadir al corpus de bibliografía previo sobre ingeniería de *software* escrita durante los últimos cincuenta años?

Las expresiones «programación» e «ingeniería de *software*» se han utilizado de manera intercambiable durante bastante tiempo en nuestra industria, aunque cada término tiene un énfasis diferente y diversas implicaciones. Los estudiantes universitarios tienden a estudiar Ciencias de la computación y consiguen trabajo escribiendo código como «programadores».

La «ingeniería de *software*», sin embargo, suena más seria, como si implicara la aplicación de algunos conocimientos teóricos para construir algo real y preciso. Los ingenieros mecánicos, los ingenieros civiles, los ingenieros aeronáuticos y los de otras disciplinas de la ingeniería practican la ingeniería. Todos trabajan en el mundo real y utilizan la aplicación de sus conocimientos teóricos para crear algo real. Los ingenieros de *software* también crean «algo real», aunque es menos tangible que las cosas que crean otros ingenieros.

A diferencia de las profesiones de ingeniería más establecidas, la teoría o práctica actual de la ingeniería de *software* no es tan rigurosa. Los ingenieros aeronáuticos deben seguir pautas y prácticas rígidas, porque los errores en sus cálculos pueden causar daños reales; la programación, en general, no ha seguido tradicionalmente prácticas tan rigurosas. Pero, a medida que el *software* se integra más en nuestras vidas, debemos adoptar y confiar en

métodos de ingeniería más rigurosos. Esperamos que este libro ayude a otros a ver un camino hacia prácticas de *software* más confiables.

Programación a lo largo del tiempo

Proponemos que la «ingeniería de *software*» abarque no solo el acto de escribir código, sino todas las herramientas y procesos que utiliza una organización para construir y mantener ese código a lo largo del tiempo. ¿Qué prácticas puede introducir una organización de *software* que mejor mantengan su valioso código a largo plazo? ¿Cómo pueden los ingenieros hacer más sostenible la base de código y más rigurosa la propia disciplina de la ingeniería del *software*? No tenemos respuestas esenciales a estas preguntas, pero esperamos que la experiencia colectiva de Google durante las últimas dos décadas ilumine posibles caminos para encontrar esas respuestas.

Una idea clave que compartimos en este libro es que la ingeniería de *software* se puede considerar como una «programación integrada a lo largo del tiempo». ¿Qué prácticas podemos introducir en nuestro código para hacerlo sostenible, capaz de reaccionar ante los cambios necesarios, a lo largo de su ciclo de vida, desde su concepción hasta su introducción, pasando por su mantenimiento y su eliminación?

En el libro, se hace hincapié en tres principios fundamentales que, en nuestra opinión, las organizaciones de *software* deberían tener en cuenta a la hora de diseñar, crear la arquitectura y escribir su código:

Tiempo y cambio
: Cómo deberá adaptarse el código a lo largo de su vida.

Escala y crecimiento
: Cómo deberá adaptarse una organización a medida que evoluciona.

Contrapartidas y costes
: Cómo una organización toma decisiones, basándose en las lecciones del tiempo y el cambio, la escala y el crecimiento.

A lo largo de los capítulos, hemos tratado de enlazar con estos temas y señalar las formas en que tales principios afectan a las prácticas de ingeniería y permiten que sean sostenibles (véase el capítulo 1 para un análisis más completo).

Perspectiva de Google

Google tiene una perspectiva única sobre el crecimiento y la evolución de un ecosistema de *software* sostenible, derivado de nuestra escala y longevidad. Esperamos que las lecciones que hemos aprendido sean útiles a medida que su organización evolucione y adopte prácticas más sostenibles.

Dividimos los temas de este libro en tres aspectos principales del panorama de la ingeniería de *software* de Google:

- Cultura
- Procesos
- Herramientas

La cultura de Google es única, pero las lecciones que hemos aprendido en el desarrollo de nuestra cultura de ingeniería son de amplia aplicación. En los capítulos sobre la cultura (parte II), se destaca la naturaleza colectiva de una empresa de desarrollo de *software,* que el desarrollo de *software* es un esfuerzo de equipo y que los principios culturales adecuados son esenciales para que una organización crezca y se mantenga en buen funcionamiento.

Las técnicas descritas en los capítulos de «Procesos» (parte II) resultan familiares para la mayoría de los ingenieros de *software,* pero la base de código de gran tamaño y larga duración de Google proporciona una prueba de esfuerzo más completa para desarrollar las mejores prácticas. En esos capítulos, hemos intentado enfatizar lo que hemos descubierto que funciona a lo largo del tiempo y a escala, así como identificar áreas en las que aún no tenemos respuestas satisfactorias.

Por último, los capítulos dedicados a las herramientas (parte IV) ilustran el modo en que aprovechamos nuestras inversiones en la infraestructura de herramientas para proporcionar beneficios a la base de código, a medida que crece y envejece. En algunos casos, estas herramientas son específicas de Google, aunque indicamos las alternativas de código abierto o de terceros cuando corresponde. Esperamos que estos conocimientos básicos se apliquen a la mayoría de las organizaciones de ingeniería.

La cultura, los procesos y las herramientas que se describen en este libro describen las lecciones que, con suerte, un ingeniero de *software* típico aprende en el trabajo. Ciertamente, Google no tiene el monopolio de los buenos consejos, y nuestras experiencias, presentadas aquí, no pretenden dictar lo que su organización debe hacer. Este libro es nuestra perspectiva, pero esperamos que le resulte útil, ya sea adoptando estas lecciones directamente o utilizándolas como punto de partida a la hora de considerar sus propias prácticas, especializadas para su propio ámbito de problemas.

Esta obra tampoco pretende ser un sermón. El propio Google todavía aplica de forma imperfecta muchos de los conceptos de estas páginas. Las lecciones que hemos aprendido las hemos asimilado a través de nuestros fracasos: todavía cometemos errores, implementamos soluciones imperfectas y necesitamos repetir para mejorar. Sin embargo, el gran tamaño de la organización de ingeniería de Google garantiza que exista una diversidad de soluciones para cada problema. Esperamos que este libro contenga lo mejor de ese grupo.

Lo que no es este libro

Este libro no pretende contener el diseño de *software,* una disciplina que requiere su propio libro (y para el que ya existe mucho contenido). Aunque en la obra hay algo de código con fines ilustrativos, los principios son neutrales en cuanto al lenguaje y hay pocos consejos de «programación» en estos capítulos. Como resultado, este texto no trata muchos temas importantes en el desarrollo de *software:* administración de proyectos,