Engineer

base de código

todo el código fuente que se mantiene como una unidad

Qué es

Una base de código, codebase en inglés, a veces escrito en dos palabras como code base, es el conjunto completo de código fuente que un equipo modifica, revisa y entrega como una sola unidad. Cuando se habla de «la base de código», se alude al lugar donde acaba cada cambio de una aplicación: el propio código, sus pruebas, los scripts que lo construyen y comprueban, los cambios de la base de datos y los documentos que explican cómo encajan las piezas.

The Twelve-Factor App, una metodología para crear servicios web, ofrece la definición de trabajo más útil porque convierte el límite en algo comprobable. En un sistema centralizado de control de versiones como Subversion, una base de código es un repositorio. En uno distribuido como Git, es cualquier conjunto de repositorios que comparte un commit raíz: el primer commit del historial, el que no tiene padre. Dos repositorios nacidos del mismo primer commit son una sola base de código, por mucho que hayan divergido. Dos que empezaron por separado son dos bases de código, aunque contengan archivos casi idénticos.

El mismo documento acompaña esa definición con otra regla: una base de código, muchos despliegues. Cada despliegue es una instancia de la aplicación en ejecución, y suele haber varias a la vez. Producción es una, el entorno de preproducción es otra y también cuenta la copia local de cada desarrollador. Todas ejecutan la misma base de código, a menudo en distintos commits: una persona tiene trabajo que aún no ha llegado a preproducción y preproducción tiene trabajo que todavía no ha llegado a producción.

De ahí se deduce qué debe contener. Deben estar el código fuente y todo lo necesario para convertirlo en un programa funcional, incluido un manifiesto que nombre cada dependencia y su versión. Las propias dependencias suelen quedar fuera y se obtienen mediante un gestor de paquetes. También quedan fuera los

valores que cambian de un despliegue a otro y, sobre todo, las credenciales: una contraseña registrada una sola vez en un commit viaja después con cada copia del historial.

Una base de código puede contener varios programas. Un monorepo mantiene un frontend, una API y el código de la base de datos juntos en un repositorio, para que un único commit pueda cambiar los tres. Según la regla del commit raíz, eso sigue siendo una base de código. La opción contraria reparte una aplicación entre repositorios con historiales separados, y entonces cada repositorio es su propia base de código. Twelve-Factor es tajante sobre el resultado: si hay varias bases de código, se trata de un sistema distribuido y cada componente es una aplicación por derecho propio.

Utilice el concepto en una decisión

Aplique «¿Qué es una base de código?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. La estructura del sistema se define primero sobre el papel para que las piezas encajen y los costes se conozcan antes de escribir código.

Explore nuestros servicios Póngase en contacto →

Compruebe los supuestos detrás del término

Utilice la explicación y los trabajos enlazados para formular preguntas sobre su proyecto. Podemos examinar los supuestos con usted antes de elegir una herramienta. Un plan de infraestructura para 14 departamentos — 14 departamentos de investigación con un único plan de infraestructura

Un plan de infraestructura para 14 departamentos Póngase en contacto →

Por qué importa

El límite decide qué puede cambiar a la vez. Dentro de una base de código, un cambio en una API y en la pantalla que la llama puede escribirse, revisarse y fusionarse como una unidad. Entre dos bases de código, el mismo cambio se convierte en dos pull requests, dos publicaciones y un periodo en el que un lado va por delante del otro. Ambas opciones funcionan; el precio depende de dónde se trazó la línea.

La base de código es también la unidad a la que se aplican las normas. Una batería de pruebas, un linter, un formateador, una regla para mensajes de commit y un ejecutor de tareas toman «la base de código» como ámbito, y lo que queda fuera no lo comprueba ninguno.

Las decisiones tempranas se consolidan en una base de código antes que en cualquier otro lugar. Cada línea escrita después de una decisión la da por supuesta, así que una regla de separación en capas o un nivel de calidad fijados

en las primeras semanas dan forma a todo lo que viene después; revertirlos más tarde exige reescribir el código que creció a su alrededor.

Para las personas, la base de código es lo que recibe quien se incorpora al equipo. Si contiene todo lo necesario para construir, probar y ejecutar la aplicación, bastan una clonación y un comando para empezar. Si el conocimiento vive en parte en el código y en parte en la memoria de alguien, la primera semana se dedica a averiguar qué está en cada sitio.

Cómo se aplica en la práctica

Go, TypeScript, SQL, Python y shell comparten el repositorio de una plataforma, donde la base de código es un monorepo políglota y cada lenguaje trae su propia forma de construir, probar y ejecutar el lint. La respuesta fue un único punto de entrada: una capa de go-task que creció hasta 578 tareas con nombre, 50 en el archivo raíz y 528 en archivos inferiores con espacios de nombres, que cubren los modos de desarrollo nativo, Docker y HTTPS, la calidad del código, la gestión de la base de datos y el despliegue. Cualquiera puede ejecutar `task --list` y ver todas las herramientas. La cifra es también una advertencia: 578 tareas superan lo que alguien puede retener en la cabeza, por lo que los nombres y los espacios de nombres son lo que mantiene utilizable el conjunto.

Otro proyecto comenzó con un repositorio vacío el 25 de noviembre de 2025, y tres decisiones de sus primeras semanas seguían vigentes nueve meses y varios miles de commits después: una pila por capas, acceso a datos mediante funciones almacenadas y una exigencia de cero advertencias establecida cuando todavía había poco código al que aplicarla. El razonamiento tras la última decisión es el núcleo de este artículo: una norma introducida en el commit 5.000 es un proyecto de limpieza, mientras que la misma norma en el commit 50 es simplemente la forma de trabajar del repositorio.

Los límites internos de una base de código pueden ser el propio diseño. En la plataforma marítima construida en tres capas a cuya arquitectura contribuyó la fundadora en NextMariner, el frontend de Next.js se ocupa de la presentación y nada más; una API de Go deliberadamente delgada, construida con Huma sobre Fiber, enruta, valida y coordina; y la lógica de negocio reside en funciones de PostgreSQL. Cuando cambia una regla, cambia en una sola capa, en SQL. Los límites se documentaron y se hicieron cumplir en las revisiones, porque una convención que nadie hace respetar deja de serlo.

La base de código de este sitio traza otro tipo de frontera. Sus páginas de proyectos se escriben en un repositorio aparte, se exportan a este mediante un script y se registran en commits, por lo que el sitio puede construirse por sí solo. Dentro de esta base de código, esas páginas son archivos generados, y las instrucciones de trabajo indican que nunca se editan a mano.

Dónde surgen los problemas

El código copiado diverge

Dos aplicaciones necesitan la misma función auxiliar y lo más rápido es copiarla de una a otra. Un año después hay dos versiones con correcciones distintas y un error arreglado en una sigue activo en la otra. Twelve-Factor considera una infracción que varias aplicaciones compartan el mismo código e indica la solución: trasladar la parte compartida a una biblioteca, incorporarla mediante el gestor de dependencias y hacer que cada base de código declare la versión que utiliza.

Un secreto llega al historial

Una cadena de conexión o una clave de API se registra en un archivo de configuración y, desde entonces, permanece en el historial que lleva cada clonación, incluso después de que otro commit borre la línea. Conviene aplicar la prueba de Twelve-Factor antes de cada commit: ¿podría publicarse hoy esta base de código como código abierto sin exponer una sola credencial? Todo lo que impida superar esa prueba pertenece al entorno de cada despliegue.

Se edita a mano un archivo generado

Un archivo escrito por un script se parece a cualquier otro archivo fuente, así que alguien corrige directamente una errata. La siguiente ejecución del script repone el texto anterior y la corrección desaparece sin aviso. Identifique claramente las rutas generadas, mantenga el generador como único proceso que las escribe y dirija cada corrección a los datos de entrada del generador.

Los límites internos se debilitan

Con una fecha de entrega encima, una consulta se introduce directamente en el manejador de una petición porque es el camino más corto, y la separación en capas con la que empezó el proyecto deja de describir el código. Cada atajo es pequeño. Juntos convierten un cambio de dos líneas en uno que toca todas las capas. Una regla escrita ayuda, la revisión ayuda más y una comprobación que hace fallar la compilación cuando una capa accede al interior de otra ayuda todavía más.

Una norma solo cubre la mitad del código

Un linter configurado para el frontend deja de lado los scripts de shell y el SQL, y esos archivos acumulan exactamente los problemas que pretendía evitar. Un nivel de calidad para toda la base de código debe incluir cada uno de sus lenguajes: ese es el coste oculto de un repositorio políglota.

Un fork inicia un segundo producto

Un equipo copia un repositorio para empezar un producto relacionado. Las dos copias conservan el mismo commit raíz, de modo que, según la regla de Twelve-Factor, siguen siendo una sola base de código, ahora al servicio de dos aplicaciones, y la divergencia anterior empieza a escala de un repositorio entero. Comparar los commits raíz muestra en qué situación se está:

```
git -C app-a rev-list --max-parents=0 HEAD
git -C app-b rev-list --max-parents=0 HEAD
```

Una salida idéntica significa una raíz y una base de código compartidas; una salida distinta significa dos historiales separados.

Lecturas adicionales

- I. Base de código — la regla de una base de código y muchos despliegues, y la prueba del commit raíz compartido en la que se apoya.
- II. Dependencias — declarar cada dependencia en un manifiesto y aislarla, para que una base de código especifique exactamente qué incorpora.
- III. Configuración — la prueba de publicación como código abierto que determina qué debe quedar fuera de una base de código.
- gitsubmodules — cómo Git incorpora un repositorio dentro de otro conservando cada uno su propio historial.
- git-rev-list — la referencia para enumerar commits, incluidos los commits raíz de un historial.



Glosario

Elija según sus restricciones reales

Describa el flujo de trabajo, las interfaces existentes y los usuarios que necesitan el cambio. Podemos definir una entrega comprobable y un traspaso que permita a su equipo mantener lo entregado.

Vea cuánto cuesta Póngase en contacto →

Haga práctica la terminología

Pregunte cómo se aplica «¿Qué es una base de código?» a su organización. Basta un ejemplo concreto de su sistema actual para iniciar la conversación.

Explore nuestros servicios

welcome@engineer.company +45 50 10 61 84 Copenhague, Dinamarca

Llévese esta página

Tarjeta de contacto

Creado con esmero y cariño por D.

CVR 45969703 · Tagensvej 105C, 3. th, 2200 København N

Derechos de autor © 2025 – actualidad · Engineer ApS · Todos los derechos reservados.

Impreso desde https://engineer.company/es/glossary/codebase/