Engineer

arquitectura de sistemas

🧩 cómo encajan las partes de un sistema y por qué

Qué es

La arquitectura, llamada en software arquitectura de software o de sistemas, es el conjunto de conceptos y propiedades fundamentales de un sistema en su entorno, expresado en sus elementos, sus relaciones y los principios que rigen su diseño y evolución. Esa formulación sigue NIST SP 800-160 Volume 1 Revision 1, Engineering Trustworthy Secure Systems, publicado en noviembre de 2022.

Cada parte de la definición cumple una función. Los elementos son las piezas: servicios, bases de datos, colas, el frontend y los módulos que contienen. Las relaciones son las dependencias entre esas piezas y lo que circula entre ellas. Los principios son las reglas que deciden dónde encaja el trabajo nuevo y evitan que elementos y relaciones se desvíen a medida que crece el sistema. La expresión «en su entorno» hace que los usuarios, los sistemas vecinos y las restricciones formen parte de la arquitectura, y «evolución» significa que una descripción válida solo el día en que se dibujó está incompleta.

No toda elección de diseño es arquitectónica. Quienes mantienen el formato Architectural Decision Record (ADR) la definen como «una elección de diseño justificada que aborda un requisito funcional o no funcional de importancia arquitectónica». Es significativo un requisito cuyo efecto sobre arquitectura y calidad puede medirse. La plantilla arc42 es más directa: la sección se dedica a «decisiones importantes, costosas, arriesgadas o controvertidas».

Como ningún dibujo puede mostrarlo todo, la arquitectura se describe mediante vistas. El modelo C4, creado por Simon Brown, propone diagramas que funcionan como mapas en los que se acerca o se aleja el punto de vista: un diagrama de contexto que muestra el sistema entre sus usuarios y los sistemas que lo rodean; uno de contenedores que muestra las aplicaciones y los almacenes de datos que

contiene; uno de componentes para un contenedor y, cuando resulta útil, uno de código. arc42 organiza la descripción escrita en doce secciones, desde objetivos y restricciones hasta decisiones, riesgos y deuda técnica.

Utilice el concepto en una decisión

Aplique «¿Qué es la arquitectura de sistemas?» 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

La cuestión que la arquitectura plantea al equipo no es tanto qué estructura elegir como qué decisiones tomar ahora, deliberadamente y por escrito, y cuáles dejar abiertas.

Resolver pronto una cuestión estructural obliga a pagar antes de saber mucho: se elige con la menor información que tendrá el proyecto y se ralentizan las primeras semanas. Dejarla abierta no impide que se resuelva: lo hace cada cambio conveniente de quien tocó el código por última vez. La salida es separar dos clases. Algunas decisiones son caras de cambiar cuando el código depende de ellas: división del sistema, propietario de reglas de negocio, acceso a datos y nivel exigido a cada cambio. Merecen tomarse pronto y a propósito. La mayoría restante es barata de revisar y puede esperar a que el código muestre qué necesita.

Documentar es la otra mitad de la decisión, y es donde sobrevive el razonamiento. Un ADR recoge una decisión con su justificación, sus contrapartidas y sus consecuencias, y el conjunto de ADR de un proyecto forma su registro de decisiones. El sitio de ADR atribuye la popularización de la idea a la publicación de Michael Nygard de 2011, «Documenting Architecture Decisions». El coste es una página de escritura por decisión. A cambio, quien la lea después puede distinguir una contrapartida deliberada de un accidente y comprobar si su motivo sigue siendo válido.

Los objetivos de calidad permiten decidir entre estructuras que funcionarían por igual. arc42 pide los tres principales, cinco como máximo, los que más importan a las principales partes interesadas, y advierte que son distintos de los objetivos del proyecto. Dos diseños con las mismas funciones pueden diferir por completo en cómo fallan, escalan o cambian, y solo los objetivos de calidad dicen qué diferencia cuenta.

Cómo se aplica en la práctica

El repositorio de la plataforma abrió vacío el 25 de noviembre de 2025, y la página sobre sus decisiones fundacionales sostiene que esas elecciones son «baratas de tomar al tercer día y casi imposibles de revertir al sexto mes». Dos de las tres decisiones tomadas entonces dieron forma a la estructura. Las tecnologías se organizaron en capas, cada una con una sola tarea: un frontend Next.js, una API Go que valida y reenvía y una capa de funciones PostgreSQL propietaria de las reglas de negocio. El acceso a datos se situó detrás de funciones almacenadas desde el principio, porque incorporarlo más tarde «habría supuesto reescribir todos los manejadores». Nueve meses y varios miles de commits después, según la página, las capas siguen sin desdibujarse y ningún manejador habla directamente con una tabla. La prueba de una decisión fundacional no es si sonaba bien, sino si sobrevivió al volumen de trabajo que vino después.

La estructura que se deja al azar se fija por accidente, como muestra la página sobre el desarrollo de mapas SIG. El mapa había crecido por partes: datos espaciales en PostgreSQL, representación en Mapbox, ReactJS delante y NodeJS detrás. Cada parte funcionaba por sí sola; «simplemente no se había construido para que todo encajara». El puesto abarcaba el desarrollo full stack y la dirección técnica, y la arquitectura se definió primero. Después, las responsabilidades «se devolvían una y otra vez a la capa que les correspondía cuando alguna empezaba a pasar a la siguiente», y buena parte del trabajo consistía en «detectar las pequeñas incoherencias antes de que se consolidaran como arquitectura».

La página del plano de contenedores considera la explicación escrita el resultado real. El motor de contenedores predeterminado en el sector publica los puertos por encima de las reglas del cortafuegos del host, por lo que el plano usa un motor sin demonio que dirige unidades generadas por el propio supervisor del sistema operativo, sin ningún puerto publicado. Una de las contrapartidas queda documentada como tal: las unidades se ejecutan a nivel de sistema en vez de hacerlo sin privilegios de root, lo que simplifica la automatización a costa del aislamiento más estricto que ofrecería rootless. La nota indica en qué dirección

cambiar si el riesgo de escape de un contenedor llega a importar más. Eso es lo que conserva un ADR: una decisión con su justificación, sus contrapartidas y sus consecuencias. La página da más valor a escribir la elección junto con su motivo que a la elección misma, porque «todos los artículos que lea recomendarán a la siguiente persona que use la opción predeterminada».

Dónde surgen los problemas

Una persona nueva y cuidadosa revierte una decisión deliberada. El motivo solo estaba en una cabeza; parece un error y la opción predeterminada de los tutoriales parece arreglarlo. Escriba un ADR para decisiones caras de revertir: contexto, opciones, elección, consecuencias y qué justificaría revisarla. Guárdelo junto al código para revisar juntos sus cambios.

Las reglas de negocio aparecen en tres capas a la vez. Cada cambio colocó su lógica donde resultaba cómodo aquella tarde, hasta que la misma regla acabó comprobándose en el frontend, la API y la base de datos, de forma un poco distinta en cada uno. Defina qué le corresponde a cada capa, déjelo por escrito y devuelva una responsabilidad a su sitio en cuanto se filtre a otra, mientras la copia desplazada aún sea pequeña. Cuando la plataforma lo permita, haga cumplir también ese límite: por ejemplo, permita que la API llame a funciones de la base de datos sin concederle derechos sobre las tablas.

Una regla introducida en el sexto mes se convierte en un proyecto de limpieza. La decisión de la que todo depende quedó abierta y cada commit posterior dio por supuesto lo que resultara ser cierto en ese momento. Tome primero las pocas decisiones costosas y deje el resto abierto deliberadamente. Dígalo en el registro de decisiones para que una pregunta abierta sea visiblemente abierta, en vez de recibir una respuesta silenciosa en el siguiente cambio.

El diagrama de arquitectura no explica nada a nadie. Un solo dibujo mezcla usuarios, servicios, clases y tablas, y las líneas entre las cajas no llevan etiqueta o solo dicen «uses». Dibuje un diagrama por nivel de acercamiento, como hace C4, y etiquete cada línea con la dirección y la intención de la relación; entre contenedores, nombre también la tecnología o el protocolo. La guía de notación de C4 pide exactamente eso y desaconseja etiquetas de una sola palabra, como «Uses».

Lecturas adicionales

- Arquitectura, glosario CSRC de NIST — definición usada, junto a las de FIPS y arquitectura de seguridad de la misma entrada.
- NIST SP 800-160 Vol. 1 Rev. 1 — Engineering Trustworthy Secure Systems, fuente de la definición.
- Architectural Decision Records — qué hace arquitectónica una decisión, plantillas y herramientas.
- El modelo C4 — cuatro niveles del contexto al código, notación y lista de revisión.
- arc42 — consejos prácticos y ejemplos para cada una de las doce secciones de la plantilla arc42.

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 la arquitectura de sistemas?» 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/architecture/