Engineer

backend

la parte de un sistema que quien lee nunca carga

Qué es

Un backend es la parte de un sistema de software que se ejecuta en máquinas controladas por su operador: almacena datos, aplica reglas sobre quién puede hacer qué y responde a solicitudes de navegadores, apps y otros servicios. También se escribe back end o back-end; el mismo ámbito suele llamarse lado del servidor.

En términos HTTP, es el servidor de origen de RFC 9110: «un programa capaz de generar respuestas autoritativas para un recurso de destino determinado». Architecture of the World Wide Web del W3C añade la perspectiva del lector: un URI identifica un recurso y lo que viaja al navegador es una representación, datos que codifican su estado. El cliente ve una página de pedido; el backend conserva pedido, existencias y pago y decide qué incluir en la respuesta.

Un backend suele exponer una API, un conjunto de endpoints HTTP que puede invocar un frontend, una app móvil o el sistema de otra empresa. La API es un contrato: qué rutas existen, qué acepta cada una, qué devuelve y qué errores puede producir. OpenAPI Specification es la forma habitual de poner ese contrato por escrito, como una descripción independiente del lenguaje de programación a partir de la cual las herramientas pueden generar clientes, documentación y pruebas. Detrás de la API, el trabajo suele dividirse en capas: gestión de solicitudes, lógica de negocio y almacenamiento.

Utilice el concepto en una decisión

Aplique «¿Qué es un backend?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. La parte que el usuario no ve: dónde se guardan los datos, qué reglas se aplican y cómo pueden comunicarse otros sistemas con ella.

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. Diseño y operación de 6 flujos ETL/ELT — 6 flujos ETL en tres motores de bases de datos

Diseño y operación de 6 flujos ETL/ELT Póngase en contacto →

Por qué importa

La primera pregunta es si el sistema necesita siquiera un backend que se ejecute al recibir cada solicitud. Una página que es igual para todos los visitantes puede generarse una vez y servirse como archivos. Un backend en ejecución se justifica cuando una solicitud tiene que cambiar algo o depende de quién la haya enviado: un pedido, un inicio de sesión o una búsqueda sobre datos privados. También conlleva un coste permanente, porque el software que responde a solicitudes de Internet debe protegerse, parchearse y pagarse.

La segunda cuestión es la confianza. Como el backend se ejecuta en máquinas controladas por su operador, es el lugar donde se puede hacer cumplir una regla. Cualquier comprobación del frontend puede saltársela alguien que envíe solicitudes directamente, por lo que todas las reglas importantes, desde un precio hasta un permiso, se comprueban en el servidor respecto al usuario que ha iniciado sesión y al registro concreto. Application Security Verification Standard de OWASP lo convierte en requisitos que un equipo puede implementar y comprobar.

La tercera cuestión es dónde reside la lógica. Una regla repartida entre un manejador, una clase de servicio y un trigger de base de datos tiene tres lugares que cambiar y tres copias que pueden divergir. Algunos equipos mantienen las reglas en el código de aplicación y usan la base de datos como almacenamiento; otros trasladan las reglas y la conformación de los datos a la base de datos y mantienen ligera la API. Ambas opciones pueden funcionar, siempre que cada regla tenga un único hogar.

La cuarta cuestión es el contrato con los clientes. HTTP da a cada respuesta un código de estado con un significado definido, y los clientes, proxies y herramientas de monitorización actúan según ese significado. Un 422 Unprocessable Content indica que la solicitud era sintácticamente correcta y usaba un formato admitido, pero era semánticamente errónea, como una reserva con fecha en el pasado. RFC 9457 añade un cuerpo estándar para los errores, llamado problem details: un pequeño objeto JSON con tipo, título, estado, detalle e instancia, enviado como `application/problem+json`, para que un cliente distinga un fallo de otro sin analizar prosa.

Cómo se aplica en la práctica

Una carrera de relevos de transformaciones es la ruta habitual entre la base de datos y el navegador: la API convierte las filas en estructuras, les cambia la forma y las serializa de nuevo como JSON. En una plataforma, el backend deja que la base de datos construya cada respuesta y la reenvía intacta. Las funciones PostgreSQL ensamblan la respuesta completa como JSON donde ya residen los datos, y el manejador Go la recibe como `json.RawMessage` y la reenvía sin tocarla. Un pequeño auxiliar `QueryJSON` convirtió esa ruta en la de menor resistencia. Desaparecieron los DTO, los mapeadores y las estructuras de respuesta, el código de los manejadores se acortó drásticamente y un cambio en lo que devuelve una función llega directamente al cliente sin que nadie edite código de manejadores.

Las solicitudes entrantes reciben el mismo tratamiento de fuente única mediante un contrato que empieza en la base de datos y avanza hacia fuera. En el backend, el esquema y sus funciones definen las estructuras y los tipos Go definen la API. Huma emite la descripción OpenAPI a partir de esos tipos, y de esa descripción se genera el cliente del frontend: 44.076 líneas de TypeScript tipado. Los límites que la interfaz aplica a un formulario proceden del mismo lugar, en vez de volver a escribirse en un validador. Cada salto tiene una comprobación, de modo que un campo no pueda cambiar solo en un lado: la discrepancia hace fallar el build minutos después del cambio. El trabajo que esto cuesta es regenerar, porque cada cambio en una estructura definida por el backend debe avanzar por la cadena volviendo a generar todos los eslabones siguientes.

En la etapa de diseño documentada, este sitio tomó la decisión contraria: se generaba por completo al construirlo y no necesitaba un backend de aplicación para atender las solicitudes. Sus cuatro requisitos, entre ellos funcionar en tres idiomas y prescindir de un servicio de aplicación facturado por separado, llevaron a esa elección: un backend de aplicación en ejecución exigiría protección, parches y una explicación en la página de privacidad. Fuera de la excepción limitada de JavaScript documentada para un filtro de portfolio, las funciones interactivas debían resolverse al construir el sitio u omitirse. Esto describe la etapa de diseño documentada, no el inventario ni la implementación actuales del sitio; no significa que el alojamiento no tuviera coste.

Dónde surgen los problemas

Los fallos del backend suelen callar hasta recibir una solicitud no prevista.

Un usuario cambia un registro que pertenece a otra persona. El endpoint comprueba que quien llama haya iniciado sesión y después carga cualquier ID que indique la URL, de modo que basta cambiar `/invoices/1041` por `/invoices/1042` en una solicitud. Compruebe en el servidor la propiedad o los permisos de cada objeto que toque la solicitud, en un único punto por el que pase todo manejador, y mantenga una prueba que solicite el registro de otra cuenta y espere un 403 o un 404.

Una acción ocurre dos veces. Un rastreador o un navegador que precarga páginas sigue un enlace que cancela algo mediante una solicitud GET, y la acción se ejecuta sin que nadie haga clic. RFC 9110 define GET, HEAD, OPTIONS y TRACE como métodos seguros para que los rastreadores y la precarga funcionen sin temor a causar daños. La otra vía es un reintento: una solicitud de pago agota el tiempo de espera, el cliente la envía otra vez y se cobra dos veces al cliente. Sitúe todo cambio de estado detrás de POST, PUT, PATCH o DELETE, y dé a las escrituras que puedan reintentarse una clave que el servidor recuerde, para que la segunda copia devuelva el resultado de la primera.

Todos los errores vuelven como 500 o como 200 con un mensaje de error dentro. Un manejador general informa de todos los fallos de la misma manera, por lo que la monitorización cuenta los errores de introducción de datos como caídas y la lógica de reintentos repite solicitudes que fallarán siempre. Asigne a cada fallo el código de estado que lo describe: 400 o 422 para una entrada incorrecta, 401 y 403 para identidad y permisos, 404 para un recurso ausente, 409 para un conflicto y 5xx solo para los fallos del propio servidor. Envíe los detalles como problem details:

```
{
    "type": "https://example.com/problems/booking-in-past",
    "title": "Booking date is in the past",
    "status": 422,
    "detail": "The date 2026-10-01 is before today.",
    "instance": "/bookings/8812"
}
```

RFC 9457 dice que los problem details «no son una herramienta de depuración de la implementación subyacente», por lo que las trazas de pila y el SQL permanecen en los registros del servidor.

Las páginas se ralentizan a medida que crecen los datos. Un manejador carga una lista de pedidos y después consulta el cliente de cada uno, por lo que una página de 50 pedidos genera 51 viajes de ida y vuelta a la base de datos. Parece rápido en desarrollo con diez filas y se arrastra en producción con diez mil. Obtenga los datos relacionados en una sola consulta con un join o deje que la base de datos ensamble toda la respuesta, y vigile el número de consultas por solicitud con la misma atención que el tiempo de respuesta.

Lecturas adicionales

- Architecture of the World Wide Web, Volume One — la Recomendación del W3C sobre recursos, representaciones e interacciones seguras e inseguras.
- RFC 9110: HTTP Semantics — el estándar del IETF para métodos, códigos de estado y qué hace que una solicitud sea segura o idempotente.
- OpenAPI Specification — el formato independiente del lenguaje para describir una API HTTP, leído por personas y por generadores de código.
- RFC 9457: Problem Details for HTTP APIs — la estructura JSON estándar para respuestas de error legibles por máquinas.
- Application Security Verification Standard de OWASP — una lista de requisitos de seguridad para construir y probar aplicaciones web.

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 un backend?» 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/backend/