Engineer

Frontend

capa de presentación e interacción de un sistema

Qué es

Un frontend es la parte de un sistema de software que las personas ven y manejan: páginas, pantallas, formularios y controles, junto al código que los dibuja y responde a las entradas. También se escribe front end o front-end, y en la web suele llamarse lado del cliente al mismo ámbito.

En la web, un frontend se compone de tres lenguajes que entiende un navegador. HTML aporta estructura y significado, CSS decide el aspecto y JavaScript añade comportamiento. El navegador es lo que *Architecture of the World Wide Web* del W3C llama agente de usuario: software que actúa en nombre de una persona, obtiene datos de servidores y los presenta. Todo lo que le envía un frontend acaba en un dispositivo ajeno.

Su contraparte es el backend, que guarda datos, aplica reglas de negocio y responde a solicitudes. Ambos hablan por HTTP, normalmente mediante una API. Resulta tentador situar la frontera en la red, pero los marcos actuales la difuminan. En Next.js, páginas y estructuras de página son Server Components de forma predeterminada: producen lo que ve el usuario y se ejecutan en un servidor. La directiva `"use client"` al principio de un archivo marca la frontera; ese archivo, sus importaciones y los componentes que renderiza entran en el paquete enviado al navegador. El código se considera frontend por su función: presentación e interacción. Dónde se ejecuta es otra decisión.

Utilice el concepto en una decisión

Aplique «¿Qué es un frontend?» a una decisión real. Podemos ayudarle a relacionar el concepto con el sistema que está construyendo o comprando. Interfaces que cargan rápido, funcionan en el teléfono y se comportan igual en los navegadores que utilizan sus clientes.

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. Dirección del desarrollo integral de mapas GIS — cartografía integral, de la base de datos al navegador

Dirección del desarrollo integral de mapas GIS Póngase en contacto →

Por qué importa

La primera decisión es dónde se realiza el trabajo. El documento de arquitectura del W3C describe un espectro entre un cliente que lo hace todo y un servidor que lo hace todo, y observa que la mayoría de los diseños combina ambos. La posición de una aplicación depende, en sus términos, de la potencia del cliente, la potencia y carga del servidor y el ancho de banda entre ellos. Expone las contrapartidas de ambas opciones: personalizar en el servidor reduce la eficacia de la caché; cuando la cantidad de clientes posibles no está acotada, llevar el cómputo a ellos escala mejor. Las opciones habituales son páginas generadas una vez, renderizadas en el servidor por solicitud, compuestas en el navegador y mezclas de todo ello. Una página igual para todos sale más barata si se genera una vez y se guarda en caché. Una que muestra los datos de un cliente se renderiza en algún lugar por solicitud; elegir el navegador hace que cada lector descargue el código que lo hace.

La segunda decisión es cuánto confiar en él. La respuesta práctica es que el servidor no confía en nada de ello. Quien maneja el navegador puede leer, modificar y reproducir todo lo entregado por el frontend. Un botón oculto o un formulario que impide el envío ayudan a un usuario honrado y son un pequeño rodeo para los demás. Por eso toda comprobación importante de permisos y validación se repite en el servidor. Lo mismo ocurre con los secretos: una clave API que llega al paquete es pública, se llame como se llame su variable. Content Security Policy limita lo que puede hacer un script inyectado; su especificación la presenta como defensa en profundidad, nunca como sustituto de validar entradas y codificar salidas cuidadosamente.

En el frontend también se gana o pierde la accesibilidad. WCAG 2.2, Recomendación del W3C publicada en diciembre de 2024, fija criterios de conformidad comprobables. Muchos dependen del frontend: si un control se puede alcanzar y manejar con teclado y si expone un nombre y un rol a tecnologías de asistencia. Los elementos HTML nativos ya cumplen varios de esos requisitos; en un widget personalizado hay que implementar a mano el comportamiento necesario para cada uno.

Por último, un frontend que sabe cómo guarda las cosas la base de datos debe cambiar cada vez que esta cambia. Si se limita a presentar, cada lado puede avanzar a su ritmo.

Cómo se aplica en la práctica

En una plataforma se decide por partes dónde renderiza el frontend. Su frontend Next.js 16 elige un modo de renderizado para cada parte de la aplicación. Las páginas de marketing y públicas se generan estáticamente; las partes interactivas se renderizan en el navegador. En el panel para usuarios autenticados y en los directorios grandes, el patrón N+1 convertía una página en una cascada de viajes de ida y vuelta. Allí, una fábrica llamada `createPrefetchedServerPage` obtiene en el servidor los datos de la página y los entrega ya cargados al cliente. Cada solicitud de datos tiene documentada una estrategia de invalidación de caché. Una página migrada a la fábrica carga en un viaje de ida y vuelta, y la siguiente en migrar hereda ese comportamiento. La migración sigue en curso: el resto de la aplicación autenticada se renderiza en el cliente y espera su turno.

La contribución arquitectónica de la fundadora en NextMariner, una plataforma marítima, da a su frontend una sola tarea. El frontend Next.js se ocupa de presentación e interactividad. Una API en Go con responsabilidades deliberadamente limitadas, construida con Huma sobre Fiber, enruta y valida solicitudes, aplica filtros de seguridad y orquesta los procesos, mientras la lógica de negocio vive en funciones PostgreSQL. El diseño sitúa el cambio de una regla de negocio en una capa, SQL, en lugar de exigir que la misma regla esté en las tres. Las fronteras se escribieron y después se hicieron cumplir en revisiones de código. El resultado documentado es separar las responsabilidades de presentación, tratamiento de solicitudes y reglas de negocio.

El trabajo de endurecimiento de la aplicación parte de que el cliente es hostil. En el frontend, el middleware proxy de Next.js en `proxy.ts` fija una Content Security Policy con un nonce generado por solicitud y `strict-dynamic`, además de HSTS y cookies SameSite. La frontera real está en el servidor. Los derechos de acceso, que abarcan nivel, rol y alcance por organización y buque, se vuelven a comprobar en el servidor en cada solicitud. Las barreras del frontend solo cuentan como experiencia de usuario: dan forma a lo visible y el servidor decide qué se permite hacer.

Dónde surgen los problemas

Los fallos del frontend suelen compartir causa: se decidió en el navegador algo que correspondía al servidor, o al revés.

Una clave API aparece en el JavaScript que cualquiera puede descargar. En Next.js, una variable de entorno solo llega al paquete del navegador si su nombre empieza por `NEXT_PUBLIC_` ; las demás leídas desde código cliente se sustituyen por una cadena vacía. Si, tras recibir una respuesta de acceso no autorizado, se cambia rápidamente el nombre de la variable para que funcione, el secreto queda insertado en un archivo público. Mantenga en el servidor el código que necesita secretos y marque su módulo con el paquete `server-only` , para que importarlo desde un Client Component haga fallar la compilación. Rote cualquier clave ya distribuida.

```
import "server-only";
export async function getInvoices(customerId: string) {
    const url = `${process.env.BILLING_URL}/customers/${customerId}/invoices`;
    const headers = { Authorization: `Bearer ${process.env.BILLING_KEY}` };
    const res = await fetch(url, { headers });
    return res.json();
}
```

Un usuario alcanza registros que la interfaz le ocultaba. La comprobación de permisos estaba en el frontend, como protección de ruta, entrada de menú oculta o botón desactivado. Quien lee el tráfico puede llamar directamente a la API con otros parámetros. Compruebe cada derecho de acceso en el servidor en cada solicitud, según el usuario autenticado y el registro concreto. Trate las comprobaciones del frontend como comodidad para usuarios honrados.

La página se completa entre sucesivos indicadores de carga. Cada componente obtiene sus datos después de renderizarse, así que la solicitud de un hijo espera a la del padre. La página llega como cascada de viajes de ida y vuelta. Obtenga juntos los datos de una página cerca de su origen, en el servidor cuando lo permita el marco, y páselos a los componentes inferiores.

Los usuarios de teclado y lector de pantalla se atascan en un control propio. Un `div` pulsable recibe el comportamiento de ratón de un botón y nada más: no toma foco, ignora Intro y Espacio y no comunica su naturaleza a tecnologías de asistencia. WCAG 2.2 pide en nivel A que toda funcionalidad se pueda manejar

con teclado, criterio 2.1.1, y que cada componente exponga nombre y rol, 4.1.2. Use `button`, `a`, `input` y `label` por lo que son, y pruebe cada pantalla nueva solo con teclado antes de publicarla.

Lecturas adicionales

- Architecture of the World Wide Web, Volume One — La Recomendación del W3C sobre agentes de usuario y reparto del trabajo entre cliente y servidor.
- Estándar HTML — El estándar vivo de WHATWG para los elementos y API de navegador que forman un frontend web.
- Pautas de accesibilidad para el contenido web (WCAG) 2.2 — La Recomendación del W3C con criterios comprobables para una interfaz accesible.
- Content Security Policy Level 3 — El borrador de trabajo del W3C que define nonces y `strict-dynamic`.
- Next.js: componentes de servidor y cliente — Cómo reparte un marco el código del frontend entre servidor y navegador.

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 frontend?» 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/frontend/