

El Frontend en una aplicación web sirve para realizar la interfaz de un sitio web, desde su estructura hasta los estilos, como pueden ser la definición de los colores, texturas, tipografías, secciones, entre otros. Su uso es determinante para que el usuario tenga una buena experiencia dentro del sitio o aplicación.

PARTES DEL FRONTEND:

1-Estructuras de navegación. Este elemento se refiere al orden en que se organizan las diferentes páginas de un sitio web y a los componentes que se vinculan entre sí para realizar diferentes funciones dentro del sitio.

2-Layout. También nombrado como diseño de página, se refiere a todos los componentes de la página web, por ejemplo: ubicación del menú, botones, footer; todo lo necesario para que un sitio sea útil y fácil de navegar.

3-Contenido web. Todo aquello que brinde información relevante o interesante para los usuarios. Es importante destacar que el contenido no tiene que ser necesariamente texto, puede incluir sonido o materiales interactivos.

4-Imágenes. Todos los recursos visuales ayudan a aumentar el interés de los usuarios. Esto también puede incluir videos, animaciones, mapas, gráficas, infografías, GIFs, ilustraciones, diagramas, etc.

5-Logotipo. Para que un sitio web tenga mayor identidad es vital que contenga el logotipo que represente a la marca o empresa.

6-Diseño gráfico. Este elemento engloba todo lo relacionado con cómo se ve el sitio web y su apariencia: colores, formas, tipografías, tamaños, etc.

CSS en frontend:

CSS frontend se refiere a una colección de enfoques y convenciones estructurados para escribir estilos y códigos CSS mantenibles, escalables y bien organizados dentro del desarrollo frontend. CSS (Hojas de estilo en cascada) es un lenguaje de hojas de estilo que se utiliza para describir y controlar la apariencia de las interfaces de usuario de aplicaciones web y móviles. A medida que los proyectos crecen, CSS se vuelve complejo, lo que lo hace propenso a sufrir numerosas deficiencias, incluidas guerras de especificidad, duplicación de código y confusión para los desarrolladores. Al aprovechar diferentes metodologías, se establecen estándares y técnicas para crear código CSS modular, reutilizable y fácil de leer, promoviendo un proceso de desarrollo más eficiente.

Las metodologías CSS de frontend populares incluyen BEM (Bloque, Elemento, Modificador), SMACSS (Arquitectura escalable y modular para CSS), OOCSS (CSS orientado a objetos), ITCSS (CSS de triángulo invertido) y Diseño atómico. Estas metodologías se dirigen a puntos débiles específicos dentro de CSS y ofrecen soluciones que mejoran la calidad, la coherencia y la mantenibilidad del código.

SMACSS, o Arquitectura modular y escalable para CSS, es una metodología arquitectónica de CSS que fomenta la categorización de reglas CSS en cinco tipos distintos: base, diseño, módulo, estado y tema. Esta categorización optimiza la organización del código, lo que facilita su navegación y mantenimiento. SMACSS enfatiza la separación de preocupaciones, promoviendo un enfoque modular y estructurado para administrar el código y al mismo tiempo ser lo suficientemente flexible para adaptarse a los requisitos únicos del proyecto.

OOCSS, o CSS orientado a objetos, es una metodología que aplica los principios de la programación orientada a objetos a CSS. Su objetivo es mejorar la reutilización, el mantenimiento y el rendimiento del código fomentando la separación de responsabilidades en los archivos CSS. Con los dos principios fundamentales de OOCSS (separación de estructura y apariencia y separación de contenedores y contenido), reduce la hinchazón, la redundancia y la complejidad en el código, lo que aumenta la eficiencia y la productividad durante el proceso de diseño del frontend.

HTML en frontend:

HTML (Hypertext Markup Language) en el frontend de una aplicación web se utiliza para estructurar y organizar el contenido de la página. Define la jerarquía y disposición de los elementos, como encabezados, párrafos, imágenes, enlaces, formularios, entre otros. HTML actúa como el "esqueleto" de la página, proporcionando la base sobre la cual se aplican estilos con CSS y se añade interactividad mediante JavaScript. En resumen, HTML es fundamental para la representación y organización de la información en la interfaz de usuario de una aplicación web.

JAVASCRIPT en frontend:

En el contexto de una aplicación web, JavaScript se utiliza principalmente para agregar interactividad y dinamismo al frontend. El frontend de una aplicación web es la parte que los usuarios interactúan directamente a través de sus navegadores. Aquí hay algunas funciones comunes de JavaScript en el frontend de una aplicación web:

1-

Manipulación del DOM (Document Object Model):

JavaScript se utiliza para acceder y manipular elementos HTML, cambiar su contenido, estilos, atributos, etc.

2-

Manejo de eventos:

JavaScript permite gestionar eventos del usuario, como clics, teclas presionadas, cambios en formularios, etc.

3-

AJAX (Asynchronous JavaScript and XML):

Se utiliza para realizar solicitudes asíncronas al servidor y actualizar partes específicas de la página sin recargarla por completo.

4-

Manipulación de datos:

JavaScript permite manipular datos en el cliente, como validar formularios, realizar cálculos, y gestionar el almacenamiento local.

5-

Animaciones y transiciones:

Puedes utilizar JavaScript para crear animaciones y transiciones en elementos HTML y CSS.

6-

Gestión de cookies y almacenamiento local:

Puedes utilizar JavaScript para trabajar con cookies del navegador y almacenamiento local para mantener información en el cliente.

7-

Frameworks y bibliotecas:

JavaScript se utiliza con diversos frameworks y bibliotecas como React, Angular o Vue.js para facilitar el desarrollo de aplicaciones web más complejas y estructuradas.

El Backend se refiere a la parte del sistema que se encarga de procesar la lógica de la aplicación, gestionar la base de datos y realizar operaciones que no pueden ser realizadas directamente por el frontend (la parte que los usuarios ven e interactúan en sus navegadores).

El backend es responsable de manejar las solicitudes del cliente (a través del frontend) y proporcionar las respuestas adecuadas.

NodeJs en backend:

Node.js es un entorno de ejecución de JavaScript del lado del servidor que permite ejecutar código JavaScript en el servidor. Es especialmente popular para desarrollar el backend de aplicaciones web debido a su naturaleza asincrónica y su capacidad para manejar una gran cantidad de conexiones simultáneas. Aca abajo un par de ejemplos clave:

1- JavaScript en el servidor:

Node.js permite a los desarrolladores utilizar JavaScript tanto en el frontend como en el backend, lo que simplifica el desarrollo al usar el mismo lenguaje en ambos lados.

2- Event-Driven y Asincrónico:

Node.js utiliza un modelo de E/S no bloqueante y basado en eventos. Esto significa que puede manejar múltiples operaciones de entrada/salida simultáneamente sin bloquear el hilo de ejecución, lo que mejora la escalabilidad y la eficiencia en aplicaciones con muchas conexiones concurrentes.

3- Paquete npm:

Node.js viene con npm (Node Package Manager), que es un sistema de gestión de paquetes que permite a los desarrolladores instalar, compartir y gestionar dependencias de proyectos fácilmente.

4- Frameworks y bibliotecas:

Hay varios frameworks y bibliotecas populares en Node.js para el desarrollo del backend, como Express.js. Express es un framework minimalista que facilita la creación de servidores web y la gestión de rutas y middleware.

5- Desarrollo rápido:

Node.js es conocido por su velocidad y eficiencia en el desarrollo. Los desarrolladores pueden crear rápidamente prototipos y construir aplicaciones web de manera eficiente debido a su naturaleza orientada a eventos.

6- Manejo de solicitudes HTTP:

Node.js puede manejar solicitudes HTTP y construir servidores web fácilmente. Esto lo hace ideal para construir API RESTful o servidores que sirven páginas web.

7- Escalabilidad:

Debido a su naturaleza asincrónica y no bloqueante, Node.js es escalable y puede manejar una gran cantidad de conexiones simultáneas, lo que es beneficioso para aplicaciones web en tiempo real y de alto rendimiento.

WebServer (EndPoint):

Un "endpoint" en el contexto del backend de una aplicación web generalmente se refiere a un punto final de una API (Interfaz de Programación de Aplicaciones). Los endpoints son URLs específicas a las cuales se pueden enviar solicitudes HTTP para realizar operaciones específicas en el servidor.

En un servidor web backend, los endpoints son puntos de entrada que manejan las solicitudes HTTP y ejecutan la lógica de la aplicación correspondiente. Cada endpoint está asociado con una ruta específica y puede aceptar diferentes tipos de solicitudes HTTP, como GET, POST, PUT, DELETE, etc.

Handlebar:

Handlebars es un motor de plantillas que te permite definir las vistas de tu aplicación de una manera más dinámica. Puedes utilizar Handlebars en el backend para renderizar dinámicamente HTML y enviarlo al frontend como respuesta a una solicitud.

Websocket:

Los WebSockets son una tecnología que permite establecer una conexión bidireccional entre el cliente (generalmente un navegador web) y el servidor. Esta conexión persistente facilita la comunicación en tiempo real y es especialmente útil para aplicaciones web que requieren actualizaciones instantáneas, como chats en vivo, notificaciones en tiempo real, juegos multijugador, entre otros.

En el backend de una aplicación web, puedes utilizar bibliotecas específicas para manejar WebSockets. A continuación, te mostraré un ejemplo utilizando Node.js y la biblioteca ws para manejar WebSockets.

Base de Datos:

En el backend de una aplicación web, la base de datos es un componente fundamental que se utiliza para almacenar y gestionar datos de manera persistente. Las bases de datos pueden ser de diferentes tipos, y la elección dependerá de las necesidades específicas de tu aplicación. Aquí hay algunas opciones comunes de bases de datos en el backend:

1-

Bases de datos relacionales (SQL):

Ejemplos: MySQL, PostgreSQL, SQLite, Microsoft SQL Server, Oracle.

Son bases de datos estructuradas que utilizan un lenguaje de consulta estructurado (SQL) para definir y manipular los datos. Son ideales para aplicaciones que requieren relaciones complejas entre diferentes conjuntos de datos.

2-

Bases de datos NoSQL:

Ejemplos: MongoDB, CouchDB, Cassandra, Redis.

Son bases de datos diseñadas para manejar grandes cantidades de datos no estructurados o semiestructurados. No siguen un modelo de tabla y son ideales para escenarios en los que la escalabilidad y la flexibilidad son cruciales.

3-

Bases de datos en memoria:

Ejemplos: Redis, Memcached.

Almacenan datos en la memoria principal del servidor en lugar de en el disco, lo que proporciona tiempos de acceso extremadamente rápidos. Son ideales para almacenar datos temporales o para cachés.

4-

Bases de datos orientadas a grafos:

Ejemplos: Neo4j, Amazon Neptune.

Están diseñadas para almacenar y consultar datos relacionales mediante grafos. Son ideales para aplicaciones que necesitan modelar y consultar relaciones complejas entre entidades.

5-

Bases de datos embebidas:

Ejemplos: SQLite, H2.

Son bases de datos que se integran directamente en la aplicación y no requieren un servidor de base de datos separado. Son útiles en aplicaciones más pequeñas o en situaciones donde la portabilidad es una consideración importante.

Api Externa:

Cuando hablamos de integrar una API externa en el backend de una aplicación web, nos referimos a hacer solicitudes a una interfaz de programación de aplicaciones (API) proporcionada por un servicio externo para obtener datos o realizar operaciones específicas. Esto es común en situaciones donde tu aplicación necesita acceder a servicios externos, como redes sociales, servicios de pago, proveedores de mapas, entre otros.

Interacciones HTTP:

En una aplicación web, las interacciones HTTP juegan un papel crucial en la comunicación entre el frontend (la interfaz de usuario que los usuarios ven en sus navegadores) y el backend (el servidor que maneja la lógica de la aplicación y la gestión de datos). Estas interacciones suelen involucrar solicitudes y respuestas HTTP para realizar operaciones como obtener datos, enviar datos, actualizar recursos, y más. Aquí hay algunas interacciones HTTP comunes en una aplicación web:

1-

Solicitudes HTTP:

GET: Se utiliza para solicitar datos del servidor. Por ejemplo, cuando un usuario carga una página web, el navegador realiza una solicitud GET para obtener el contenido de esa página.

POST: Se utiliza para enviar datos al servidor. Por ejemplo, cuando un usuario envía un formulario, el navegador realiza una solicitud POST para enviar los datos del formulario al servidor.

PUT y PATCH: Se utilizan para actualizar recursos en el servidor. PUT se usa típicamente para actualizar el recurso completo, mientras que PATCH se utiliza para realizar actualizaciones parciales.

DELETE: Se utiliza para eliminar recursos en el servidor.

2-

Respuestas HTTP:

Códigos de estado: Las respuestas HTTP incluyen códigos de estado que indican el resultado de la operación. Por ejemplo, el código 200 OK indica que la solicitud se ha procesado correctamente, mientras que 404 Not Found indica que el recurso solicitado no se encontró.

Cuerpo de la respuesta: Puede incluir datos devueltos por el servidor, como en una respuesta de una solicitud GET.

3-

Autenticación y Autorización:

Las aplicaciones web a menudo requieren autenticación y autorización. Las solicitudes pueden incluir encabezados para enviar tokens de acceso, nombres de usuario y contraseñas para verificar la identidad del usuario.

4-

Cookies y Sesiones:

Las aplicaciones web a menudo utilizan cookies y sesiones para realizar un seguimiento del estado del usuario. Las cookies se envían con cada solicitud HTTP y pueden contener información como tokens de sesión.

Interacciones de WebSocket:

Las interacciones de WebSocket en una aplicación web proporcionan una comunicación bidireccional en tiempo real entre el frontend y el backend. A diferencia de las solicitudes HTTP tradicionales, que son unidireccionales, WebSocket permite una conexión persistente donde ambas partes pueden enviar mensajes en cualquier momento. Esto es especialmente útil para aplicaciones que requieren actualizaciones en tiempo real, como chats, notificaciones en vivo, juegos multijugador, entre otros.

Aquí hay una visión general de cómo funcionan las interacciones de WebSocket en una aplicación web:

1-

Establecimiento de la conexión:

El frontend establece una conexión WebSocket con el backend mediante la creación de un objeto WebSocket en JavaScript. La URL del WebSocket generalmente apunta al endpoint en el servidor que manejará las conexiones WebSocket.

2-

Manejo de eventos:

En el frontend, puedes manejar eventos para diferentes estados de la conexión WebSocket, como la apertura, cierre, errores y mensajes recibidos.

3-

Envío de mensajes:

El frontend puede enviar mensajes al servidor utilizando el método send.

4-

Manejo en el backend:

En el backend, debes implementar un servidor que pueda manejar conexiones WebSocket. La implementación puede depender del lenguaje y el marco que estés utilizando.

Por ejemplo, en Node.js con el paquete ws:

5-

Seguridad:

Es importante implementar medidas de seguridad, especialmente si estás transmitiendo información sensible. Puedes utilizar conexiones seguras mediante wss:// en lugar de ws:// y considerar la autenticación y autorización de usuarios.

Interacciones de SQL:

Las interacciones de SQL en una aplicación web generalmente se centran en realizar operaciones de base de datos para almacenar, recuperar, actualizar o eliminar datos. Estas operaciones se realizan a través de consultas SQL (Structured Query Language) que interactúan con la base de datos del backend. Aquí hay algunas interacciones comunes de SQL en una aplicación web:

1-

Consulta de datos (SELECT):

La operación SELECT se utiliza para recuperar datos de una base de datos. En una aplicación web, esto podría involucrar la obtención de información para mostrarla en la interfaz de usuario.

2-

Inserción de datos (INSERT):

La operación INSERT se utiliza para agregar nuevos registros a una tabla de la base de datos. En una aplicación web, esto podría ocurrir cuando un usuario crea una cuenta o envía información a través de un formulario.

3-

Actualización de datos (UPDATE):

La operación UPDATE se utiliza para modificar registros existentes en una tabla. En una aplicación web, esto podría ser necesario cuando un usuario actualiza su perfil o realiza cambios en la información.

4-

Eliminación de datos (DELETE):

La operación DELETE se utiliza para eliminar registros de una tabla. En una aplicación web, esto podría ocurrir cuando un usuario elimina su cuenta o cuando se realiza alguna acción que requiere la eliminación de datos.

5-

Transacciones:

Las transacciones en SQL se utilizan para agrupar varias operaciones en una única unidad. Esto es útil para garantizar la consistencia de los datos y para revertir los cambios en caso de un error. Por ejemplo, en una aplicación web de comercio electrónico, una transacción podría involucrar la actualización del inventario y la creación de un registro de pedido al mismo tiempo.

6-

Procedimientos almacenados y funciones:

Puedes utilizar procedimientos almacenados y funciones en SQL para encapsular lógica de negocio en el servidor de la base de datos. Estos son útiles cuando hay operaciones complejas que deben realizarse de manera consistente.