

Agencia de
Aprendizaje
a lo largo
de la vida

FULL STACK FRONTEND

Clase 25

Node 1

Les damos la bienvenida

Vamos a comenzar a grabar la clase

Clase 24

- JOIN: Inner, Left, Right.
- Funciones de agregación GROUP BY, HAVING.
- Funciones escalares: Caracteres o cadena, Conversion o cadena,
- Conversion, Fecha y Tiempo,
- Matemáticas.
- Subconsultas

Clase 25

Fundamentos de Node

- Repaso anterior
- ¿Que es el Backend? Introducción a Node. Entorno

Clase 26

Node – Primeros Pasos

¿Qué es el Frontend?

Es todo lo que pasa del lado del cliente (en el navegador).

Está compuesto por archivos estáticos en los lenguajes que el navegador puede interpretar. Estos lenguajes son HTML para la estructura de información, CSS para los estilos y JavaScript para la interacción dentro nuestro sitio web.

El desarrollador de front-end se encarga de implementar todo lo relacionado con la parte visible, lo que "toca" el usuario cuando navega por la web.

Elemento claves del frontend

HTML (HyperText Markup Language): Es el lenguaje de marcado utilizado para estructurar el contenido en la web. Define elementos como títulos, párrafos, enlaces, imágenes, y otros contenidos que se mostrarán en la página web.

CSS (Cascading Style Sheets): Es el lenguaje utilizado para describir la presentación de un documento HTML. Controla el aspecto visual, incluyendo el diseño, los colores, las fuentes, y la disposición de los elementos en la página.

JavaScript: Es el lenguaje de programación utilizado para agregar interactividad a la página web. Permite crear experiencias dinámicas, como formularios interactivos, animaciones, manejo de eventos (clics, desplazamiento, etc.), y comunicación con servidores (por ejemplo, mediante AJAX o Fetch API).

Frameworks y Librerías: Herramientas que simplifican y estructuran el desarrollo frontend. Ejemplos comunes incluyen React, Angular, Vue.js, y Svelte. Estos frameworks/librerías ayudan a gestionar el estado de la aplicación, el enrutamiento, y la manipulación del DOM de manera más eficiente.



¿Qué es el Backend?

Es todo lo que pasa del lado del servidor.

El **backend** es la parte del desarrollo web que se encarga de que toda la lógica de una página web funcione. Se trata del conjunto de acciones que pasan en una web pero que no vemos como, por ejemplo, la consulta de datos a una BBDD y posterior envío al cliente.

Algunas de las funciones que se gestionan en la parte del back-end son:

- El desarrollo de funciones que simplifiquen el proceso de desarrollo.
- Acciones de lógica.
- Conexión con bases de datos.
- Uso de librerías del servidor web (por ejemplo, para implementar temas de caché o para comprimir las imágenes de la web).

Elemento claves del backend

Servidor: La máquina física o virtual que ejecuta el software del backend. Puede ser un servidor local, un servidor en la nube o un servidor dedicado.

Aplicación del Servidor: El software que gestiona las solicitudes de los clientes (generalmente el frontend) y proporciona las respuestas adecuadas. Este software puede estar desarrollado en varios lenguajes de programación, **como JavaScript (Node.js)**, Python (Django, Flask), Ruby (Ruby on Rails), Java (Spring), PHP (Laravel), entre otros.

API (Application Programming Interface): Conjunto de reglas y definiciones que permiten la comunicación entre diferentes partes de la aplicación o entre diferentes aplicaciones. Las APIs RESTful y GraphQL son ejemplos comunes de cómo el backend expone funcionalidades y datos al frontend.

Base de Datos: Sistema de gestión de bases de datos que almacena y recupera los datos necesarios para la aplicación. Puede ser una base de datos relacional (como **MySQL**, PostgreSQL) o una base de datos NoSQL (como MongoDB, Cassandra).

Lógica de Negocio: Conjunto de reglas y procesos que definen cómo los datos se crean, almacenan y manipulan para cumplir con los requisitos específicos de la aplicación. Esto puede incluir desde la validación de datos hasta cálculos complejos y procesamiento de datos.

Elemento claves del backend

Autenticación y Autorización: Mecanismos para verificar la identidad de los usuarios (autenticación) y determinar qué acciones pueden realizar (autorización). Herramientas y estándares como OAuth, JWT (JSON Web Tokens), y OpenID Connect son comunes en el backend.

Gestión de Sesiones: Técnica para mantener el estado de la interacción del usuario con la aplicación a lo largo de múltiples solicitudes HTTP. Esto puede incluir cookies, tokens, y almacenamiento de sesiones.

Servicios y Microservicios: En arquitecturas modernas, el backend puede estar compuesto por múltiples servicios pequeños y especializados (microservicios), que se comunican entre sí para realizar tareas específicas.

Escalabilidad y Rendimiento: Estrategias y técnicas para asegurar que el backend pueda manejar grandes cantidades de tráfico y operaciones de manera eficiente. Esto incluye balanceo de carga, caché, y escalado horizontal y vertical.

Seguridad: Prácticas para proteger los datos y las operaciones del backend contra accesos no autorizados, ataques de inyección, y otras amenazas de seguridad.

Flujo cliente/servidor

El flujo cliente-servidor describe cómo se comunican los componentes frontend (cliente) y backend (servidor) en una aplicación para procesar y responder a las solicitudes del usuario. A continuación se detalla un flujo típico en una arquitectura cliente-servidor:

Cliente

Envía una petición al servidor y se queda esperando por una respuesta.
Su tiempo de vida es finito una vez que son servidas sus solicitudes, termina el trabajo.
Un cliente accede a un servidor y recupera servicios especiales o datos de él.
Es tarea del cliente estandarizar las solicitudes, transmitirlos al servidor y procesar los datos obtenidos para que puedan visualizarse en un dispositivo de salida como una pantalla.

Servidor

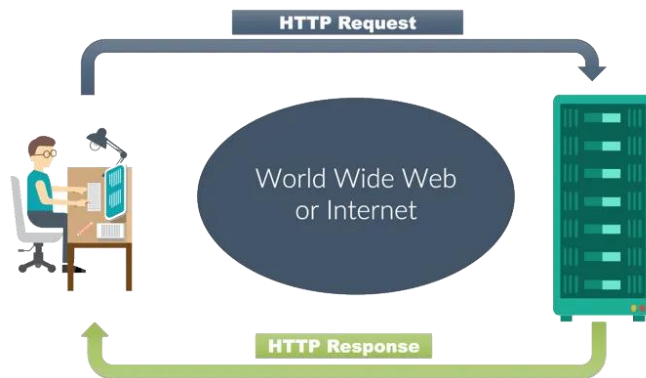
Es un programa que ofrece un servicio que se puede obtener en una red.
Acepta la petición desde la red, realiza el servicio y devuelve el resultado al solicitante.
El servidor comienza su ejecución antes de comenzar la interacción con el cliente.
Su tiempo de vida o de interacción es “interminable”, una vez comienza a correr, se queda esperando las solicitudes que pudieran llegar desde los diversos clientes.

Request / Response

El flujo cliente-servidor describe cómo se comunican los componentes frontend (cliente) y backend (servidor) en una aplicación para procesar y responder a las solicitudes del usuario.

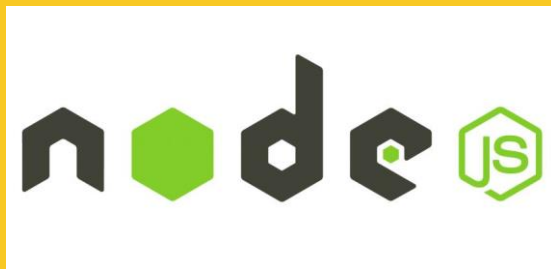
Una **request** es una petición enviada desde el cliente (generalmente el navegador) al servidor para solicitar información o realizar una acción específica.

Una **response** es la respuesta del servidor a una solicitud del cliente.



¿Qué es Node js?

Es un entorno de ejecución (NO es un lenguaje ni un framework) de JavaScript construido sobre el motor V8 de Google Chrome. Fue diseñado para construir aplicaciones de red escalables y de alto rendimiento, permitiendo a los desarrolladores ejecutar JavaScript en el servidor, fuera del navegador.



Usos comunes de Node

- **Desarrollo de Servidores Web y APIs RESTful:** es ideal para construir servidores web y APIs RESTful gracias a su capacidad para manejar múltiples solicitudes de manera eficiente y no bloqueante.
- **Aplicaciones en Tiempo Real:** es excelente para aplicaciones en tiempo real que requieren comunicación bidireccional entre el cliente y el servidor, como chats, aplicaciones de colaboración en línea y juegos multijugador.
- **Aplicaciones de Microservicios:** es una opción popular para construir arquitecturas de microservicios debido a su capacidad para manejar múltiples servicios pequeños e independientes que pueden comunicarse entre sí.
- **Desarrollo de Aplicaciones Backend para IoT (Internet of Things):** es adecuado para manejar la comunicación y procesamiento de datos de dispositivos IoT debido a su naturaleza asíncrona y eficiente.
- **Tareas de DevOps:** se utiliza para escribir herramientas y scripts de DevOps que ayudan a gestionar la infraestructura y el despliegue de aplicaciones.

Instalación

Ingresamos a <https://nodejs.org/en/> y descargamos la versión LTS (long term support) ya que es la más reciente y con soporte oficial recomendada para proyectos “reales” o productivos.



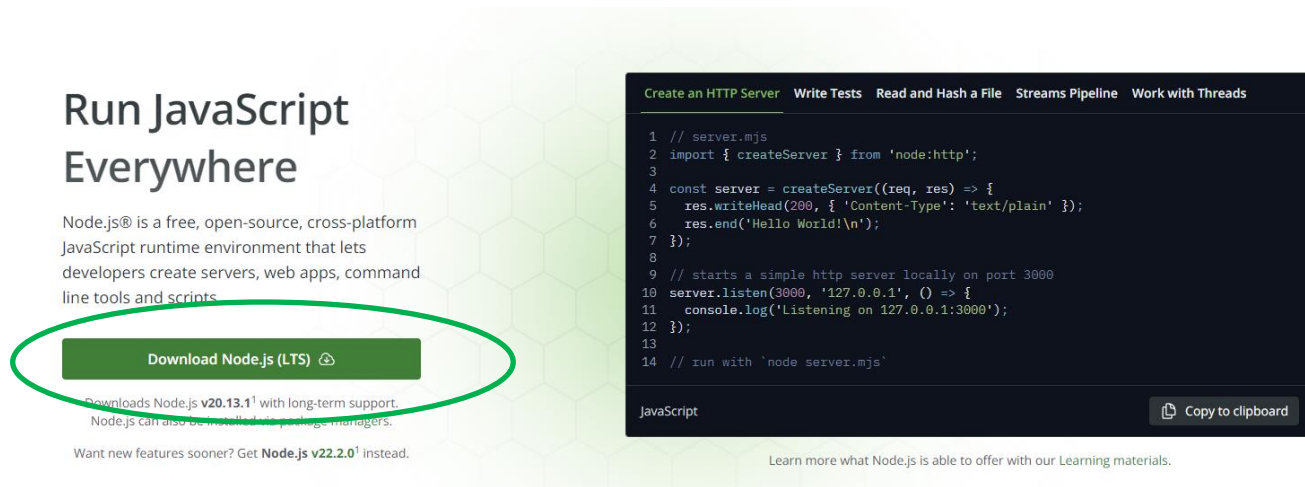
The screenshot shows the Node.js website with the heading "Run JavaScript Everywhere". Below the heading, it states: "Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts." A green oval highlights the "Download Node.js (LTS)" button. Below the button, it says: "Downloads Node.js v20.13.1 with long-term support. Node.js can also be installed via package managers." At the bottom, it says: "Want new features sooner? Get Node.js v22.2.0 instead." On the right side, there is a code editor with the following code:

```
1 // server.mjs
2 import { createServer } from 'node:http';
3
4 const server = createServer((req, res) => {
5   res.writeHead(200, { 'Content-Type': 'text/plain' });
6   res.end('Hello World!\n');
7 });
8
9 // starts a simple http server locally on port 3000
10 server.listen(3000, '127.0.0.1', () => {
11   console.log('Listening on 127.0.0.1:3000');
12 });
13
14 // run with `node server.mjs`
```

Below the code editor, it says "JavaScript" and "Copy to clipboard". At the bottom right, it says "Learn more what Node.js is able to offer with our Learning materials."

Instalación

Ingresamos a <https://nodejs.org/en/> y descargamos la versión LTS (long term support) ya que es la más reciente y con soporte oficial recomendada para proyectos “reales” o productivos.



The screenshot shows the Node.js website with the heading "Run JavaScript Everywhere". Below the heading, it states: "Node.js® is a free, open-source, cross-platform JavaScript runtime environment that lets developers create servers, web apps, command line tools and scripts." A green button labeled "Download Node.js (LTS)" is circled in green. Below the button, it says: "Downloads Node.js v20.13.1 with long-term support. Node.js can also be installed via package managers." Further down, it mentions: "Want new features sooner? Get Node.js v22.2.0 instead." On the right side of the screenshot, there is a code editor with the following code:

```
1 // server.mjs
2 import { createServer } from 'node:http';
3
4 const server = createServer((req, res) => {
5   res.writeHead(200, { 'Content-Type': 'text/plain' });
6   res.end('Hello World!\n');
7 });
8
9 // starts a simple http server locally on port 3000
10 server.listen(3000, '127.0.0.1', () => {
11   console.log('Listening on 127.0.0.1:3000');
12 });
13
14 // run with `node server.mjs`
```

At the bottom of the code editor, it says "JavaScript" and "Copy to clipboard".

Primeros pasos con NodeJS

Ahora que tenemos NODE instalado en nuestra PC podemos trabajar con él del mismo modo que lo hacíamos con Javascript.

En esta ocasión para ejecutar nuestro código en lugar de usar la consola del navegador, vamos a usar la terminal de VS CODE o de nuestra PC



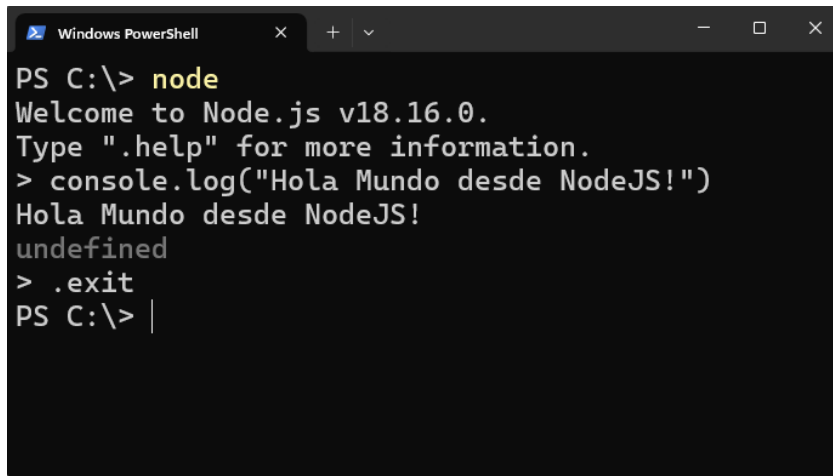
```
index.js
1 console.log("Hola Mundo");
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL

```
● >> node 02:26 node index.js
  Hola Mundo
○ >> node 02:26
```

Primeros pasos con NodeJS

También podemos escribir y ejecutar nuestro código NODE a través de la consola mediante el comando **node**. Para salir de este modo, usamos el comando **.exit**



```
Windows PowerShell
PS C:\> node
Welcome to Node.js v18.16.0.
Type ".help" for more information.
> console.log("Hola Mundo desde NodeJS!")
Hola Mundo desde NodeJS!
undefined
> .exit
PS C:\> |
```


No te olvides de dar el presente

Recordá:

- **Revisar la Cartelera de Novedades.**
- **Hacer tus consultas en el Foro.**
- **Realizar los Ejercicios obligatorios.**

Todo en el Aula Virtual.

Muchas gracias por tu atención.

Nos vemos pronto