



Instructivo desarrollo de aplicaciones

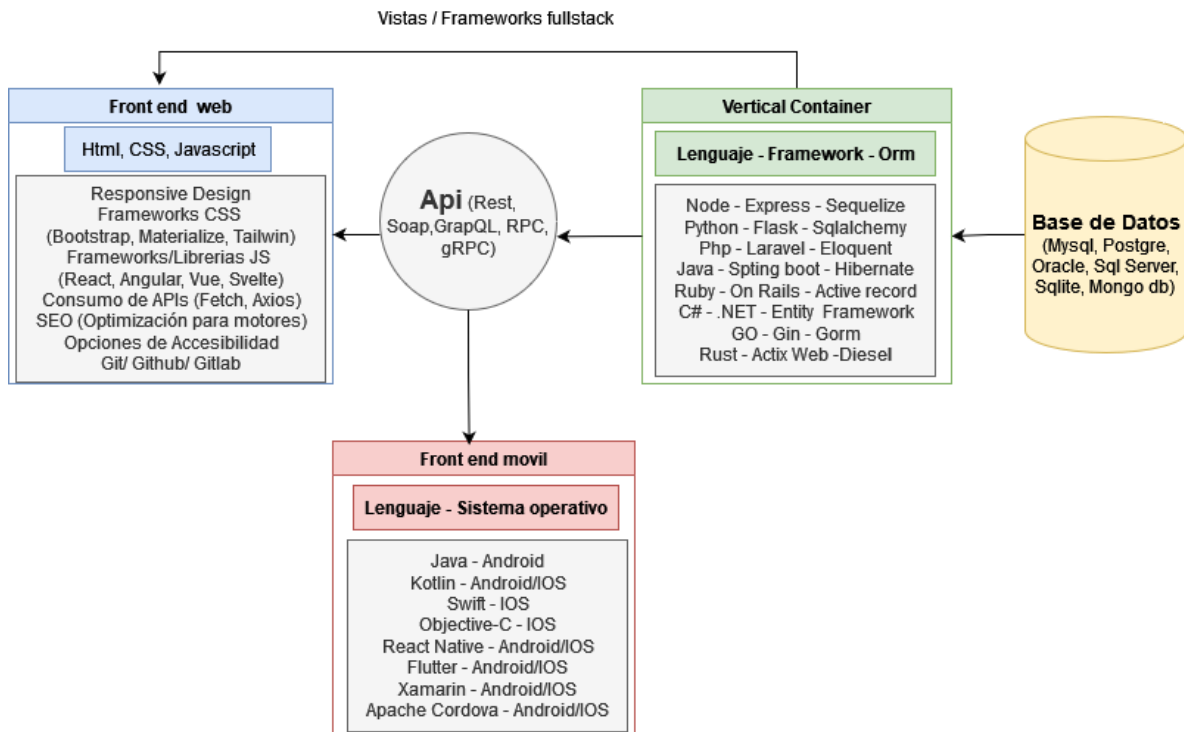


Figura 1: Arquitectura de una aplicación web / móvil

I. Definiciones

- 1. Front end:** La interfaz es la parte del sitio web con la que interactúan los usuarios y se ejecuta en el cliente (navegador web). Incluye elementos como formularios, botones, texto, imágenes, videos, gráficos, tablas y menús de navegación. Generalmente, se utilizan los lenguajes **HTML**, **CSS** y **JavaScript** para implementar la estructura, el diseño, el comportamiento y el contenido de todo lo que se visualiza en las pantallas.

- 1.1. HTML (Lenguaje de marcado de hipertexto):** estructura el contenido de una página web, usando elementos como encabezados, párrafos, enlaces, imágenes, y formularios. Ver más:

<https://developer.mozilla.org/es/docs/Web/HTML>

<https://developers.google.com/profile/badges/playlists/webdev/learn-html?hl=es-419>

<https://www.w3schools.com/html/default.asp>

- 1.2. CSS (Hojas de estilo en cascada):** controlar el diseño visual de los elementos HTML, como colores, tipografías, márgenes, y disposición en pantalla. Ver más: <https://web.dev/learn/css?hl=es>



<https://developer.mozilla.org/es/docs/Web/CSS>

<https://www.w3schools.com/css/default.asp>

1.2.1. Responsive Design: Técnica que asegura que una página web funcione correctamente en diferentes dispositivos (teléfonos, tablets, computadores). Ver más: https://developer.mozilla.org/en-US/docs/Learn/CSS/CSS_layout/Responsive_Design
<https://web.dev/learn/design>

1.2.2. Frameworks CSS: Herramientas como Bootstrap, Tailwind o Materialize que facilitan el desarrollo al ofrecer componentes y estilos predefinidos para el diseño web.

1.2.3. Bootstrap: Bootstrap es un framework multiplataforma o conjunto de herramientas de código abierto para diseño de sitios y aplicaciones web. *Wikipedia.* Ver más: <https://getbootstrap.com/>

1.2.4. Tailwind CSS: es un framework de CSS de código abierto para el diseño de páginas web. no genera una serie de clases predefinidas para elementos como botones o tablas como Bootstrap. *Wikipedia.* Ver más: <https://tailwindcss.com/>

1.2.5. Materialize: es un framework front-end basado en el diseño de Material Design, creado por Google. Este framework proporciona una serie de herramientas y componentes que permiten a los desarrolladores construir aplicaciones web modernas y responsivas de manera más eficiente y coherente. Ver más: <https://materializecss.com/>

1.2.6. Material Design: es un lenguaje de diseño desarrollado por Google, utiliza más diseños basados en cuadrículas, animaciones y transiciones receptivas, relleno y profundidad, efectos como luces y sombras. *Wikipedia.* <https://m3.material.io/>

1.3. JavaScript (JS): Lenguaje de programación que permite agregar interactividad y dinamismo a las páginas web, como validación de formularios, animaciones, o actualizaciones de contenido sin recargar la página. Ver más: <https://web.dev/learn/javascript>
<https://developer.mozilla.org/es/docs/Web/JavaScript>
<https://www.w3schools.com/js/default.asp>



1.3.1. JavaScript Frameworks/Libraries: herramientas como que ayudan a construir interfaces de usuario más complejas y manejan de forma eficiente los datos y el estado de la aplicación por ejemplo React, Vue.js, Angular o Svelte.

1.3.2. jQuery: jQuery es una biblioteca multiplataforma de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML, manipular el árbol DOM, manejar eventos, desarrollar animaciones y agregar interacción con la técnica AJAX a páginas web. *Wikipedia.* Ver más: <https://jquery.com/>

1.3.3. AngularJS: framework para aplicaciones web desarrollado en TypeScript, de código abierto, mantenido por Google, que se utiliza para crear y mantener aplicaciones web de una sola página. *Wikipedia.* Ver más: <https://angular.dev/>

1.3.4. React.js: biblioteca Javascript de código abierto diseñada para crear interfaces de usuario para facilitar el desarrollo de aplicaciones en una sola página (SPA). Es mantenido por Facebook y la comunidad de software libre. *Wikipedia.* Ver más: <https://es.react.dev/>

1.3.5. Vue.js: es un framework de JavaScript de código abierto para la construcción de interfaces de usuario y aplicaciones de una sola página (SPA). Fue creado por Evan You, y es mantenido por él y por el resto de los miembros activos del equipo central que provienen de diversas empresas como Netlify y Netguru. *Wikipedia.* Ver más: <https://vuejs.org/>

1.3.6. Svelte: es un compilador Front End gratuito y de código abierto creado por Rich Harris y mantenido por los miembros del equipo central de Svelte. *Wikipedia.* Ver más: <https://svelte.dev/>

1.3.7. Alpine.js: es una herramienta robusta y ligera para componer el comportamiento directamente en su marcado. Similar a jQuery pero para la web moderna. Compuesta por una colección de 15 atributos, 6 propiedades y 2 métodos. Ver más: <https://alpinejs.dev/>

1.4. API (Application Programming Interface): es un conjunto de reglas y protocolos que permite que dos aplicaciones diferentes se comuniquen entre sí.

Ejemplo: Cuando usas una aplicación que se conecta a un servidor para obtener datos (como maps de google), la app utiliza una API para comunicarse



con ese servidor y obtener los datos. Puede usar diferentes arquitecturas (SOAP, REST, GraphQL, RPC, etc.).

| Arquitectura | Protocolos | Formato de datos | Complejidad | Uso recomendado |
|--------------|-----------------|------------------|-------------|--|
| REST | HTTP | JSON, XML, etc. | Baja | APIs web simples, servicios distribuidos |
| SOAP | HTTP, SMTP, TCP | XML | Alta | Entornos empresariales, alta seguridad |
| GraphQL | HTTP | JSON | Media | APIs flexibles, optimización de datos |
| RPC | Varios | JSON, XML | Media | Ejecución remota de procedimientos |
| gRPC | HTTP/2 | Protocol Buffers | Alta | Microservicios, comunicación eficiente |

Figura 2: Cuadro comparativo arquitecturas para APIs.

1.5. Consumo de APIs: Técnicas para conectar la interfaz de usuario con servicios externos o bases de datos mediante APIs, utilizando tecnologías como Fetch API o Axios. Ver más:

<https://developer.mozilla.org/es/docs/Web/API/Window/fetch>

1.6. API REST (Representational State Transfer): es un conjunto de restricciones y principios para crear servicios web escalables. Tiene los siguientes principios clave:

- **Stateless:** No guarda el estado del cliente entre las solicitudes. Cada solicitud contiene toda la información necesaria.
 - **Uso de HTTP:** Utiliza los métodos HTTP estándar (GET, POST, PUT, DELETE, etc.) para realizar operaciones.
 - **Recursos:** Los recursos se identifican mediante URLs.
 - **Representación:** Los recursos pueden enviarse en diferentes formatos, como JSON o XML.
 - **Uniform Interface:** Debe tener una interfaz uniforme, es decir, las interacciones con el servicio siguen un conjunto común de reglas.
 - **Ejemplo:** Una API que te permite obtener información de productos de una tienda online utilizando **GET /productos** es una API REST.
- Ver más:



1.7. PWA: son aplicaciones que se compilan mediante tecnologías web y que se pueden instalar y ejecutar en todos los dispositivos, desde un código base. Las PWA proporcionan experiencias nativas a los usuarios en dispositivos auxiliares. *Microsoft Learn*. Ver más:

<https://learn.microsoft.com/es-es/microsoft-edge/progressive-web-apps-chromium/>
<https://web.dev/learn/pwa>

1.8. Control de versiones (Git): Gestiona cambios en el código de un proyecto de manera colaborativa, permitiendo llevar un historial y revertir cambios si es necesario. Ver mas: <https://git-scm.com/book/es/v2/Inicio---Sobre-el-Control-de-Versiones-Acerca-del-Control-de-Versiones>

1.8.1. GitHub: es un repositorio de código escrito en Ruby on Rails , que utiliza el sistema de control de versiones Git. *Wikipedia*. Ver más:

<https://github.com/> <https://docs.github.com/es/get-started/start-your-journey/hello-world> <https://www.freecodecamp.org/espanol/news/guia-para-principiantes-de-git-y-github/>
<https://www.youtube.com/watch?v=4YIHAETkPs>

1.8.2. GitLab: es una plataforma de desarrollo colaborativo basada en Git, diseñada para facilitar la gestión de proyectos, el control de versiones de código y la automatización del ciclo de vida del desarrollo de software. GitLab ofrece una serie de herramientas que permiten a los equipos de desarrollo trabajar de manera eficiente desde la planificación hasta la implementación y mantenimiento de sus aplicaciones. Ver más:

<https://about.gitlab.com/> <https://docs.gitlab.com/ee/tutorials/>
<https://www.youtube.com/watch?v=6NREQqA5wHE>

1.9. Browser DevTools: Herramientas integradas en los navegadores (como Chrome DevTools) que permiten depurar código, analizar el rendimiento de la página, y hacer pruebas de diseño. Ver más:

<https://developer.chrome.com/docs/devtools?hl=es-419>

1.10. Accesibilidad: Prácticas para asegurar que las páginas web sean accesibles para personas con discapacidad, siguiendo estándares como el WCAG. Ver más: <https://developer.mozilla.org/es/docs/Web/Accessibility>

1.11. Performance Optimization: Métodos para mejorar la velocidad y el rendimiento de una web, como minimizar archivos, cargar imágenes de manera eficiente, y usar estrategias de cacheado. Ver más:



<https://www.posizionate.com/blog/que-es-el-web-performance-optimization-elementos-y-herramientas-wpo/>

1.12. SEO (Search Engine Optimization): Técnicas para optimizar el sitio web y mejorar su posicionamiento en motores de búsqueda, como Google. Ver mas: <https://developers.google.com/search/docs/fundamentals/seo-starter-guide?hl=es>

1.13. Cross-Browser Compatibility: Prácticas para garantizar que la web funcione de manera uniforme en diferentes navegadores (Chrome, Firefox, Edge, Safari, etc.). Ver más: <https://www.freecodecamp.org/news/what-is-cross-browser-compatibility/>

1.14. Package Managers: Herramientas como npm o Yarn que permiten gestionar dependencias (librerías y paquetes de código) en un proyecto.

1.14.1. Npm: es el sistema de gestión de paquetes por defecto para Node.js, un entorno de ejecución para JavaScript, bajo Artistic License 2.0. *Wikipedia*. Ver más: <https://www.npmjs.com/>

1.14.2. Yarn: es un instalador de paquetes JavaScript y gestor de dependencias lanzado por Facebook en colaboración con otros desarrolladores como Google. *Wikipedia*. Ver más: <https://yarnpkg.com/>

1.15. Testing: Métodos para probar la funcionalidad y estabilidad del código, utilizando herramientas como Jest, Cypress, o Mocha. Ver más: <https://web.dev/learn/testing>

1.15.1. Jest: es un marco de prueba de JavaScript centrado en la simplicidad. Funciona con proyectos que utilizan: Babel, TypeScript, Node, React, Angular, Vue y más. Ver más: <https://jestjs.io/>

1.15.2. Cypress: es una herramienta de automatización de pruebas frontend de código abierto para pruebas de regresión de aplicaciones web. Se ejecuta en Windows, Linux y macOS. *Wikipedia*. Ver más: <https://www.cypress.io/>

1.15.3. Mocha: es un marco de pruebas JavaScript robusto y lleno de funciones, que puede ejecutarse tanto en Node.js como en el navegador. Facilita la realización de pruebas asincrónicas de manera sencilla y eficiente. Las pruebas en Mocha se ejecutan de forma



secuencial, lo que permite notificaciones flexibles y precisas, además de capturar correctamente las excepciones y asignarlas a los casos de prueba correspondientes. Ver más: <https://mochajs.org/>

1.16. Vite: es una herramienta de desarrollo **rápida y ligera** diseñada para la creación de proyectos **front-end** modernos. Fue creada por **Evan You**, el autor de Vue.js, para resolver problemas de velocidad y rendimiento en el desarrollo con herramientas como Webpack o Parcel. Vite utiliza un enfoque diferente al de estos empaquetadores tradicionales, lo que le permite ser mucho más eficiente.

1.16.1. Principales características de Vite:

1.16.1.1. Compilación instantánea: Vite utiliza ESM (Módulos de JavaScript nativos) para cargar módulos directamente en el navegador, eliminando la necesidad de empaquetar todo el proyecto durante el desarrollo. Esto permite un arranque del servidor de desarrollo prácticamente inmediato, incluso en proyectos grandes.

1.16.1.2. Recarga rápida: Los cambios en el código son reflejados casi instantáneamente en la aplicación en ejecución, lo que acelera la productividad al trabajar en aplicaciones grandes.

1.16.1.3. Build eficiente: Durante la fase de producción, Vite usa **Rollup**, un empaquetador optimizado, para generar archivos altamente eficientes y optimizados para el navegador.

1.16.1.4. Frameworks soportados: Aunque fue inicialmente diseñado para Vue.js, Vite funciona muy bien con otros frameworks como React, Preact, Svelte, y Vanilla JavaScript.

2. Desarrollo móvil: se refiere a la creación de aplicaciones diseñadas para ejecutarse en dispositivos móviles, como smartphones y tabletas. Estas aplicaciones pueden ser desarrolladas para diferentes sistemas operativos móviles, como Android, iOS y Windows.

2.1.1. Android: Android es un sistema operativo móvil basado en el núcleo Linux y otros softwares de código abierto. *Wikipedia.*

2.1.2. IOS: OS es un sistema operativo móvil de código cerrado desarrollado por Apple Inc. Originalmente desarrollado para el iPhone, después se utilizó en dispositivos como el iPod touch y el iPad. *Wikipedia.*



2.1.3. Flutter: es un SDK de desarrollo de interfaz de usuario de código abierto administrado por Google. Funciona con el lenguaje de programación Dart. Crea aplicaciones compiladas de forma nativa de buen rendimiento y aspecto para dispositivos móviles (iOS, Android), web y de escritorio a partir de una única base de código. Ver más: <https://flutter.dev/>

2.1.4. Apache Cordova: es un popular entorno de desarrollo de aplicaciones móviles, originalmente creado por Nitobi. Adobe compró Nitobi en 2011, le cambió el nombre a PhoneGap, y más tarde liberó una versión de código abierto del software llamado Apache Cordova. *Wikipedia*. Ver más: <https://cordova.apache.org/>

2.1.5. Kotlin: es un lenguaje de programación multiplataforma, estáticamente tipado, de alto nivel y propósito general con inferencia de tipos. *Wikipedia*. Ver más: <https://kotlinlang.org/>

2.1.6. React Native: es un framework de código abierto creado por Meta Platforms, Inc. Se utiliza para desarrollar aplicaciones para Android, Android TV, iOS, macOS, tvOS, Web, Windows y UWP al permitir que los desarrolladores usen React con las características nativas de estas plataformas. *Wikipedia*. Ver más: <https://reactnative.dev/>

2.1.7. Flet: De acuerdo con la web oficial Flet construye aplicaciones multiplataforma en Python alimentadas por Flutter Flet. Permite a los desarrolladores construir fácilmente aplicaciones web, móviles y de escritorio en tiempo real en Python. Sin requerir experiencia frontend. Ver más: <https://flet.dev/>

3. Back End: El backend es la parte de una aplicación encargada de gestionar la lógica, los datos y la comunicación con el frontend. Esto incluye la implementación de la lógica de negocio, la administración de bases de datos y la creación de APIs, a través de las cuales el frontend puede enviar y recibir datos, facilitando la interactividad en la aplicación. También abarca aspectos de seguridad, como la autenticación y la protección de datos. El desarrollo backend se realiza con diversos lenguajes y frameworks, como JavaScript (Node.js), Python (Django, Flask), Ruby (Ruby on Rails), PHP (Laravel), Java (Spring) y C# (ASP.NET), y se ejecuta en servidores físicos o en la nube, que procesan las solicitudes de los usuarios.

II. HTML



HTML es el lenguaje básico que se emplea para el desarrollo de páginas de internet.

Esta constituido de elementos que el navegador interpreta y las despliega en la pantalla de acuerdo a su objetivo. Presenta elementos para disponer imágenes sobre una página, hipervínculos que nos permiten dirigirnos a otras páginas, listas, tablas, botones, etc.

Para poder crear una página HTML se requiere un editor de texto, por ejemplo:

Bloc de notas, Notepad++, Visual studio code, etc.

y un navegador de internet como: Chrome, IExplorer, FireFox, Safari, Opera, Edge, Brave, etc.

1. Estructura de una página HTML

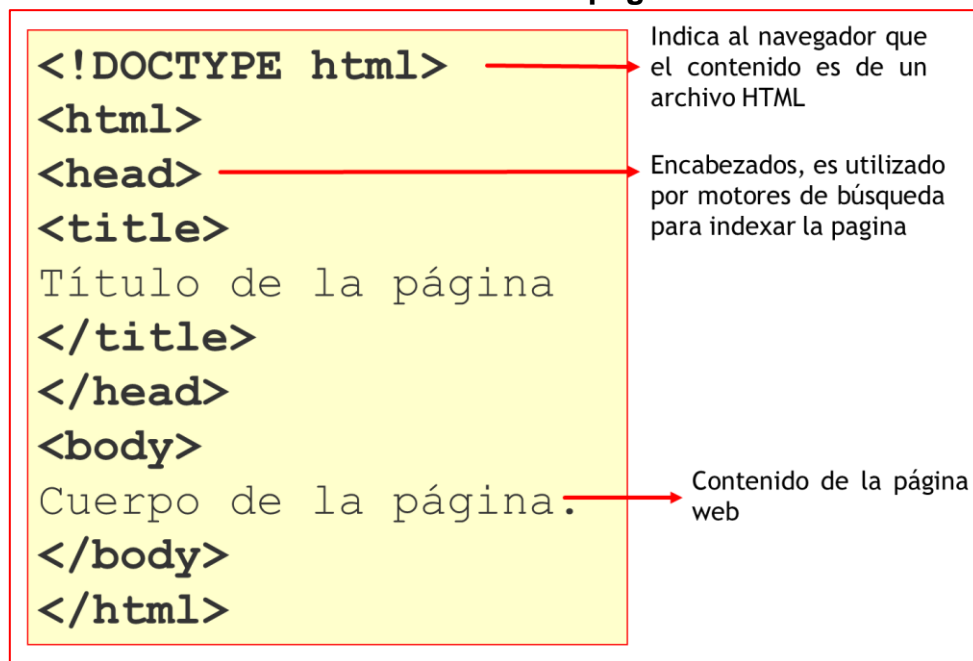


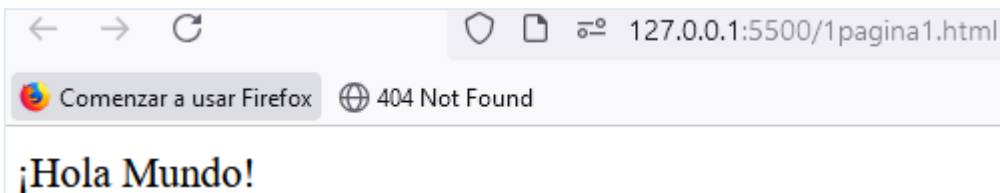
Figura 3: Estructura de HTML

- HTML no es sensible a mayúsculas y minúsculas.
- No requiere que dispongamos cada marca en una línea ejemplo `<head><title>Titulo</title></head>`
- Al guardar el archivo debemos colocarle la extensión ".html" por ejemplo podemos colocarle a nuestra página **pagina1.html**.

```
<!DOCTYPE html>
<html lang="es">
<head>
```



```
<!-- UTF-8 es un estándar de codificación de caracteres que
permite representar la mayoría de los caracteres de los idiomas
del mundo, incluidos símbolos y emojis.-->
<meta charset="UTF-8">
<!-- Permite que las páginas web sean responsivas, es decir se
vean bien en pantallas de diferentes tamaños.
<meta name="viewport">: Define las propiedades de la ventana
de visualización
    content="width=device-width": Establece el ancho de la ventana
de visualización para que coincida con el ancho de la pantalla del
dispositivo, sea móvil, tableta o computadora.
    initial-scale=1.0: Establece el nivel de zoom inicial en 1.0,
lo que significa que cuando la página se cargue, se mostrará sin
ningún zoom (escala al 100%). -->
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>¡Hola Mundo!</title>
</head>
<body>
    ¡Hola Mundo!
</body>
</html>
```



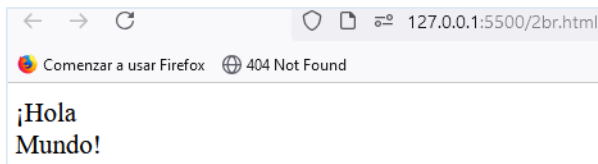
2. Elementos de HTML

- 2.1. Salto de línea
:** Realiza salto de línea. no necesita marca de cerrado, como buena práctica se recomienda no abusar de esta etiqueta y en lo posible usar CSS.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
```



```
<meta name="viewport" content="width=device-width, initial-  
scale=1.0">  
<title>¡Hola Mundo!</title>  
</head>  
<body>  
  ¡Hola <br> Mundo!  
</body>  
</html>
```

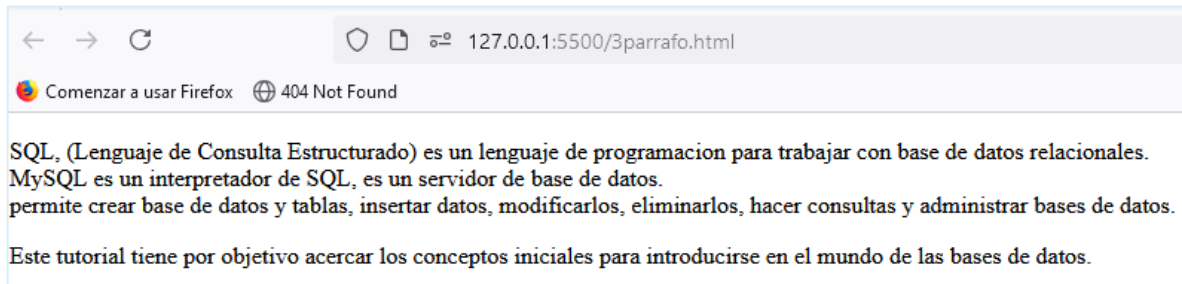


2.2. Párrafo <p>: Un párrafo es una oración o conjunto de oraciones referentes a un mismo tema. Todo lo que encerremos entre las marcas <p> y </p> aparecerá separado por un espacio con respecto al próximo párrafo.

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-  
scale=1.0">  
  <title>Ejercicio con parrafos</title>  
</head>  
<body>  
  <p> SQL, (Lenguaje de Consulta Estructurado) es un lenguaje  
    de programacion para trabajar con base de datos  
    relacionales.<br> MySQL es un interpretador de SQL, es un servidor  
    de base de datos. <br> permite crear base de datos y tablas,  
    insertar datos, modificarlos, eliminarlos, hacer consultas y  
    administrar bases de datos. </p> <p> Este tutorial tiene por  
    objetivo acercar los conceptos iniciales para introducirse en el  
    mundo de las bases de datos. </p>  
</body>
```

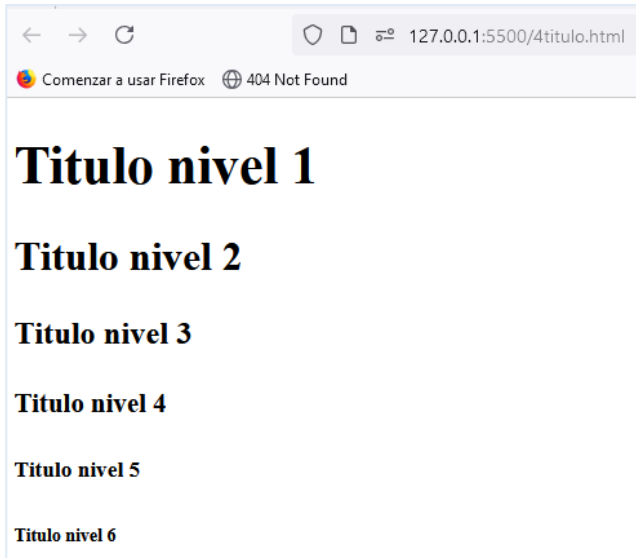


</html>



2.3. Títulos <h1><h2><h3><h4><h5><h6> : Son utilizados según la importancia del título, en el cual <h1> es el que tiene mayor nivel y así sucesivamente.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Títulos</title>
</head>
<body>
  <h1>Título nivel 1</h1>
  <h2>Título nivel 2</h2>
  <h3>Título nivel 3</h3>
  <h4>Título nivel 4</h4>
  <h5>Título nivel 5</h5>
  <h6>Título nivel 6</h6>
</body>
</html>
```



2.4. Énfasis (): Sirven para resaltar las palabras u oraciones de nuestra página, resalta en negrita y resalta en itálica.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Resaltar</title>
</head>
<body>
  <h1>Tipos de datos en MySQL</h1> <h2>varchar</h2>
  <p> se usa para almacenar cadenas de caracteres. <br> El tipo
  <strong>"varchar"</Strong> define una cadena de longitud
variable en la cual determinamos el máximo de caracteres. Puede
guardar hasta 255 caracteres.</p> <h2>int</h2>
  <p> Se usa para guardar valores <em>numéricos</em> enteros.<br>
  Definimos campos de este tipo cuando queremos representar, por
  ejemplo, cantidades.</p>
</body>
</html>
```



← → ↻ 127.0.0.1:5500/5resaltar.html

Comenzar a usar Firefox 404 Not Found

Tipos de datos en MySQL

varchar

se usa para almacenar cadenas de caracteres.
El tipo "varchar" define una cadena de longitud variable en la cual determinamos el máximo de caracteres. Puede guardar hasta 255 caracteres.

int

Se usa para guardar valores *numéricos* enteros.
Definimos campos de este tipo cuando queremos representar, por ejemplo, cantidades.

- 2.5. Hipervínculo a otra página del mismo sitio <a>:** El hipervínculo, nos permite cargar otra página en el navegador. La marca de hipervínculo a otra página del mismo sitio tiene la siguiente sintaxis: **Contexto**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Título de la página 1</title>
</head>
<body>
<h1>Página principal.</h1>
<a href="pagina2.html">Ir a pagina 2</a>
</body>
</html>
```

← → ↻ 127.0.0.1:5500/6pagina1.html

Comenzar a usar Firefox 404 Not Found

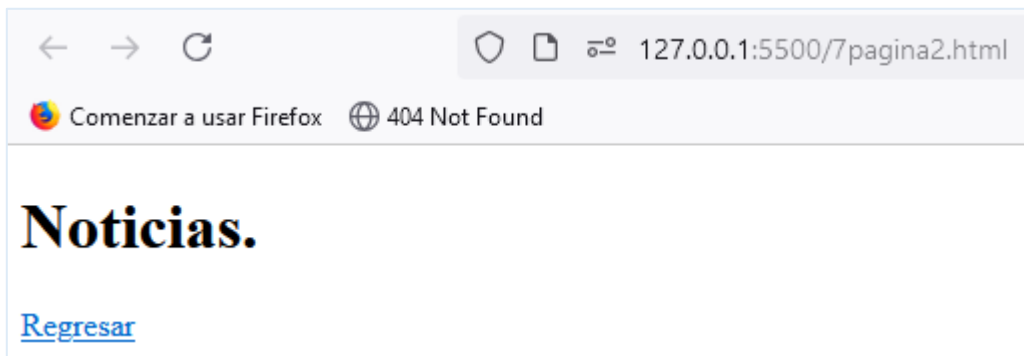
Página principal.

[Ir a pagina 2](#)

```
<!DOCTYPE html>
<html lang="es">
<head>
```



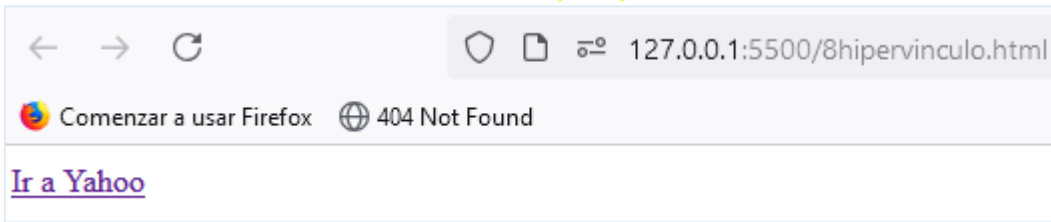
```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Título de la página 2</title>
</head>
<body>
<h1>Noticias.</h1>
<a href="pagina1.html">Regresar</a>
</body>
</html>
```



2.6. Hipervínculo a otro sitio de internet <a>: La sintaxis para disponer un hipervínculo a otro sitio de internet es:

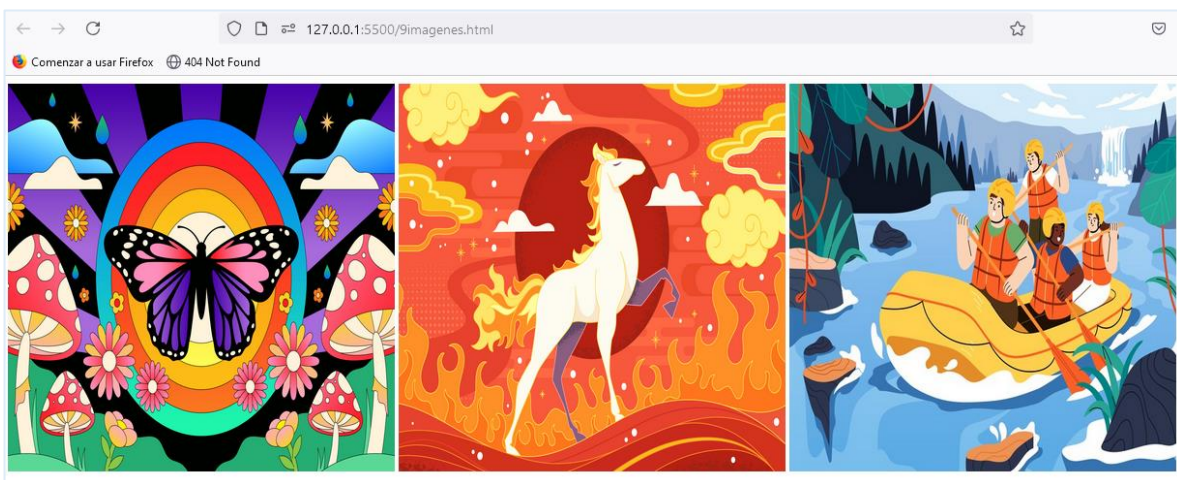
Buscador Google

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Visitar Yahoo</title>
</head>
<body>
  <a href= "http://www.yahoo.es">Ir a Yahoo</a>
</body>
</html>
```

2.7. Imágenes dentro de una página : podemos incrustar una imagen en un documento html.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
</head>
<title>Imágenes</title>
</head>
<body>
  
  
  
</body>
</html>
```





- 2.8. Hipervínculo mediante una imagen <a> y :** Consiste en disponer la marca encerrada entre la marca de comienzo y fin del enlace(<a>)

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Hipervínculo con imágenes</title>
</head>
<body>
  <h2>Presione alguna de las imágenes</h2>
  <a href="pagina1.html"> </a>
  <br>
  <a href="pagina2.html" target="_blank">
  </a>
</body>
</html>
```

- 2.9. Anclas llamadas desde la misma página:** Una práctica común cuando queremos desplazarnos dentro de una página de gran tamaño. Se disponen hipervínculos a diferentes anclas.

La sintaxis para definir un ancla es: ``

La sintaxis para ir a un ancla desde un hipervínculo es la siguiente:

`Introducción
`

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Anclas en la misma página</title>
</head>
<body>
  <h1>Tutorial de MySQL</h1>
  <a href="#introduccion">Introducción</a><br>
  <a href="#mostrarbasedatos">show databases</a><br>
  <a href="#creaciontabla">Creación de una tabla y mostrar sus
campos</a><br>
```



[Carga de registros a una tabla y su recuperación](#cargarregistros)

[](#introduccion)

Introducción

SQL, Structure Query Language (Lenguaje de Consulta Estructurado) es un lenguaje de programación para trabajar con base de datos relacionales como MySQL, Oracle, etc. MySQL es un interpretador de SQL, es un servidor de base de datos.

MySQL permite crear base de datos y tablas, insertar datos, modificarlos, eliminarlos, ordenarlos, hacer consultas y realizar muchas operaciones, etc., resumiendo: administrar bases de datos. Ingresando instrucciones en la línea de comandos o embebidas en un lenguaje como PHP nos comunicamos con el servidor. Cada sentencia debe acabar con punto y coma (;).

La sensibilidad a mayúsculas y minúsculas, es decir, si hace diferencia entre ellas, depende del sistema operativo, Windows no es sensible, pero Linux sí. Por ejemplo Windows interpreta igualmente las siguientes sentencias:

```
create database administracion;
```

```
Create DataBase administracion;
```

Pero Linux interpretará como un error la segunda.

Se recomienda usar siempre minúsculas. Es más el sitio `mysql.com.ar` está instalado sobre un servidor Linux por lo que todos los ejercicios deberán respetarse mayúsculas y minúsculas.

[](#mostrarbasedatos)

show databases

Una base de datos es un conjunto de tablas.

Una base de datos tiene un nombre con el cual accederemos a ella.

Vamos a trabajar en una base de datos ya creada en el sitio, llamada "administracion".

Para que el servidor nos muestre las bases de datos existentes, se lo solicitamos enviando la instrucción:

```
show databases;
```

Nos mostrará los nombres de las bases de datos, debe aparecer en este sitio "administracion".

[](#creaciontabla)

Creación de una tabla y mostrar sus campos

Una base de datos almacena sus datos en tablas.



Una tabla es una estructura de datos que organiza los datos en columnas y filas; cada columna es un campo

(o atributo) y cada fila, un registro. La intersección de una columna con una fila, contiene un dato específico, un solo valor.
 Cada registro contiene un dato por cada columna de la tabla.
 Cada campo (columna) debe tener un nombre. El nombre del campo hace referencia a la información que almacenará.

Cada campo (columna) también debe definir el tipo de dato que almacenará.
</p>

<h2>Carga de registros a una tabla y su recuperación</h2>

<p>Usamos "insert into". Especificamos los nombres de los campos entre paréntesis y separados por comas y luego los valores para cada campo, también entre paréntesis y separados por comas.
 Es importante ingresar los valores en el mismo orden en que se nombran los campos, si ingresamos los datos en otro orden, no aparece un mensaje de error y los datos se guardan de modo incorrecto.
 Note que los datos ingresados, como corresponden a campos de cadenas de caracteres se colocan entre comillas simples. Las comillas simples son obligatorias.</p>

</body>

</html>

← → ↻ 127.0.0.1:5500/11andclassself.html#cargarregistros

Comenzar a usar Firefox 404 Not Found

Tutorial de MySQL

[Introducción](#)
[show databases](#)
[Creación de una tabla y mostrar sus campos](#)
[Carga de registros a una tabla y su recuperación](#)

Introducción

SQL, Structure Query Language (Lenguaje de Consulta Estructurado) es un lenguaje de programación para trabajar con bases de datos. MySQL es un intérprete de SQL, es un servidor de base de datos. MySQL permite crear base de datos y tablas, insertar datos, modificarlos, eliminarlos, ordenarlos, hacer consultas. Ingresando instrucciones en la línea de comandos o embebidas en un lenguaje como PHP nos comunicamos con el servidor. La sensibilidad a mayúsculas y minúsculas, es decir, si hace diferencia entre ellas, depende del sistema operativo. Las siguientes sentencias:

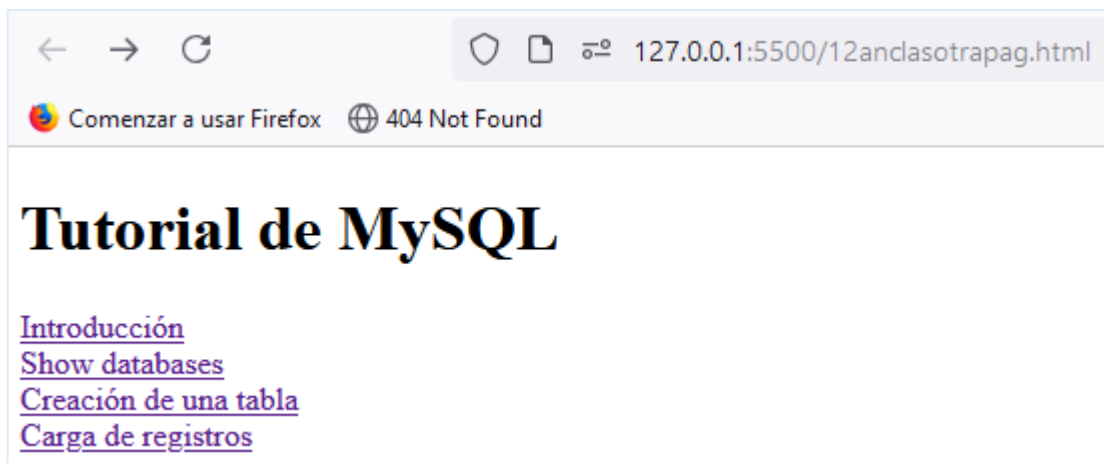
```
create database administracion;  
Create DataBase administracion;
```

Pero Linux interpretará como un error la segunda.
Se recomienda usar siempre minúsculas. Es más el sitio mysqlia.com.ar está instalado sobre un servidor Linux.

- 2.10. Anclas llamadas desde otra página:** Debemos conocer el nombre de la página a llamar y el nombre del ancla, luego la sintaxis para la llamada al ancla es: Introducción



```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Anclas a otras páginas</title>
</head>
<body>
  <h1>Tutorial de MySQL</h1>
  <a href="anclas.html#introduccion">Introducción</a><br>
  <a href="anclas.html#mostrarbasedatos">show databases</a><br>
  <a href="anclas.html#creaciontabla">Creación de una
tabla</a><br>
  <a href="anclas.html#cargarregistros">Carga de
registros</a><br>
</body>
</html>
```

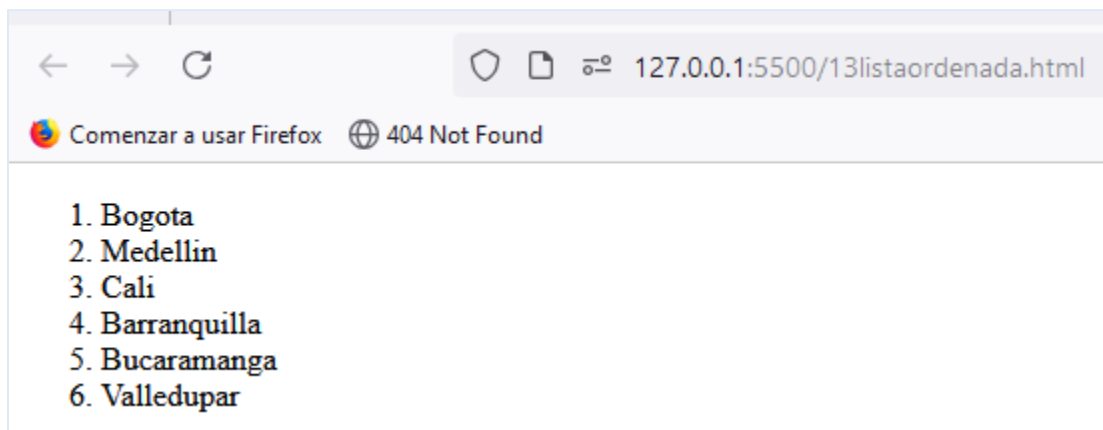


2.11. Lista ordenada (): Se utiliza cuando debemos numerar o listar una serie de objetos.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Lista Ordenada</title>
</head>
<body>
```



```
<ol>
  <li>Bogotá</li>
  <li>Medellín</li>
  <li>Cali</li>
  <li>Barranquilla</li>
  <li>Bucaramanga</li>
  <li>Valledupar</li>
</ol>
</body>
</html>
```

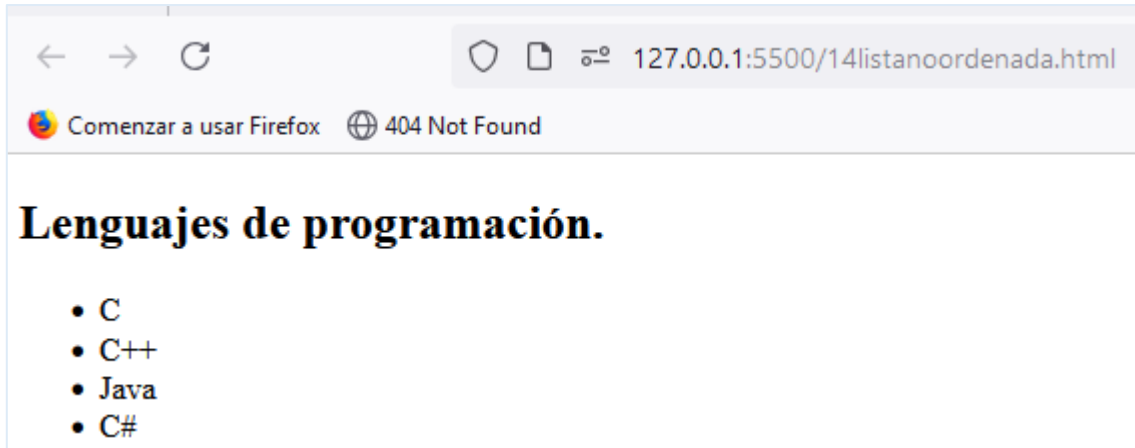


2.12. Lista no ordenada (): No utiliza un número delante de cada item sino un pequeño símbolo gráfico. La forma de implementar este tipo de listas es idéntica a las listas ordenadas.

```
<!DOCTYPE html>
<html Lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Lista no ordenada</title>
</head>
<body>
  <h2>Lenguajes de programación. </h2>
  <ul>
    <li>C</li>
    <li>C++</li>
    <li>Java</li>
    <li>C#</li>
  </ul>
```



```
</body>  
</html>
```

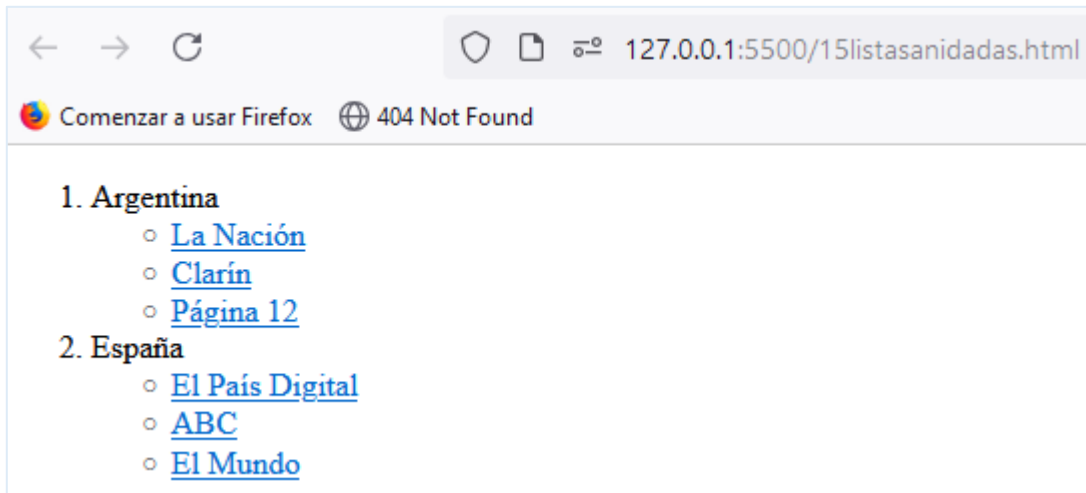


2.13. Listas anidadas: HTML nos permite insertar una lista dentro de otra. Se pueden anidar listas de distinto tipo, por ejemplo, podemos tener una lista no ordenada y uno de los ítem puede ser una lista ordenada.

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-  
scale=1.0">  
</head>  
  <title>Listas anidadas</title>  
</head>  
<body>  
  <ol>  
    <li>Argentina  
      <ul>  
        <li><a href="http://www.lanacion.com.ar">La Nación</a></li>  
        <li><a href="http://www.clarin.com.ar">Clarín</a></li>  
        <li><a href="http://www.pagina12.com.ar">Página 12</a></li>  
      </ul>  
    </li>  
    <li>España  
      <ul>  
        <li><a href="http://www.elpais.es">El País Digital</a></li>  
        <li><a href="http://www.abc.es">ABC</a></li>  
        <li><a href="http://www.elmundo.es">El Mundo</a></li>  
      </ul>  
    </li>  
  </ol>
```




```
</body>  
</html>
```



2.14. Tabla (<table><tr><td>): El objetivo de las tablas es mostrar una serie de datos en forma ordenada, organizado en filas y columnas.

Para la creación de una tabla intervienen una serie de elementos:

<table>: Es el elemento contenedor principal que define la tabla.

<thead>: Define el encabezado de la tabla. Generalmente se usa para agrupar las filas de encabezados (normalmente las que contienen los títulos de cada columna).

<tbody>: Agrupa el cuerpo de la tabla, donde se ubican las filas de datos principales. Es útil para separar el contenido de la tabla del encabezado y el pie.

<tfoot>: Define el pie de la tabla. Generalmente se usa para agrupar las filas que contienen sumas o conclusiones de los datos.

<tr> (Table Row): Representa una fila dentro de la tabla. Contiene celdas de datos o de encabezado.

<th> (Table Header): Representa una celda de encabezado dentro de una fila (<tr>). Su contenido generalmente aparece en negrita y se alinea en el centro de la celda de forma predeterminada.

<td> (Table Data): Representa una celda de datos en una tabla. Contiene el contenido de las filas de datos en el cuerpo de la tabla.



<caption>: Proporciona un título o descripción para la tabla. Generalmente se coloca encima o debajo de la tabla.

<colgroup>: Define un grupo de una o más columnas dentro de una tabla para aplicar estilos o atributos de forma conjunta.

<col>: Se utiliza dentro de un <colgroup> para definir las propiedades de una columna específica.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Tabla Completa en HTML</title>
  <style>
    /* Estilos básicos para la tabla */
    table {
      width: 100%; /* Ancho total de la tabla */
      border-collapse: collapse; /* Combina los bordes de las celdas */
    }
    th, td {
      border: 1px solid #000; /* Borde negro para celdas */
      padding: 8px; /* Espacio interior de las celdas */
      text-align: left; /* Alineación del texto a la izquierda */
    }
    th {
      background-color: #f2f2f2; /* Color de fondo para los encabezados */
    }
    caption {
      font-size: 1.5em; /* Tamaño del texto del título de la tabla */
      margin-bottom: 10px; /* Espacio debajo del título */
    }
    /* Estilo para las columnas definidas en <colgroup> */
    colgroup col {
      background-color: #f9f9f9; /* Color de fondo predeterminado para
las columnas */
    }
    tfoot td {
      font-weight: bold; /* Negrita en las celdas del pie de tabla */
    }
  </style>
</head>
<body>
  <!-- Tabla de inventario de productos -->
  <table>
```



```
<!-- Título o descripción de La tabla -->
<caption>Inventario de Productos en Tienda</caption>
<!-- Definición de Las columnas, con estilo para personalizar cada
columna -->
<colgroup>
  <!-- La primera columna tendrá un fondo azul claro -->
  <col span="1" style="background-color: #e0f7fa;">
  <!-- Las siguientes dos columnas tendrán el color de fondo
predeterminado (#f9f9f9) -->
  <col span="2">
</colgroup>
<!-- Encabezado de La tabla -->
<thead>
  <tr>
    <!-- Títulos de Las columnas -->
    <th>Producto</th>
    <th>Precio Unitario</th>
    <th>Cantidad en Inventario</th>
  </tr>
</thead>
<!-- Cuerpo de La tabla: aquí están Los datos -->
<tbody>
  <tr>
    <!-- Fila 1 de datos -->
    <td>Manzanas</td>
    <td>$2.00</td>
    <td>50</td>
  </tr>
  <tr>
    <!-- Fila 2 de datos -->
    <td>Plátanos</td>
    <td>$1.00</td>
    <td>100</td>
  </tr>
  <tr>
    <!-- Fila 3 de datos -->
    <td>Peras</td>
    <td>$1.50</td>
    <td>80</td>
  </tr>
</tbody>
<!-- Pie de La tabla: usualmente para mostrar totales o resúmenes -->
<tfoot>
  <tr>
    <!-- Primera celda de La fila del pie -->
    <td>Total</td>
    <!-- Colspan fusiona Las dos celdas siguientes en una sola -->
    <td colspan="2">230 Productos</td>
  </tr>
</tfoot>
```



```
</tr>
</tfoot>
</table>
</body>
</html>
```

| Inventario de Productos en Tienda | | |
|-----------------------------------|-----------------|------------------------|
| Producto | Precio Unitario | Cantidad en Inventario |
| Manzanas | \$2.00 | 50 |
| Plátanos | \$1.00 | 100 |
| Peras | \$1.50 | 80 |
| Total | 230 Productos | |

2.15. Tabla y combinación de celdas: En algunas situaciones se necesita que una celda ocupe el lugar de dos o más celdas en forma horizontal o vertical, para estos casos el elemento td o th dispone de dos propiedades llamadas rowspan y colspan.

A estas propiedades se les asigna un valor entero a partir de 2.

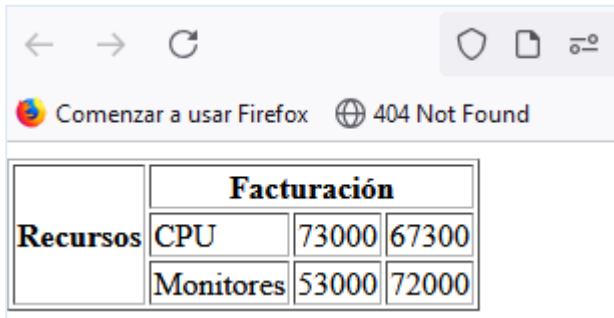
Si queremos que una celda ocupe tres columnas luego inicializamos la propiedad colspan con el valor 3: `<td colspan="3">CONTEXTO1</td>`

Si por el contrario queremos que una celda se extienda a nivel de filas luego hacemos: `<td rowspan="3">Secciones</td>`

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Combinación de celdas</title>
</head>
<body>
<table border="1">
  <tr>
    <th rowspan="3">Recursos</th>
    <th colspan="3">Facturación</th>
  </tr>
  <tr>
    <td>CPU</td>
    <td>73000</td>
    <td>67300</td>
```



```
</tr>
<tr>
  <td>Monitores</td>
  <td>53000</td>
  <td>72000</td>
</tr>
</table>
</body>
</html>
```



| Facturación | | | |
|-------------|-----------|-------|-------|
| Recursos | CPU | 73000 | 67300 |
| | Monitores | 53000 | 72000 |

2.16. Comentarios: Un comentario en el código sirve para documentarlo, ya que el navegador ignora su contenido. Son útiles para facilitar el mantenimiento del sitio, especialmente si otro desarrollador continúa el trabajo en el futuro, ya que lo que es obvio para uno puede no serlo para otro. Además, los comentarios se usan comúnmente para deshabilitar temporalmente partes del código durante el desarrollo.

La sintaxis es la siguiente **<!-- Aquí va el comentario -->**

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>¡Hola Mundo!</title> <!-- Título-->
</head>
<body>
  <!-- comentario-->
  ¡Hola Mundo!
</body>
</html>
```

2.17. División <div>: es un contenedor genérico en HTML que se utiliza para agrupar elementos y aplicarles estilos o scripts. No tiene un significado



específico por sí solo, pero es muy útil para estructurar el contenido en secciones y bloques en una página web.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Elemento div</title>
</head>
<body>
  <div style="background-color: rgb(231, 247, 20); padding:
10px;">
    <h2>Título de la Sección</h2>
    <p>parrafo en contenedor div</p>
  </div>
</body>
</html>
```

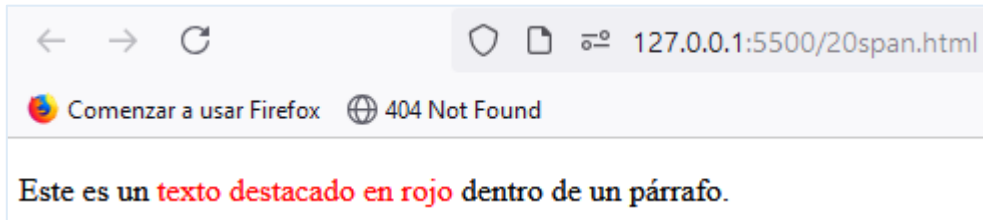


2.18. : es un contenedor en línea (inline) que se utiliza para agrupar texto u otros elementos dentro de una línea. Al igual que el <div>, no tiene un significado propio, pero es útil para aplicar estilos o scripts a partes específicas de un texto sin interrumpir el flujo del contenido.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Elemento span</title>
</head>
```



```
<body>
  <p>Este es un <span style="color: red;">texto destacado en
rojo</span> dentro de un párrafo.</p>
</body>
</html>
```



2.19. Formularios: Un formulario HTML es un contenedor que permite a los usuarios enviar datos a un servidor. Utiliza una combinación de controles como campos de texto, casillas de verificación, botones de radio, botones de envío, entre otros, para recopilar la información del usuario. El formulario se define usando la etiqueta `<form>` y es uno de los principales medios de interacción en páginas web.

2.20. Elementos de un Formulario HTML:

<form>: El contenedor principal de un formulario. Incluye atributos importantes como:

- **action**: La URL a la que se envían los datos.
- **method**: El método de envío, como GET o POST.

<input>: Es uno de los elementos más versátiles del formulario, y su tipo se define con el atributo `type`. Ejemplos comunes:

- **type="text"**: Campo de texto simple.
- **type="password"**: Campo para contraseñas, que oculta el texto ingresado.
- **type="radio"**: Botones de selección única.
- **type="checkbox"**: Casillas de verificación.
- **type="submit"**: Botón para enviar el formulario.
- **type="email"**: Campo para ingresar una dirección de correo electrónico con validación.
- **type="number"**: Campo para ingresar números.



- **type="file"**: Campo para subir archivos.
- **type="color"**: Campo para seleccionar color.

<label>: Etiqueta que describe los campos de entrada. El atributo `for` se asocia con el id del campo al que está vinculada.

<textarea>: Un área de texto para que el usuario ingrese múltiples líneas de texto.

<select> y **<option>**: Un menú desplegable de opciones. El elemento `<option>` se utiliza dentro de `<select>` para definir cada opción.

<button>: Un botón que puede usarse para enviar el formulario, o realizar una acción definida por JavaScript.

<fieldset> y **<legend>**: Agrupan elementos relacionados dentro del formulario. `<legend>` proporciona un título para el grupo.

<datalist>: Define una lista de opciones predefinidas que los usuarios pueden elegir en un campo de texto.

<output>: Muestra el resultado de un cálculo o acción basada en la entrada del usuario.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Formulario Completo</title>
  <!-- Enlace a Font Awesome para usar íconos -->
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/font-awesome/6.0.0-
beta3/css/all.min.css">
</head>
<body>
  <!-- Formulario que envía datos al servidor -->
  <form action="/procesar_datos" method="POST">
    <!-- Campo de texto para nombre con etiqueta -->
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre" required>
    <br><br>
    <!-- Campo de correo electrónico con validación -->
    <label for="email">Correo electrónico:</label>
    <input type="email" id="email" name="email" required>
```



```
<br><br>
<!-- Campo para ingresar contraseña -->
<label for="password">Contraseña:</label>
<input type="password" id="password" name="password" required>
<br><br>
<!-- Grupo de botones de radio para selección de género -->
<fieldset>
  <legend>Género:</legend>
  <input type="radio" id="masculino" name="genero"
value="masculino">
  <label for="masculino">Masculino</label>
  <input type="radio" id="femenino" name="genero"
value="femenino">
  <label for="femenino">Femenino</label>
</fieldset>
<br>
<!-- Casillas de verificación para hobbies -->
<fieldset>
  <legend>Hobbies:</legend>
  <input type="checkbox" id="leer" name="hobby" value="leer">
  <label for="leer">Leer</label>
  <input type="checkbox" id="deporte" name="hobby"
value="deporte">
  <label for="deporte">Deporte</label>
</fieldset>
<br>
<!-- Área de texto para comentarios -->
<label for="comentarios">Comentarios:</label><br>
<textarea id="comentarios" name="comentarios" rows="4"
cols="50"></textarea>
<br><br>
<!-- Menú desplegable para seleccionar país -->
<label for="pais">País:</label>
<select id="pais" name="pais">
  <option value="colombia">Colombia</option>
  <option value="mexico">México</option>
  <option value="españa">España</option>
</select>
<br><br>
<!-- Lista de sugerencias para ciudad usando datalist -->
<label for="ciudad">Ciudad:</label>
<input list="ciudades" id="ciudad" name="ciudad">
<datalist id="ciudades">
  <option value="Bogotá">
```



```
    <option value="Medellín">
    <option value="Cali">
</datalist>
<br><br>
<!-- Carga de archivo -->
<label for="archivo">Subir archivo:</label>
<input type="file" id="archivo" name="archivo">
<br><br>
<!-- Selector de color -->
<label for="colorFavorito">Selecciona tu color
favorito:</label>
<input type="color" id="colorFavorito" name="colorFavorito">
<br><br>
<!-- Botón de tipo button con ícono -->
<button type="button" onclick="alert('¡Botón presionado!')">
    <i class="fas fa-info-circle"></i> Información
</button>
<br><br>
<!-- Botón de enviar -->
<input type="submit" value="Enviar">
</form>
</body>
</html>
```

Nombre:

Correo electrónico:

Contraseña:

Género:

☐ Masculino ☐ Femenino

Hobbies:

☐ Leer ☐ Deporte

Comentarios:

País:

Ciudad:

Subir archivo: No se ha seleccionado ningún archivo.

Selecciona tu color favorito:



2.21. Atributos de `<input type="text">`:

name: Define el nombre del campo de texto, que se utiliza cuando los datos se envían al servidor.

id: Proporciona un identificador único para el campo de texto, que puede ser usado para asociar etiquetas (`<label>`) o para aplicar estilos o interactividad con JavaScript.

value: Establece un valor predeterminado en el campo de texto.

placeholder: Muestra un texto temporal dentro del campo de entrada que desaparece cuando el usuario empieza a escribir. Sirve como sugerencia de lo que el usuario debería ingresar.

required: Obliga a que el usuario complete este campo antes de enviar el formulario. Si no se completa, el formulario no se enviará y mostrará un mensaje de error.

readonly: Hace que el campo sea de solo lectura. El usuario puede ver el valor pero no puede modificarlo.

disabled: Desactiva el campo de entrada, lo que significa que no es editable ni interactivo. No se enviará su valor cuando se envíe el formulario.

maxlength: Define el número máximo de caracteres que el usuario puede ingresar.

minlength: Define el número mínimo de caracteres que el usuario debe ingresar.

size: Especifica el número de caracteres visibles en el campo de texto. Afecta la anchura del campo de entrada.

autocomplete: Controla si el navegador debería ofrecer autocompletar el campo de texto. Los valores posibles son on (activado) o off (desactivado).



pattern: Permite definir una expresión regular para validar lo que se ingresa en el campo de texto. Si la entrada no coincide con el patrón, el formulario no se enviará.

autofocus: Coloca el foco automáticamente en el campo de entrada cuando la página se carga.

form: Asocia el campo de texto a un formulario específico si el campo está fuera del contenedor `<form>`.

list: Asocia el campo de texto con un elemento `<datalist>` para proporcionar una lista de sugerencias predefinidas.

spellcheck: Define si se debe habilitar la verificación ortográfica en el campo. Los valores posibles son `true` o `false`.

inputmode: Especifica el tipo de teclado que se debe mostrar en dispositivos móviles (por ejemplo, numérico, texto, correo electrónico).

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Ejemplo de input type="text"</title>
</head>
<body>
  <form action="/procesar_datos" method="POST">
    <!-- Campo de texto con nombre e id -->
    <label for="nombre">Nombre:</label>
    <input type="text" id="nombre" name="nombre" required
maxlength="30" placeholder="Ingresa tu nombre completo">
    <br><br>
    <!-- Campo de texto con valor predeterminado y solo lectura -->
    <label for="nombreLectura">Nombre predeterminado (solo
lectura):</label>
    <input type="text" id="nombreLectura" name="nombreLectura"
value="Juan Pérez" readonly>
    <br><br>
    <!-- Campo de texto deshabilitado -->
    <label for="nombreDeshabilitado">Este campo está
deshabilitado:</label>
```



```
<input type="text" id="nombreDeshabilitado"
name="nombreDeshabilitado" value="No editable" disabled>
<br><br>
<!-- Campo de texto con verificación de longitud mínima y
máxima -->
<label for="usuario">Nombre de usuario (3-15
caracteres):</label>
<input type="text" id="usuario" name="usuario" minlength="3"
maxlength="15" required>
<br><br>
<!-- Campo de texto con patrón (solo letras y espacios) -->
<label for="patron">Solo letras y espacios:</label>
<input type="text" id="patron" name="patron" pattern="[A-Za-
z\s]+" title="Solo se permiten letras y espacios">
<br><br>
<!-- Campo de texto con sugerencias de autocompletado -->
<label for="pais">País:</label>
<input list="países" id="pais" name="pais"
placeholder="Selecciona o escribe tu país">
<datalist id="países">
  <option value="Colombia">
  <option value="México">
  <option value="España">
</datalist>
<br><br>
<!-- Campo de texto con tamaño visual ajustado -->
<label for="ajustado">Campo ajustado (5 caracteres
visibles):</label>
<input type="text" id="ajustado" name="ajustado" size="5">
<br><br>
<!-- Campo de texto con enfoque automático -->
<label for="foco">Este campo tiene autofocus:</label>
<input type="text" id="foco" name="foco" autofocus>
<br><br>
<!-- Campo de texto con autocompletar desactivado -->
<label for="sinAutocompletar">Campo sin autocompletar:</label>
<input type="text" id="sinAutocompletar"
name="sinAutocompletar" autocomplete="off" placeholder="Escribe
algo aquí...">
<br><br>
<!-- Botón de enviar -->
<input type="submit" value="Enviar">
</form>
</body>
```



</html>

A screenshot of a web browser window. The address bar shows the URL '127.0.0.1:5500/22atributosinput.html'. The browser's status bar indicates 'Comenzar a usar Firefox' and '404 Not Found'. The form contains the following elements:

- Nombre:** A text input field with the placeholder text 'Ingresa tu nombre comple'.
- Nombre predeterminado (solo lectura):** A text input field containing the value 'Juan Pérez'.
- Este campo está deshabilitado:** A text input field with the value 'No editable'.
- Nombre de usuario (3-15 caracteres):** An empty text input field.
- Solo letras y espacios:** An empty text input field.
- País:** A text input field with the placeholder text 'Selecciona o escribe tu pa'.
- Campo ajustado (5 caracteres visibles):** A text input field.
- Este campo tiene autofocus:** A text input field with a blue border, indicating it has focus.
- Campo sin autocompletar:** A text input field with the placeholder text 'Escribe algo aquí...'.
- Enviar:** A button with the text 'Enviar'.

III. HOJAS DE ESTILO EN CASCADA - CSS

Nos permite controlar la apariencia de una página web ya que, en un principio, los sitios web se concentraban más en su contenido que en su presentación.

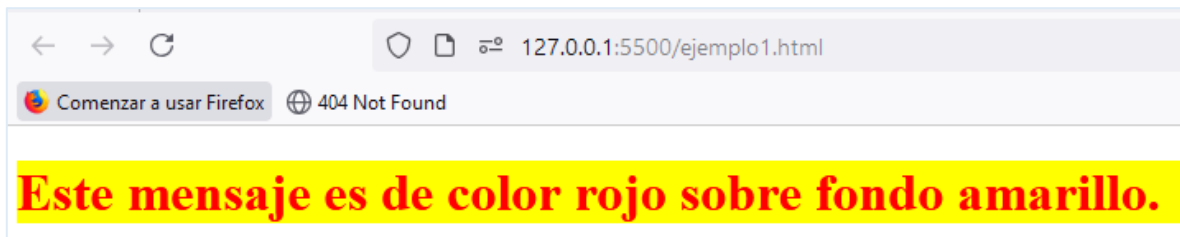
Las podemos asociar a nuestra página HTML de tres maneras: directamente en las respectivas etiquetas lo cual no es recomendable, en el head (encabezado) de la página o agrupar las reglas de estilo en un archivo independiente con extensión *.css (esta es la forma más adecuada).

1. Definición de estilos a nivel de elemento HTML

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
```



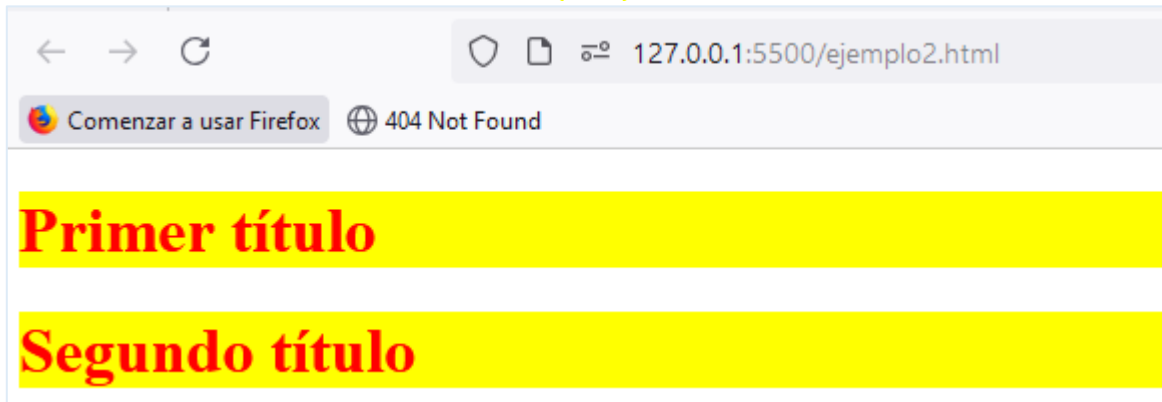

```
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<head> <title>Ejemplo css a nivel de etiqueta</title></head>
<body>
  <h1 style="color:#ff0000;background-color:#ffff00">
    Este mensaje es de color rojo sobre fondo amarillo. </h1>
</body>
</html>
```



2. Definición de estilos a nivel de página

También podemos hacer la definición de estilos en una sección de la cabecera (HEAD) que la encerramos entre las marcas `<style>` `</style>` (en su interior definimos los estilos para los elementos HTML que necesitamos), este método es más eficiente que el anterior donde se agregaban los estilos en el cuerpo de la página.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Ejemplo css a nivel de página</title>
  <style>
    h1 {color:#ff0000;background-color:#ffff00}
  </style>
</head>
<body>
  <h1>Primer título</h1>
  <h1>Segundo título</h1>
</body>
</html>
```



3. Definición de hojas de estilo en un archivo externo

Hasta ahora hemos visto la definición de estilos a nivel de elemento HTML y la definición de estilos a nivel de página. La primera forma es muy poco recomendada y la segunda es utilizada cuando queremos definir estilos que serán empleados sólo por esa página.

La metodología más empleada es la definición de una hoja de estilo en un archivo separado que deberá tener la extensión css. Este archivo contendrá las reglas de estilo, pero estarán separadas del archivo HTML.

- La ventaja fundamental es que con esto podemos aplicar las mismas reglas de estilo a cualquier página del sitio web. Es decir, cambiando las reglas de estilo de este archivo estaremos cambiando la apariencia de múltiples páginas del sitio.
- También tiene como ventaja que al programador le resulta más ordenado tener lo referente a HTML en un archivo y las reglas de estilo en un archivo aparte.
- Otra ventaja es que cuando un navegador solicita una página, se le envía el archivo HTML y el archivo CSS, quedando guardado este último archivo en la caché de la computadora, con lo cual, en las sucesivas páginas que requieran el mismo archivo de estilos, ese mismo archivo se rescata de la caché y no requiere que el servidor web se lo reenvíe (ahorrando tiempo de transferencia)

Ejemplo3.html

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
```



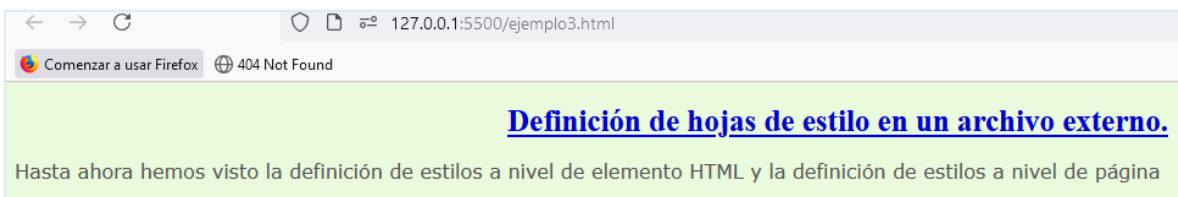
```
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<title>Ejemplo css archivo externo</title>
<link rel="StyleSheet" href="css/estilos.css" type="text/css">
</head>
<body>
  <h1>Definición de hojas de estilo en un archivo externo.</h1>
  <p> Hasta ahora hemos visto la definición de estilos a nivel
de elemento HTML y la definición de estilos a nivel de página</p>
</body>
</html>
```

Estilos.css

```
body {
  background-color:#eafadd;
}

h1 {
  color:#0000cc;
  font-family:times new roman;
  font-size:25px;
  text-align:center;
  text-decoration:underline;
}

p {
  color:#555555;
  font-family:verdana;
  text-align:justify;
}
```





3.1. Normalize.css

Normalize.css es una hoja de estilos que busca uniformar la apariencia de los elementos HTML en todos los navegadores. A diferencia de los frameworks de "reset" que eliminan todos los estilos, Normalize.css conserva los valores útiles predeterminados del navegador y soluciona inconsistencias comunes de renderizado.

Puntos clave:

Consistencia: Arregla errores típicos de visualización en navegadores como Firefox, Chrome y Safari.

Accesibilidad mejorada: Optimiza los estilos para mejor legibilidad en distintas plataformas.

Preserva estilos útiles: No borra todos los estilos, manteniendo valores importantes en botones y formularios.

Ventajas:

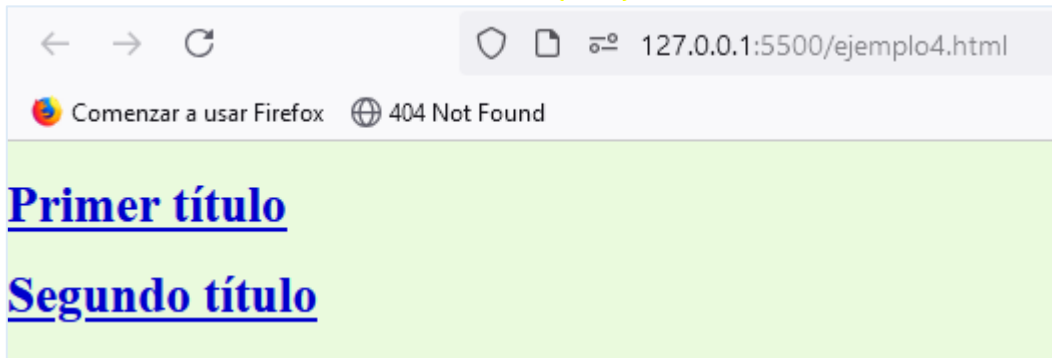
Garantiza uniformidad en la apariencia web sin modificar todos los estilos.

Evita comportamientos inesperados en diferentes navegadores, especialmente en formularios y tablas.

Enlace a Normalize.css

<https://necolas.github.io/normalize.css/>

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css"
type="text/css">
  <link rel="StyleSheet" href="css/estilos.css" type="text/css">
  <title>Ejemplo css con normalize</title>
</head>
<body>
  <h1>Primer título</h1>
  <h1>Segundo título</h1>
</body>
</html>
```

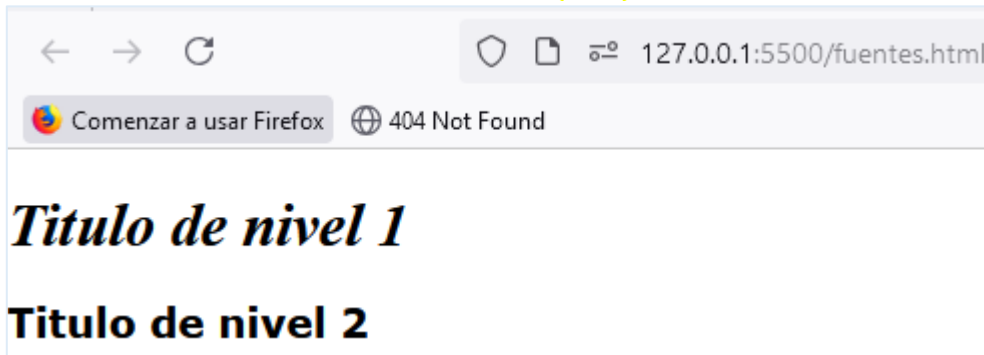


3.2. Propiedades relacionadas a fuentes.

Existe una serie de propiedades relacionadas a fuentes: font-family, font-size, font-style, font-weight, font-variant,

En el siguiente ejemplo por lo comididad se dejara todo en el mismo archivo.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css"
type="text/css">
  <title>Fuentes</title>
  <style>
    h1 { font-family:times new roman;
        font-size:30px;
        font-style:italic;
        font-weight:bold;
        }
    h2 { font-family:verdana;
        font-size:20px;
        }
  </style>
</head>
<body>
  <h1>Titulo de nivel 1</h1>
  <h2>Titulo de nivel 2</h2>
</body>
</html>
```



3.3. Agrupación de varios elementos HTML con una misma regla de estilo

Esta característica nos permite ahorrar la escritura de reglas duplicadas para diferentes elementos de HTML.

La sintaxis es disponer los nombres de los elementos HTML separados por comas.

Supongamos que queremos la misma fuente y color para los elementos h1, h2 y

h3: `h1,h2,h3 { font-family:verdana; color:#0000ff; }`

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css"
type="text/css">
  <style>
    h1,h2,h3 { font-family:verdana; color:#0000ff; }
  </style>
</head>
<body>
  <h1>Titulo de nivel 1</h1>
  <h2>Titulo de nivel 2</h2>
  <h3>Titulo de nivel 3</h3>
</body>
</html>
```



3.4. Definición de varias reglas para un mismo elemento HTML.

Podemos definir más de una regla para un elemento HTML, en este ejemplo el elemento h1 tiene dos reglas:

```
h1,h2,h3,h4,h5,h6 {  
    font-family:Verdana;  
}  
  
h1 {  
    font-size:40px;  
}
```

3.5. Propiedades relacionadas al texto (color, text-align, text-decoration)

color, nos permite definir el color del texto, lo podemos indicar por medio de tres valores hexadecimales que indican la mezcla de rojo, verde y azul. Por ejemplo si queremos rojo puro debemos indicar: **color:#ff0000;**

Otra forma de indicar el color, es por medio de la siguiente sintaxis: **color:rgb(255,0,0);** Es decir, por medio de la función rgb(red,green,blue), indicamos la cantidad de rojo, verde y azul en formato decimal.

La segunda propiedad relacionada al texto es text-align, que puede tomar alguno de estos cuatro valores: left, right, Center, justify Si especificamos: **text-align:center;**

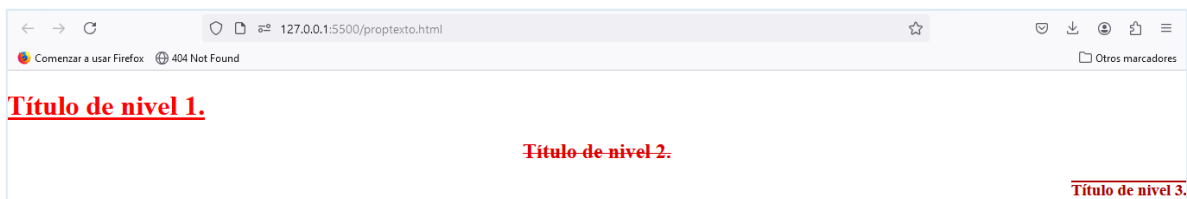
El texto aparecerá centrado. Si queremos justificar a derecha, emplearemos el valor right y si queremos a la izquierda, el valor será left.



La tercera propiedad relacionada al texto es text-decoration que nos permite que aparezca subrayado el texto, tachado o una línea en la parte superior, los valores posibles de esta propiedad son:

None, underline, overline, line-through.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css"
type="text/css">
  <style>
    h1 { color:#ff0000; text-align:left; text-
decoration:underline; }
    h2 { color:#dd0000; text-align:center; text-
decoration:line-through; }
    h3 { color:#aa0000; text-align:right; text-
decoration:overline; }
  </style>
</head>
<body>
  <h1>Título de nivel 1.</h1>
  <h2>Título de nivel 2.</h2>
  <h3>Título de nivel 3.</h3>
</body>
</html>
```



3.6. Otras propiedades relacionadas al texto (letter-spacing, word-spacing, text-indent, text-transform)

La propiedad letter-spacing y word-spacing permiten indicar el espacio que debe haber entre los caracteres y entre las palabras respectivamente.



La propiedad `text-indent`, indenta la primera línea de un texto. A partir de la segunda línea, el texto aparece sin indentación. Podemos indicar un valor negativo con lo que la indentación es hacia la izquierda.

Por último, la propiedad `text-transform` puede inicializarse con alguno de los siguientes valores:

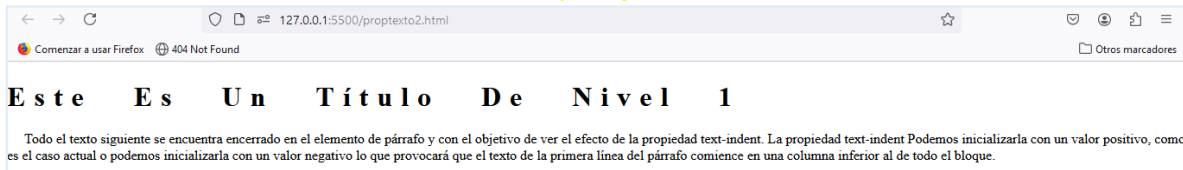
`capitalize`: Dispone en mayúsculas el primer carácter de cada palabra.

`lowercase`: Convierte a minúsculas todas las letras del texto.

`uppercase`: Convierte a mayúsculas todas las letras del texto.

`none`: No provoca cambios en el texto.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css"
type="text/css">
  <style>
    h1 { letter-spacing:10px; word-spacing:30px;
text-transform:capitalize; }
    p { text-indent:20px; }
  </style>
</head>
<body>
  <h1>Este es un título de nivel 1</h1>
  <p>Todo el texto siguiente se encuentra encerrado en el
elemento
  de párrafo y con el objetivo de ver el efecto de la propiedad
text-indent. La propiedad text-indent Podemos inicializarla
con
  un valor positivo, como es el caso actual o podemos
inicializarla
  con un valor negativo lo que provocará que el texto de la
primera
  línea del párrafo comience
  en una columna inferior al de todo el bloque. </p>
</body>
</html>
```



3.7. Herencia de propiedades de estilo.

La mayoría de los estilos se heredan, es decir si definimos la propiedad color para el elemento h1, si dicho elemento incorpora un texto con el elemento em en su interior, la propiedad color del elemento em tendrá el mismo valor que la propiedad h1 (es decir el elemento em hereda las propiedades del elemento h1).

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css"
type="text/css">
  <title>Herencia</title>
  <style>
    body { color:#0000ff; font-family:verdana; }
  </style>
</head>
<body>
  <h1>Este es un título de nivel 1 y con el elemento
  'em' la palabra:<em>Hola</em>
  </h1>
  <p>Todo este párrafo debe ser de color azul ya que
  lo hereda del elemento body.</p>
</body>
</html>
```



3.8. Definición de estilos por medio de clases



Una regla de estilo puede ser igual para un conjunto de elementos HTML, en esos casos conviene plantear una regla de estilo con un nombre genérico que posteriormente se puede aplicar a varios elementos de HTML.

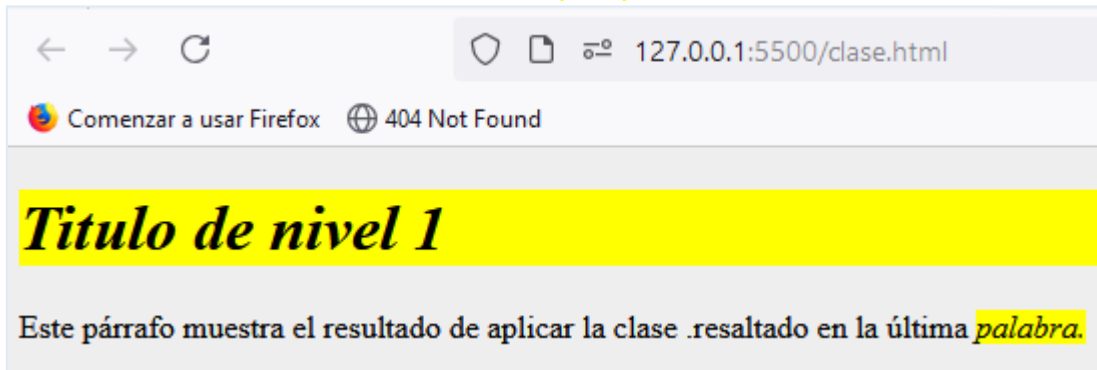
Para la definición de una regla de estilo por medio de una clase creamos un nombre de clase y le antecedemos un punto:

```
.resaltado{  
    color:#000000;  
    background-color:#ffff00;  
    font-style:italic;  
}
```

Luego para asignar dicha regla a un elemento HTML definimos la propiedad class al elemento que necesitamos fijarle este estilo:

<h1 class="resaltado">Este elemento h1 aparece con la clase resaltado</h1>

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-scale=1.0">  
    <style>  
        body {  
            background-color:#eeeeee;  
        }  
        .resaltado{  
            color:#000000;  
            background-color:#ffff00;  
            font-style:italic;  
        }  
    </style>  
</head>  
<body>  
    <h1 class="resaltado">Titulo de nivel 1</h1>  
    <p> Este párrafo muestra el resultado de aplicar la clase  
    .resaltado en la última <span  
class="resaltado">palabra.</span> </p>  
</body>  
</html>
```



3.9. Definición de estilos por medio de id.

Con esta opción sólo podremos aplicar dicho estilo a una solo elemento HTML dentro de la página, ya que todos los id que se definen en una página HTML deben tener nombres distintos

```
#cabecera {  
    font-family:Times New Roman;  
    font-size:30px;  
    text-align:center;  
    color:#0000ff;  
    background-color:#bbbbbb;  
}
```

Luego, sólo un elemento HTML dentro de una página puede definir un estilo de este tipo: `<div id="cabecera">`

Sólo un elemento HTML puede definir la propiedad id con el valor de cabecera.

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width, initial-  
scale=1.0">  
    <link rel="StyleSheet" href="css/normalize.css"  
type="text/css">  
    <title>Definición de Id</title>  
    <style>  
        #cabecera {  
            font-family:Times New Roman;  
            font-size:30px;
```



```
        text-align:center;
        color:#0000ff;
        background-color:#bbbbbb;
    }
</style>
</head>
</head>
<body>
<div id="cabecera">
<h1>Título de la cabecera</h1>
</div>
</body>
</html>
```



3.10. *Propiedades relacionadas al borde*

Todo elemento que se crea dentro de una página HTML genera una caja. Imaginemos los controles que hemos creado h1, h2, h3, p, em, etc. si fijamos la propiedad background-color veremos que el contenido se encuentra dentro de un rectángulo.

Podemos acceder a las propiedades del borde de ese rectángulo mediante las hojas de estilo CSS; las propiedades más importantes a las que tenemos acceso son:

border-width, border-style, border-color

Se disponen de los siguientes estilos de borde:

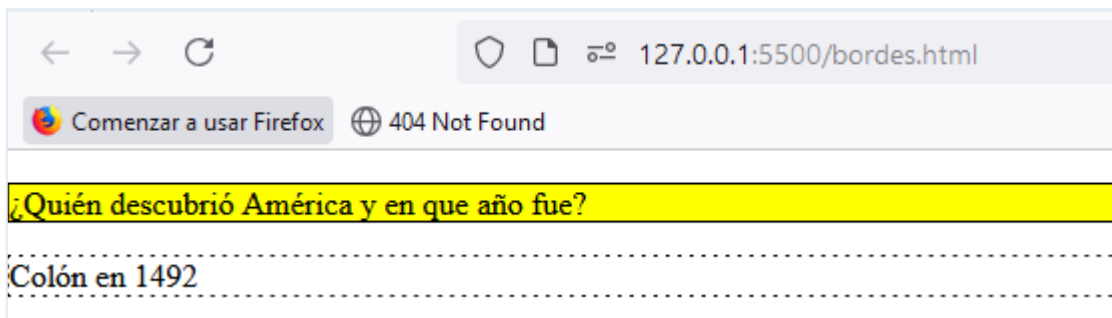
None, hidden, dotted, dashed, solid, double, groove, ridge, inset, outset

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <link rel="StyleSheet" href="css/normalize.css"
type="text/css">
```



```
<title>Bordes</title>
<style>
    .pregunta {
        background-color:#ffff00;
        border-width:1px;
        border-style:solid;
        border-color:#000000;
    }

    .respuesta {
        border-width:1px;
        border-style:dashed;
        border-color:#000000;
    }
</style>
</head>
<body>
    <p class="pregunta">¿Quién descubrió América y en que año
fue?</p>
    <p class="respuesta">Colón en 1492</p>
</body>
</html>
```



3.11. Margin y padding

3.11.1. Padding: Es el espacio interno entre el contenido de un elemento (como texto o imágenes) y el borde de ese elemento.

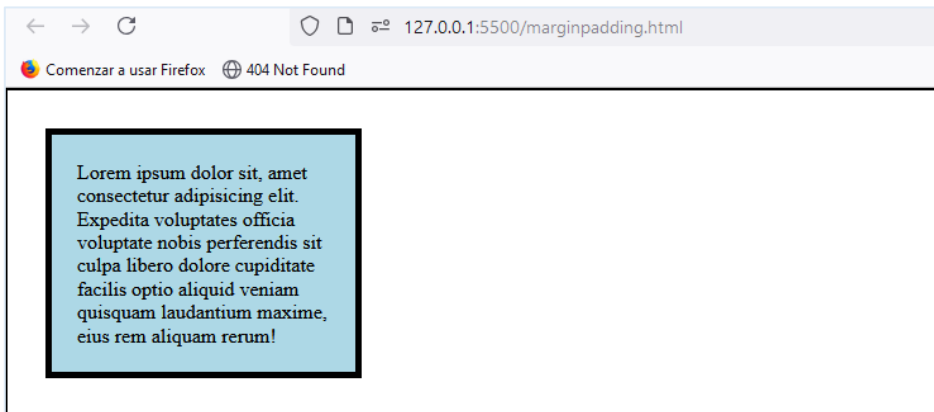
3.11.2. Margin: Es el espacio externo entre el borde de un elemento y los elementos que lo rodean.

3.11.3. Border: Es el borde que rodea el padding y el contenido de un elemento.

```
<!DOCTYPE html>
<html lang="es">
<head>
```



```
<meta charset="UTF-8">
<meta name="viewport" content="width=device-width, initial-
scale=1.0">
<link rel="StyleSheet" href="css/normalize.css" type="text/css">
<title>Ejemplo de Padding, Margin y Borde</title>
<style>
  .box {
    width: 200px;
    padding: 20px; /* Espacio interno */
    border: 5px solid black; /* Borde */
    margin: 30px; /* Espacio externo */
    background-color: lightblue;
  }
  #marco{
    border: 2px solid black; /* Borde */
  }
</style>
</head>
<body>
  <div id="marco">
    <div class="box">
      Lorem ipsum dolor sit, amet consectetur adipisicing elit.
      Expedita voluptates officia voluptate
      nobis perferendis sit culpa libero dolore cupiditate
      facilis optio aliquid veniam quisquam
      laudantium maxime, eius rem aliquam rerum!
    </div>
  </div>
</body>
</html>
```





3.12. Unidades de medida

3.12.1. Unidades Absolutas: Definen tamaños fijos que no cambian según el dispositivo o la ventana.

px (píxeles): Unidad más común, representa un píxel en la pantalla del dispositivo.

cm (centímetros), mm (milímetros), in (pulgadas): Medidas físicas basadas en el mundo real, útiles en impresiones.

pt (puntos): Usado principalmente para impresiones, 1 pt es 1/72 de pulgada.

3.12.2. Unidades Relativas: El tamaño depende de otros valores, como el contenedor o la ventana del navegador.

% (porcentaje): Relativo al tamaño del contenedor padre. Ejemplo: 50% del ancho del contenedor.

em: Relativo al tamaño de la fuente del contenedor padre. Ejemplo: 2em es el doble del tamaño de la fuente del padre.

rem: Relativo al tamaño de la fuente del root o elemento raíz (generalmente el tamaño definido en el html). Ejemplo: 1rem es igual al tamaño de la fuente raíz.

vw (viewport width): Relativo al ancho del viewport (ventana visible del navegador). 1vw es el 1% del ancho del viewport.

vh (viewport height): Relativo a la altura del viewport. 1vh es el 1% de la altura del viewport.

ex y ch: Relativos a características de la fuente (altura de la letra "x" o el ancho del carácter "0", respectivamente).

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css"
type="text/css">

  <title>Ejemplo de Unidades en CSS</title>
  <style>
    /* Definimos el tamaño base de la fuente del documento en 16px
  */
```




```
:root {
  font-size: 16px;
}

/* Caja con medidas en px (píxeles) */
.box-px {
  width: 200px; /* 200 píxeles de ancho */
  height: 100px; /* 100 píxeles de alto */
  background-color: lightcoral;
}

/* Caja con medidas en porcentaje */
.box-percent {
  width: 50%; /* 50% del ancho del contenedor */
  height: 100px;
  background-color: lightblue;
}

/* Caja con fuente relativa a em (basado en el tamaño de la
fuente del contenedor padre) */
.box-em {
  font-size: 1.5em; /* 1.5 veces el tamaño de la fuente del
padre */
  background-color: lightgreen;
}

/* Caja con medidas relativas al viewport (ventana visible) */
.box-vw-vh {
  width: 50vw; /* 50% del ancho del viewport */
  height: 30vh; /* 30% de la altura del viewport */
  background-color: lightgoldenrodyellow;
}

/* Caja con fuente relativa a rem (basado en el tamaño de la
fuente raíz) */
.box-rem {
  font-size: 2rem; /* 2 veces el tamaño de la fuente raíz
(32px si el root tiene 16px) */
  background-color: lightpink;
}
</style>
</head>
<body>
<!-- Caja con tamaño en píxeles -->
```



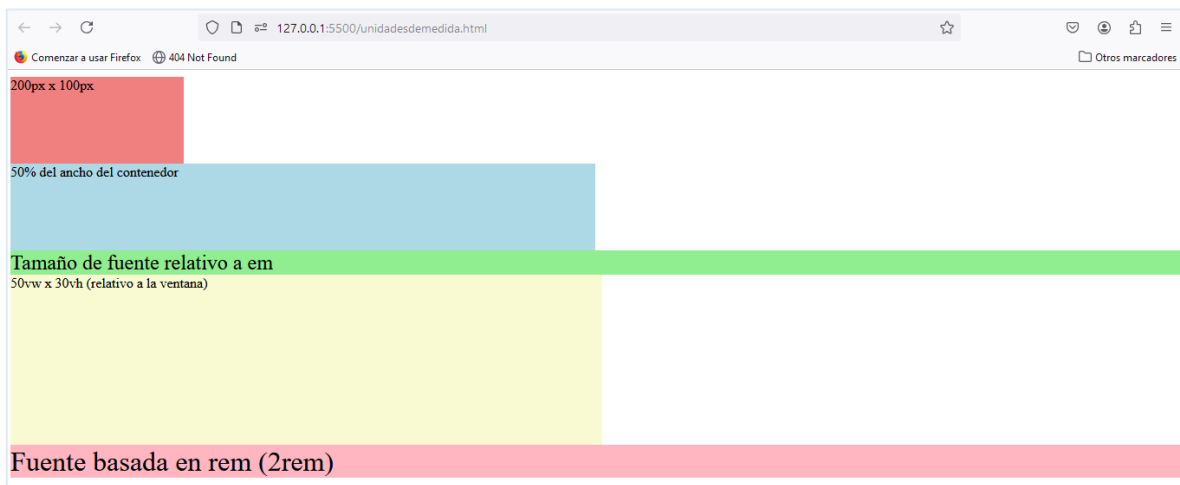
```
<div class="box-px">200px x 100px</div>

<!-- Caja con tamaño en porcentaje -->
<div class="box-percent">50% del ancho del contenedor</div>

<!-- Caja con tamaño de fuente basado en em -->
<div class="box-em">Tamaño de fuente relativo a em</div>

<!-- Caja con tamaño relativo a la ventana del navegador -->
<div class="box-vw-vh">50vw x 30vh (relativo a la ventana)</div>

<!-- Caja con tamaño de fuente basado en rem -->
<div class="box-rem">Fuente basada en rem (2rem)</div>
</body>
</html>
```



3.13. Diseño responsivo

El diseño responsivo (o responsive design) es una técnica de diseño web que permite que una página se adapte y se vea bien en diferentes dispositivos y tamaños de pantalla, como móviles, tablets y computadoras de escritorio. Utiliza layouts flexibles, imágenes adaptativas y media queries para ajustar el contenido según las dimensiones del viewport.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
```



```
<meta name="viewport" content="width=device-width,
initial-scale=1.0">
<link rel="StyleSheet" href="css/normalize.css"
type="text/css">
<title>Ejemplo de Diseño Responsivo</title>
<style>
  /* Estilos básicos */
  body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
  }

  .container {
    width: 100%;
    padding: 20px;
    background-color: lightgray;
    text-align: center;
  }

  /* Estilo para pantallas de escritorio */
  @media (min-width: 1024px) {
    .container {
      background-color: lightcoral;
    }
  }

  /* Estilo para tablets (pantallas medianas) */
  @media (min-width: 768px) and (max-width: 1023px) {
    .container {
      background-color: lightblue;
    }
  }

  /* Estilo para móviles (pantallas pequeñas) */
  @media (max-width: 767px) {
    .container {
      background-color: lightgreen;
    }
  }
```



```
}  
</style>  
</head>  
<body>  
  <div class="container">  
    <h1>Diseño Responsivo</h1>  
    <p>El color de fondo cambia según el tamaño de la  
pantalla.</p>  
  </div>  
</body>  
</html>
```



3.14. Flexbox vs Grid Layout en CSS

Flexbox y CSS Grid son dos potentes sistemas de diseño en CSS que permiten organizar y distribuir elementos dentro de un contenedor. Aunque tienen algunas similitudes, están diseñados para diferentes propósitos y se utilizan de manera distinta según el tipo de layout que se quiera crear.

| Característica | Flexbox | CSS Grid |
|-----------------------------|---|---|
| Dimensionalidad | Unidimensional (eje horizontal o vertical) | Bidimensional (filas y columnas) |
| Ideal para | Layouts simples y alineación de elementos | Layouts complejos y con estructura de rejilla |
| Uso típico | Menús, barras de navegación, tarjetas, etc. | Tableros, galerías, grids de fotos, etc. |
| Espacio entre elementos | Se ajusta automáticamente según el espacio disponible en un eje | Se puede definir el espacio entre filas y columnas explícitamente |
| Control de filas y columnas | No tiene control directo sobre filas y columnas | Control total sobre filas y columnas |

Ejemplo de flexbox



```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css" type="text/css">
  <title>Layout Completo con Flexbox</title>
  <style>
    body {
      margin: 0;
      font-family: Arial, sans-serif;
    }

    /* Contenedor principal */
    .container {
      display: flex;
      flex-direction: column; /* Organización en columna (cabecera
- contenido - pie) */
      min-height: 100vh; /* Altura mínima para que ocupe toda la
ventana */
    }

    /* Estilo de la cabecera */
    .header {
      background-color: #4CAF50;
      color: white;
      padding: 20px;
      text-align: center;
    }

    /* Contenido principal con barra lateral y contenido */
    .main {
      display: flex;
```



```
        flex: 1; /* El contenido principal ocupa todo el espacio
disponible */
    }
    /* Barra Lateral */
    .sidebar {
        background-color: #f4f4f4;
        width: 250px;
        padding: 20px;
    }

    /* Contenido central */
    .content {
        flex: 1;
        background-color: #ddd;
        padding: 20px;
    }

    /* Pie de página */
    .footer {
        background-color: #333;
        color: white;
        padding: 10px;
        text-align: center;
    }
</style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>Mi Sitio Web</h1>
            <p>Cabecera</p>
        </div>
        <div class="main">
            <div class="sidebar">
```



```
<h2>Barra Lateral</h2>
<p>Contenido de la barra lateral.</p>
</div>
<div class="content">
  <h2>Contenido Principal</h2>
  <p>Este es el contenido principal de la página.</p>
</div>
</div>
<div class="footer">
  <p>Pie de Página</p>
</div>
</div>
</body>
</html>
```

127.0.0.1:5500/flexbox.html

Comenzar a usar Firefox 404 Not Found Otros marcadores

Mi Sitio Web
Cabecera

| | |
|--|--|
| Barra Lateral Contenido de la barra lateral. | Contenido Principal Este es el contenido principal de la página. |
|--|--|

Pie de Página

Ejemplo css grid

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css" type="text/css">
```



```
<title>Layout Completo con CSS Grid</title>
<style>
  body {
    margin: 0;
    font-family: Arial, sans-serif;
  }

  /* Contenedor principal utilizando Grid */
  .container {
    display: grid;
    grid-template-columns: 250px 1fr; /* Dos columnas: 250px
para la barra lateral, el resto para el contenido */
    grid-template-rows: auto 1fr auto; /* Tres filas: auto para
la cabecera y pie, 1fr para el contenido principal */
    grid-template-areas:
      "header header"
      "sidebar content"
      "footer footer";
    min-height: 100vh;
  }

  /* Estilos de las áreas de la cabecera, contenido, barra
lateral y pie */
  .header {
    grid-area: header;
    background-color: #4CAF50;
    color: white;
    padding: 20px;
    text-align: center;
  }

  .sidebar {
    grid-area: sidebar;
    background-color: #f4f4f4;
    padding: 20px;
  }

  .content {
    grid-area: content;
    background-color: #ddd;
    padding: 20px;
  }

  .footer {
```

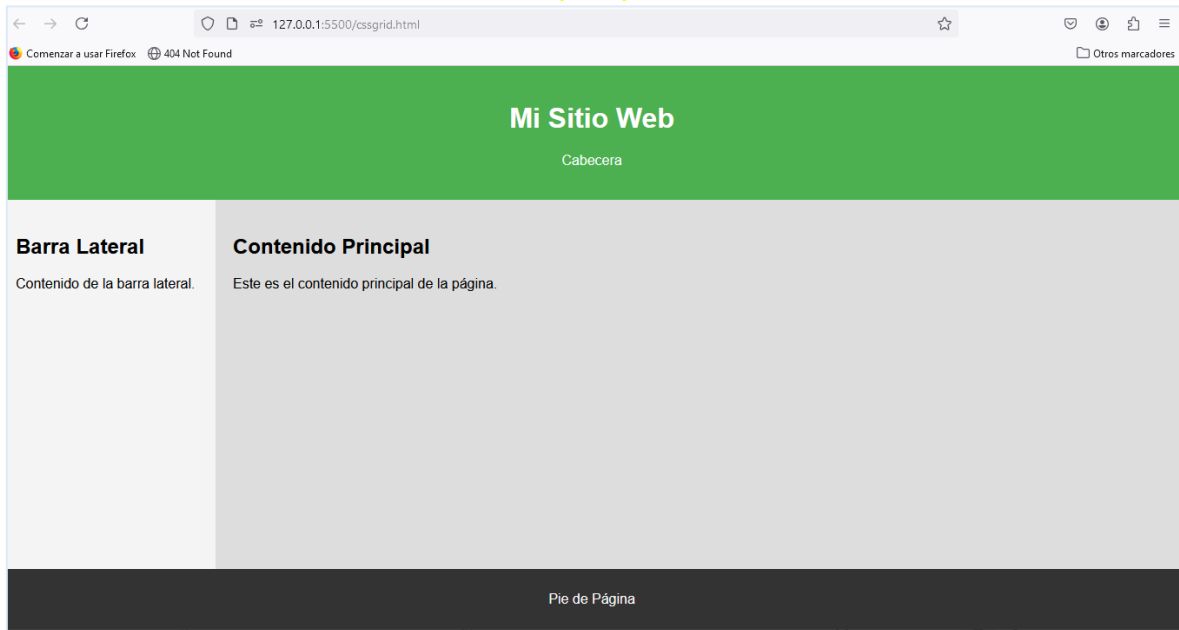



```
        grid-area: footer;
        background-color: #333;
        color: white;
        text-align: center;
        padding: 10px;
    }
</style>
</head>
<body>
    <div class="container">
        <div class="header">
            <h1>Mi Sitio Web</h1>
            <p>Cabecera</p>
        </div>

        <div class="sidebar">
            <h2>Barra Lateral</h2>
            <p>Contenido de la barra lateral.</p>
        </div>

        <div class="content">
            <h2>Contenido Principal</h2>
            <p>Este es el contenido principal de la página.</p>
        </div>

        <div class="footer">
            <p>Pie de Página</p>
        </div>
    </div>
</body>
</html>
```



Otro ejemplo:

https://developer.mozilla.org/en-US/docs/Web/CSS/CSS_grid_layout

3.15. Selectores

Son la forma en que identificamos los elementos HTML a los que queremos aplicar estilos. A continuación, se presenta los principales tipos de selectores con su respectivo ejemplo:

3.15.1. Selector de tipo o etiqueta: Aplica estilos a todos los elementos de un mismo tipo o etiqueta HTML.

```
p {  
  color: blue;  
  font-size: 16px;  
}
```

3.15.2. Selector de clase: Aplica estilos a uno o más elementos que tienen una clase específica.

```
.highlight {  
  background-color: yellow;  
  font-weight: bold;  
}
```



```
}
```

3.15.3. Selector de ID: Aplica estilos a un elemento específico que tiene un ID único.

```
#main-title {  
    font-size: 24px;  
    text-align: center;  
}
```

3.15.4. Selector universal: Aplica estilos a todos los elementos de la página.

```
* {  
    margin: 0;  
    padding: 0;  
}
```

3.15.5. Selector de atributo: Aplica estilos a elementos que tienen un atributo específico (como href, src, type, etc.).

```
a[href*="https"] {  
    color: green;  
}
```

3.15.6. Selector descendente: Aplica estilos a elementos que son descendientes de un elemento padre.

```
div p {  
    color: red;  
}
```

3.15.7. Selector de hijo directo: Aplica estilos a elementos que son hijos directos de un elemento padre.

```
div > p {  
    font-style: italic;  
}
```



3.15.8. Selector de pseudo-clases: Aplica estilos en base a un estado específico de un elemento.

3.15.8.1. Dinámicas (interacción del usuario)

3.15.8.1.1 :hover: Aplica un estilo cuando el usuario pasa el ratón sobre un elemento.

```
a:hover {  
  color: red;  
}
```

3.15.8.1.2 :focus: Aplica un estilo a un elemento cuando recibe foco, como un campo de formulario al seleccionarlo.

```
input:focus {  
  outline: none;  
  border-color: blue;  
}
```

3.15.8.1.3. :active: Aplica un estilo mientras un elemento es activado por el usuario, como al hacer clic en un enlace o botón.

```
button:active {  
  background-color: green;  
}
```

3.15.8.2. Estructurales (posición dentro del DOM)

3.15.8.2.1. :first-child: Selecciona el primer hijo de un elemento padre.

```
p:first-child {  
  color: blue;  
}
```

3.15.8.2.2. :last-child: Selecciona el último hijo de un elemento padre.

```
p:last-child {  
  color: green;  
}
```

3.15.8.2.3. :nth-child(n): Selecciona el hijo n-ésimo de un elemento padre.



```
p:nth-child(2) {  
  color: orange;  
}
```

3.15.8.2.4. :nth-last-child(n): Selecciona el n-ésimo hijo contando desde el final.

```
p:nth-last-child(2) {  
  color: purple;  
}
```

3.15.8.2.5. :nth-of-type(n): Selecciona el n-ésimo elemento de un tipo específico.

```
p:nth-of-type(2) {  
  color: teal;  
}
```

3.15.8.2.6. :first-of-type: Selecciona el primer elemento de un tipo específico.

```
p:first-of-type {  
  color: pink;  
}
```

3.15.8.2.7. :last-of-type: Selecciona el último elemento de un tipo específico.

```
p:last-of-type {  
  color: brown;  
}
```

3.15.8.2.8. :only-child: Selecciona un elemento si es el único hijo de su padre.

```
p:only-child {  
  color: purple;  
}
```

3.15.8.2.9. :only-of-type: Selecciona un elemento si es el único de su tipo en su padre.

```
p:only-of-type {  
  color: magenta;  
}
```



3.15.8.2.10. :empty: Selecciona elementos que no tienen hijos (ni texto ni elementos hijos).

```
div:empty {  
    background-color: gray;  
}
```

3.15.8.3. Pseudo-clases basadas en el estado del formulario

3.15.8.3.1. :checked: Selecciona inputs que están marcados (checkbox o radio).

```
input:checked {  
    border-color: green;  
}
```

3.15.8.3.2. :disabled: Selecciona inputs que están deshabilitados.

```
input:disabled {  
    background-color: lightgray;  
}
```

3.15.8.3.3. :enabled: Selecciona inputs que están habilitados.

```
input:enabled {  
    border-color: blue;  
}
```

3.15.8.3.4. :required: Selecciona inputs que son requeridos.

```
input:required {  
    border: 2px solid red;  
}
```

3.15.8.3.5. :optional: Selecciona inputs que no son requeridos.

```
input:optional {  
    border: 2px solid green;  
}
```

3.15.8.4. Relacionales



3.15.8.4.1. :not(selector): Selecciona todos los elementos que no coinciden con el selector dado.

```
p:not(.highlight) {  
  color: darkgray;  
}
```

3.15.8.4.2. :is(selector): Selecciona los elementos que coincidan con cualquiera de los selectores en la lista.

```
:is(h1, h2, h3) {  
  color: blue;  
}
```

3.15.8.4.3. :where(selector): Similar a :is(), pero con menor especificidad.

```
:where(h1, h2, h3) {  
  color: purple;  
}
```

3.15.8.4.4. :has(selector): Selecciona elementos si contienen un elemento que coincide con el selector.

```
div:has(img) {  
  border: 2px solid red;  
}
```

3.15.8.5. Pseudo-clases UI específicas

3.15.8.5.1. :root: Selecciona el elemento raíz (normalmente el <html>).

```
:root {  
  --main-color: #333;  
}
```

3.15.8.5.2. :lang(language): Selecciona elementos de acuerdo con su atributo de idioma.

```
p:lang(en) {  
  font-style: italic;  
}
```



3.15.9. Operador de combinación de elementos hermanos adyacentes (+): Aplica estilos a un elemento que es hermano inmediatamente adyacente a otro elemento.

```
h2 + p {  
    color: purple;  
}
```

3.15.10. Operador de combinación general de elementos hermanos (~): Aplica estilos a todos los elementos hermanos que aparecen después de otro elemento específico en el mismo nivel.

```
h2 ~ p {  
    font-size: 14px;  
    color: brown;  
}
```

3.15.11. Los pseudo-elementos: permiten seleccionar y estilizar partes específicas de un elemento, como la primera letra, la primera línea, o insertar contenido antes o después de un elemento. Son muy útiles para crear efectos sin agregar más HTML.

3.15.11.1. ::before y ::after: Se usan para insertar contenido antes o después del contenido de un elemento.

```
h2::before {  
    content: "👉 "; /* Añade un ícono o texto antes del h2 */  
}  
h2::after {  
    content: "💡"; /* Añade un ícono o texto después del h2 */  
}
```

3.15.11.2. ::first-letter: Aplica estilos solo a la primera letra de un elemento de bloque.

```
p::first-letter {  
    font-size: 24px;  
    font-weight: bold;  
    color: red;  
}
```




3.15.11.3. ::first-line: Aplica estilos solo a la primera línea de texto de un elemento de bloque.

```
p::first-line {  
    font-style: italic;  
    color: darkblue;  
}
```

3.15.11.4. ::selection: Cambia el color de fondo y el color del texto cuando el usuario selecciona el contenido con el cursor.

```
::selection {  
    background-color: yellow;  
    color: black;  
}
```

3.15.11.5. ::placeholder: Aplica estilo al texto del marcador de posición en un campo de entrada.

```
input::placeholder {  
    color: gray;  
    font-style: italic;  
}
```

3.15.11.6. ::marker: Estiliza los marcadores de listas (viñetas o números).

```
li::marker {  
    color: red;  
    font-size: 20px;  
}
```

3.15.11.7. ::backdrop: Estiliza el fondo de elementos modales como <dialog>.

```
dialog::backdrop {  
    background-color: rgba(0, 0, 0, 0.8);  
}
```

3.15.11.8. ::file-selector-button: Estiliza el botón de selección de archivos en un <input type="file">.

```
input[type="file"]::file-selector-button {
```



```
background-color: lightblue;
border: 1px solid #333;
}
```

Ejemplo:

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <link rel="StyleSheet" href="css/normalize.css" type="text/css">
  <title>Ejemplo: Pseudo-clases, Pseudo-elementos y
Selectores</title>
  <style>
    /*-----
    SELECTORES BÁSICOS
    -----*/
    /* Selector universal: Aplica a todos los elementos */
    * {
      margin: 0;
      padding: 0;
      box-sizing: border-box;
    }

    /* Selector de etiqueta: Aplica a todas las etiquetas <body>
    */
    body {
      font-family: Arial, sans-serif;
      line-height: 1.6;
      padding: 20px;
    }

    /* Selector de clase: Aplica a todos los elementos con clase
    "highlight" */
    .highlight {
      background-color: lightyellow;
      border: 1px solid #333;
    }
  </style>
</head>
<body>
```



```
/* Selector de ID: Aplica al elemento con ID "titulo-
principal" */
#titulo-principal {
    text-align: center;
    color: darkblue;
    font-size: 28px;
}

/* Selector de atributo: Aplica a los elementos <input> con
atributo placeholder */
input[placeholder] {
    border: 1px solid lightgray;
}

/* Selector descendiente: Aplica a los <li> que son
descendientes de <ul> */
ul li {
    margin-bottom: 5px;
}

/* Selector hijo directo: Aplica a <li> que son hijos directos
de <ul> */
ul > li {
    list-style-type: square;
}

/* Selector hermano adyacente (+): Estiliza el párrafo que
sigue directamente a un <h2> */
h2 + p {
    color: darkred;
}

/* Selector hermanos generales (~): Aplica estilo a todos los
párrafos hermanos después de <h2> */
h2 ~ p {
    font-style: italic;
    color: navy;
}

/*-----
PSEUDO-CLASES Y PSEUDO-ELEMENTOS
-----*/
```



```
/* Pseudo-elemento ::before y ::after: Inserta contenido antes
y después de los títulos */
h2::before {
  content: "💡 ";
}

h2::after {
  content: "💡";
}

/* Pseudo-elemento ::first-letter: Estiliza la primera letra
del párrafo */
p::first-letter {
  font-size: 24px;
  font-weight: bold;
  color: red;
}

/* Pseudo-elemento ::first-line: Estiliza la primera línea del
párrafo */
p::first-line {
  font-style: italic;
  color: darkblue;
}

/* Pseudo-elemento ::selection: Estiliza el texto seleccionado
*/
::selection {
  background-color: yellow;
  color: black;
}

/* Pseudo-clase :hover: Aplica al pasar el ratón por encima de
los enlaces */
a:hover {
  color: red;
}

/* Pseudo-clase :focus: Aplica al recibir foco */
input:focus {
  outline: none;
  border-color: blue;
}
```



```
/* Pseudo-clase :nth-child(n): Aplica al segundo hijo */
p:nth-child(2) {
    color: orange;
}

/* Pseudo-clase :checked: Aplica a checkboxes seleccionados */
input:checked {
    border: 2px solid green;
}

/* Pseudo-clase :disabled: Aplica a inputs deshabilitados */
input:disabled {
    background-color: lightgray;
}

/* Pseudo-clase :last-child y :first-child: Aplica al primer y
último hijo */
p:first-child {
    color: blue;
}

p:last-child {
    color: green;
}

/* Pseudo-elemento ::placeholder: Estiliza el texto del
placeholder de los inputs */
input::placeholder {
    color: gray;
    font-style: italic;
}

/* Pseudo-elemento ::marker: Estiliza los marcadores de la
lista */
li::marker {
    color: red;
    font-size: 20px;
}

/*-----
DISEÑO RESPONSIVO
-----*/

/* Estilos para pantallas pequeñas (móviles) */
```



```
@media (max-width: 600px) {  
  body {  
    background-color: lightblue;  
  }  
  
  #titulo-principal {  
    font-size: 20px;  
    color: darkred;  
  }  
  
  /* Ajustar el tamaño del texto en dispositivos pequeños */  
  p {  
    font-size: 14px;  
  }  
  
  input {  
    width: 100%;  
  }  
}  
  
/* Estilos para pantallas medianas (tabletas) */  
@media (min-width: 601px) and (max-width: 1024px) {  
  body {  
    background-color: lightgreen;  
  }  
  
  #titulo-principal {  
    font-size: 24px;  
  }  
  
  p {  
    font-size: 16px;  
  }  
}  
  
/* Estilos para pantallas grandes (escritorios) */  
@media (min-width: 1025px) {  
  body {  
    background-color: white;  
  }  
  
  #titulo-principal {  
    font-size: 28px;  
  }  
}
```



```
p {
  font-size: 18px;
}

</style>
</head>
<body>

  <!-- Título principal con ID -->
  <h2 id="titulo-principal">Ejemplo Completo de Selectores,
Pseudo-clases y Pseudo-elementos</h2>

  <!-- Párrafos con pseudo-elemento ::first-letter y ::first-line
-->
  <p>Este es el primer párrafo. Su primera letra y la primera
línea tienen estilos especiales.</p>

  <!-- Párrafo con clase highlight y con ::selection -->
  <p class="highlight">Este párrafo tiene una clase especial
llamada .highlight y será resaltado al seleccionarlo.</p>

  <!-- Párrafos con nth-child y combinadores de hermanos
adyacentes y generales -->
  <p>Este es un segundo párrafo estilizado con :nth-child y es el
primer hermano adyacente de un h2.</p>
  <p>Este es un tercer párrafo que será estilizado con el
combinador de hermanos generales (~).</p>

  <!-- Enlaces con pseudo-clase :hover -->
  <p>Visita <a href="https://www.ejemplo.com">mi sitio seguro</a>
o <a href="http://www.ejemplo.com">mi sitio no seguro</a>.</p>

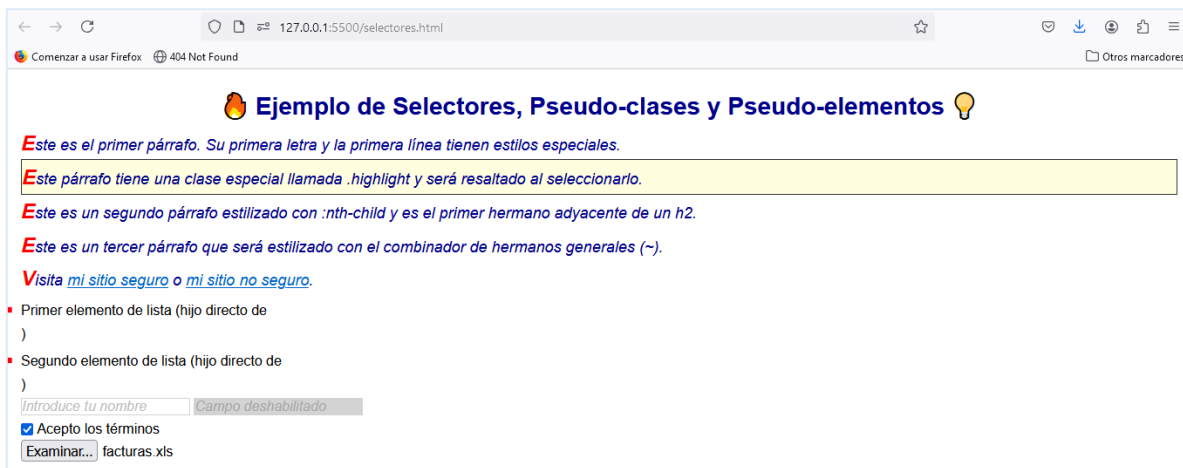
  <!-- Lista con pseudo-elemento ::marker y selectores de hijos
directos y descendientes -->
  <ul>
    <li>Primer elemento de lista (hijo directo de <ul>)</li>
    <li>Segundo elemento de lista (hijo directo de <ul>)</li>
  </ul>

  <!-- Campos de formulario con pseudo-clases y pseudo-elementos -
->
  <form>
```



```
<input type="text" placeholder="Introduce tu nombre">
<input type="text" placeholder="Campo deshabilitado" disabled>
<br>
<input type="checkbox" checked> Acepto los términos
<br>
<input type="file">
</form>

</body>
</html>
```



Ver más:

https://developer.mozilla.org/es/docs/Learn/CSS/Building_blocks/Selectors

IV. Materialize

Es un framework CSS basado en Material Design de Google, que facilita la creación de interfaces de usuario modernas y receptivas. Su propósito es simplificar el desarrollo web proporcionando componentes prediseñados, estilos y una estructura coherente que sigue los principios de diseño de Google.

Ver documentación oficial: <https://materializecss.com/>

1. Instalación

Para el uso de Materialize tenemos varias opciones:

1.1. Usar CDN (Red de distribución de contenido):



es la manera más rápida y simple de usar Materialize, no es necesario descargar archivos. Deben agregarse los enlaces al archivo HTML.

```
<head>
  <!-- Materialize CSS -->
  <link rel="stylesheet"
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css
/materialize.min.css">
  <!-- Importar Material Icons -->
  <link
href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
</head>
```

Se Agrega el enlace para el JavaScript justo antes de cerrar la etiqueta </body>:

```
<!-- Materialize JS -->

<script
src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/m
aterialize.min.js"></script>
```

En el caso de ser necesario pueden inicializarse componentes de javascript con este código, debajo del enlace a Materialize JS:

```
<script>
  document.addEventListener('DOMContentLoaded', function() {
    var elems = document.querySelectorAll('.modal');
    var instances = M.Modal.init(elems);
  });
</script>
```

1.2. Descargando los Archivos

Si se prefiere trabajar de manera local:

Nos dirigimos a la pagina: <https://materializecss.com/getting-started.html>

Se da click en el botón Download para descargar el archivo .zip con los archivos necesarios.

Se extrae el contenido del .zip y copia las carpetas css y js en el proyecto.

En el archivo HTML, se enlazan los archivos locales de Materialize:

```
<head>
  <!-- Enlace al archivo local de Materialize CSS -->
```



```
<link rel="stylesheet" href="css/materialize.min.css">
<!-- Importar Material Icons -->
<link
href="https://fonts.googleapis.com/icon?family=Material+Icons"
rel="stylesheet">
</head>
<body>
  <!-- Tu contenido aquí -->
  <!-- Enlace al archivo local de Materialize JS -->
  <script src="js/materialize.min.js"></script>
</body>
```

1.3. Usar npm para Instalar Materialize

Si se usa npm dentro del proyecto, en una terminal ubicado en la carpeta del proyecto se ejecuta el siguiente comando para instalar Materialize:

```
npm install materialize-css
```

Luego, en el archivo JavaScript principal (por ejemplo, app.js), se importa el CSS y JavaScript de Materialize:

```
import 'materialize-css/dist/css/materialize.min.css';
import 'materialize-css/dist/js/materialize.min.js';
```

En caso de ser necesario se inicializan los componentes, igual que en los otros métodos:

```
document.addEventListener('DOMContentLoaded', function() {
  var elems = document.querySelectorAll('.modal');
  var instances = M.Modal.init(elems);
});
```

2. Características Principales de Materialize

2.1. Grid System (Sistema de cuadrícula): Al igual que otros frameworks CSS (como Bootstrap), Materialize usa un sistema de cuadrícula flexible basado en columnas que facilita el diseño responsivo.

2.2. Componentes:

2.2.1. Botones: Incluyen varios estilos y tamaños.

2.2.2. Tarjetas (Cards): Para mostrar contenido de manera organizada.

2.2.3. Modales: Ventanas emergentes fáciles de implementar.



2.2.4. Navbar (Barra de navegación): Barra de navegación prediseñada y adaptable.

2.2.5. Forms: Campos de formulario con validación y estilos modernos.

2.3. JavaScript: Incluye funcionalidades listas para usar como deslizadores (sliders), colapsables, modales y parallax scrolling, entre otros.

2.4. Simplicidad: Es fácil de integrar y requiere poco conocimiento para empezar. Ofrece documentación detallada con ejemplos.

2.5. Diseño Responsivo: Todos los componentes están preparados para funcionar bien en cualquier tamaño de pantalla.

3. Ejemplos Básicos

3.1. Grid System (Sistema de Cuadrícula)

```
<div class="row">
  <div class="col s12 m6 l4">Columna 1</div>
  <div class="col s12 m6 l4">Columna 2</div>
  <div class="col s12 m6 l4">Columna 3</div>
</div>
```

s12, m6, l4: Estos representan el tamaño de la columna en diferentes tamaños de pantalla (pequeña, mediana y grande).

3.2. Botones

```
<a class="waves-effect waves-light btn">Botón normal</a>
<a class="btn-floating btn-large waves-effect waves-light red"><i
class="material-icons">add</i></a>
```

Los botones pueden ser simples o flotantes. El botón flotante es un pequeño botón circular que destaca más en el diseño.

3.3. Tarjetas (Cards)

```
<div class="card">
  <div class="card-image">
    
    <span class="card-title">Título de la tarjeta</span>
  </div>
  <div class="card-content">
    <p>Contenido de la tarjeta aquí.</p>
  </div>
  <div class="card-action">
    <a href="#">Enlace de acción</a>
  </div>
</div>
```



```
</div>  
</div>
```

3.4. Navbar (Barra de navegación)

```
<nav>  
  <div class="nav-wrapper">  
    <a href="#" class="brand-logo">Logo</a>  
    <ul id="nav-mobile" class="right hide-on-med-and-down">  
      <li><a href="#">Inicio</a></li>  
      <li><a href="#">Sobre mí</a></li>  
      <li><a href="#">Contacto</a></li>  
    </ul>  
  </div>  
</nav>
```

Ejemplo:

```
<!DOCTYPE html>  
<html lang="es">  
<head>  
  <meta charset="UTF-8">  
  <meta name="viewport" content="width=device-width, initial-  
scale=1.0">  
  <title>Portafolio Web con Modal</title>  
  <!-- Importar Materialize CSS -->  
  <link  
href="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/css  
/materialize.min.css" rel="stylesheet">  
  <!-- Importar Material Icons -->  
  <link  
href="https://fonts.googleapis.com/icon?family=Material+Icons"  
rel="stylesheet">  
</head>  
<body>  
  
  <!-- Navbar -->  
  <nav>  
    <div class="nav-wrapper blue">  
      <a href="#" class="brand-logo">Mi Portafolio</a>  
      <ul id="nav-mobile" class="right hide-on-med-and-down">  
        <li><a href="#about">Sobre mí</a></li>  
        <li><a href="#projects">Proyectos</a></li>  
        <li><a href="#contact">Contacto</a></li>  
      </ul>
```



```
</div>
</nav>

<!-- Sección "Sobre mí" -->
<section id="about" class="section">
  <div class="container">
    <h3 class="center-align">Sobre mí</h3>
    <div class="row">
      <div class="col s12 m6">
        
      </div>
      <div class="col s12 m6">
        <p>¡Hola! Soy un desarrollador web apasionado por crear
aplicaciones modernas y funcionales. Tengo experiencia en HTML,
CSS, JavaScript y frameworks como Materialize.</p>
      </div>
    </div>
  </div>
</section>

<!-- Sección "Proyectos" -->
<section id="projects" class="section grey lighten-4">
  <div class="container">
    <h3 class="center-align">Proyectos</h3>
    <div class="row">
      <div class="col s12 m6 l4">
        <div class="card">
          <div class="card-image">
            
            <span class="card-title">Proyecto 1</span>
          </div>
          <div class="card-content">
            <p>Descripción breve del proyecto 1. Este proyecto
fue desarrollado utilizando tecnologías modernas.</p>
          </div>
          <div class="card-action">
            <a href="#modal1" class="modal-trigger">Ver más
detalles</a>
          </div>
        </div>
      </div>
    </div>
  </div>
</section>
```







```
&ixlib=rb-1.2.1&q=80&w=400" alt="Proyecto 1" class="responsive-
img">
</div>
<div class="modal-footer">
  <a href="#" class="modal-close waves-effect waves-green
btn-flat">Cerrar</a>
</div>
</div>

<div id="modal2" class="modal">
  <div class="modal-content">
    <h4>Proyecto 2 - Detalles</h4>
    <p>Este proyecto fue desarrollado utilizando frameworks
modernos para frontend como React y Materialize.</p>
    
    </div>
    <div class="modal-footer">
      <a href="#" class="modal-close waves-effect waves-green
btn-flat">Cerrar</a>
    </div>
  </div>

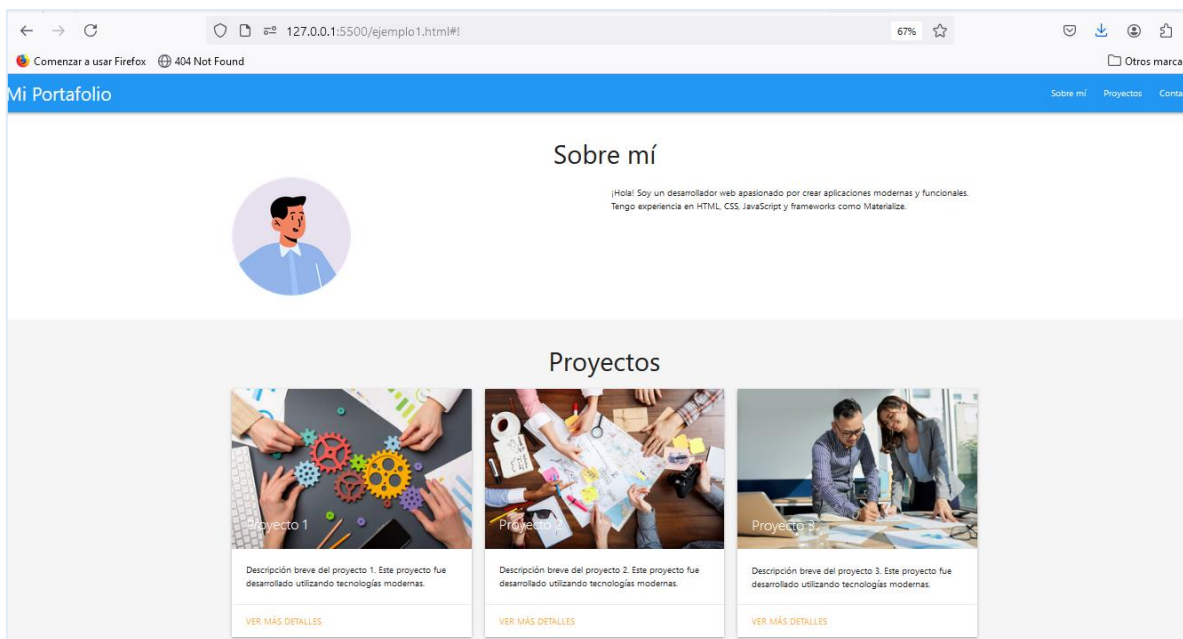
  <div id="modal3" class="modal">
    <div class="modal-content">
      <h4>Proyecto 3 - Detalles</h4>
      <p>Este proyecto es un sitio web completamente responsivo
que se adapta a todas las pantallas.</p>
      
      </div>
      <div class="modal-footer">
        <a href="#" class="modal-close waves-effect waves-green
btn-flat">Cerrar</a>
      </div>
    </div>

    <!-- Importar Materialize JS -->
```




```
<script
src="https://cdnjs.cloudflare.com/ajax/libs/materialize/1.0.0/js/m
aterialize.min.js"></script>

<!-- Inicializar modales -->
<script>
  document.addEventListener('DOMContentLoaded', function() {
    var elems = document.querySelectorAll('.modal');
    var instances = M.Modal.init(elems);
  });
</script>
</body>
</html>
```



V. Bootstrap

Bootstrap es un framework de código abierto que facilita el desarrollo de sitios web y aplicaciones web responsive. Utiliza HTML, CSS y JavaScript para proporcionar una colección de componentes predefinidos que se adaptan automáticamente a diferentes tamaños de pantalla (móviles, tabletas y computadoras de escritorio).

1. Instalación con CDN (Content Delivery Network)



En este método solo es necesario incluir los enlaces a Bootstrap en el archivo HTML, esto incluirá automáticamente las bibliotecas CSS y JS de Bootstrap desde un CDN, permitiéndote usar las clases y componentes de Bootstrap.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Mi sitio con Bootstrap</title>
  <!-- Bootstrap CSS -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>
  <!-- Aquí va tu contenido -->
  <!-- Bootstrap JS (opcional si quieres usar componentes como
modal, tooltip, etc.) -->
  <script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```

2. Instalación mediante descarga local

Es posible descargar los archivos de Bootstrap directamente y alojarlos localmente, para ello es necesario ir al sitio oficial, efectuar la descarga, descomprimir y vincular al proyecto.

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Mi sitio con Bootstrap</title>
  <!-- Bootstrap CSS (local) -->
  <link rel="stylesheet" href="ruta-a-
bootstrap/css/bootstrap.min.css">
</head>
<body>
  <!-- Aquí va tu contenido -->
  <!-- Bootstrap JS (local) -->
```



```
<script src="ruta-a-  
bootstrap/js/bootstrap.bundle.min.js"></script>  
</body>  
</html>
```

3. Instalación con NPM (Node Package Manager)

Si se está trabajando en un entorno con Node.js y un gestor de paquetes, es posible instalar Bootstrap a través de NPM, ideal para proyectos más grandes. Para ello debe tenerse Node.js y NPM instalados, se abre una terminal en el directorio del proyecto y se ejecuta el siguiente comando: **npm install Bootstrap**

Una vez instalado, se enlaza Bootstrap en el archivo JavaScript o CSS principal. Por ejemplo:

```
// Importa Bootstrap en tu archivo JavaScript principal  
import 'bootstrap/dist/css/bootstrap.min.css';  
import 'bootstrap/dist/js/bootstrap.bundle.min.js';
```

4. Instalación con SASS (para personalización avanzada)

Si se desea personalizar Bootstrap, es posible instalarlo usando SASS para compilar solo las partes necesarias de Bootstrap.

Se instala Bootstrap vía NPM como se indicó anteriormente.

Luego, se crea un archivo SASS en el proyecto y se compila para personalizar Bootstrap.

```
// En tu archivo SASS  
@import 'node_modules/bootstrap/scss/bootstrap';
```

Luego se compila usando un preprocesador de SASS para generar el CSS final.

5. Instalación mediante Frameworks o Generadores

Muchos frameworks modernos como Angular, React, Vue.js, presentan formas de integrar Bootstrap. Dependiendo del framework que se use, debe consultarse la documentación de integración de Bootstrap.

6. Características principales



6.1. Sistema de Rejillas: El sistema de rejillas de Bootstrap te permite dividir el contenido en columnas que se ajustan automáticamente en diferentes dispositivos.

```
<div class="container">
  <div class="row">
    <div class="col-md-4">Columna 1 (4/12)</div>
    <div class="col-md-4">Columna 2 (4/12)</div>
    <div class="col-md-4">Columna 3 (4/12)</div>
  </div>
</div>
```

La clase **col-md-4** divide la fila en tres columnas de igual tamaño (4 columnas cada una en una cuadrícula de 12). El contenido se ajusta de forma responsiva en pantallas medianas ($\geq 768\text{px}$).

6.2. Componentes (Botones, Formularios, Tarjetas)

Bootstrap incluye componentes listos para usar como botones, formularios y tarjetas.

```
<!-- Botón -->
<button class="btn btn-primary">Botón Primario</button>

<!-- Tarjeta -->
<div class="card" style="width: 18rem;">
  
  <div class="card-body">
    <h5 class="card-title">Título de la tarjeta</h5>
    <p class="card-text">Esta es una breve descripción dentro de la
    tarjeta.</p>
    <a href="#" class="btn btn-primary">Ir a algún lugar</a>
  </div>
</div>
```

Los botones (btn btn-primary) son fácilmente personalizables con diferentes estilos y las tarjetas (card) son contenedores que organizan contenido con imagen, texto y un botón.

6.3. Compatibilidad con Navegadores



Bootstrap funciona bien en los navegadores modernos. Es posible probar los códigos con los diferentes navegadores (Chrome, Firefox, Edge, Safari) y observar cómo el diseño se mantiene consistente.

6.4. Diseño Responsivo (Responsive Design)

Bootstrap ajusta automáticamente el contenido a diferentes tamaños de pantalla, lo que lo hace ideal para dispositivos móviles y de escritorio.

Ejemplo de Rejilla Responsiva:

```
<div class="container">
  <div class="row">
    <div class="col-sm-12 col-md-6 col-lg-4">Columna 1</div>
    <div class="col-sm-12 col-md-6 col-lg-4">Columna 2</div>
    <div class="col-sm-12 col-md-6 col-lg-4">Columna 3</div>
  </div>
</div>
```

En pantallas pequeñas (sm), cada columna ocupa el 100% del ancho.

En pantallas medianas (md), las columnas se dividen en dos.

En pantallas grandes (lg), las tres columnas se muestran en fila.

6.5. Utilidades (Espaciado, Colores, Alineación de Texto)

Las utilidades de Bootstrap permiten aplicar rápidamente estilos comunes sin necesidad de escribir CSS personalizado.

```
<div class="p-5 bg-primary text-white">Espaciado de 5
(Padding)</div>
```

```
<div class="m-3 text-center text-success">Margen de 3 y texto
centrado y en verde</div>
```

.p-5 añade un padding de 5 unidades alrededor del elemento.

.m-3 añade un margen de 3 unidades.

.text-center centra el texto, y .text-success aplica un color verde.

6.6. Ejemplo de un portafolio web básico

```
<!DOCTYPE html>
<html lang="es">
```



```
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Mi Portafolio</title>
  <!-- Enlace a Bootstrap -->
  <link href="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-
alpha1/dist/css/bootstrap.min.css" rel="stylesheet">
</head>
<body>

<!-- Navegación -->
<nav class="navbar navbar-expand-lg navbar-light bg-light">
  <div class="container-fluid">
    <a class="navbar-brand" href="#">Mi Portafolio</a>
    <button class="navbar-toggler" type="button" data-bs-
toggle="collapse" data-bs-target="#navbarNav" aria-
controls="navbarNav" aria-expanded="false" aria-label="Toggle
navigation">
      <span class="navbar-toggler-icon"></span>
    </button>
    <div class="collapse navbar-collapse" id="navbarNav">
      <ul class="navbar-nav ms-auto">
        <li class="nav-item">
          <a class="nav-link active" href="#sobre-mi">Sobre mí</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#proyectos">Proyectos</a>
        </li>
        <li class="nav-item">
          <a class="nav-link" href="#contacto">Contacto</a>
        </li>
      </ul>
    </div>
  </div>
</nav>
```



```
</div>
</nav>

<!-- Sección de Presentación -->
<header class="bg-primary text-white text-center py-5">
  <div class="container">
    <h1 class="display-4">Hola, soy Juan Pérez</h1>
    <p class="lead">Desarrollador Web y Diseñador Gráfico</p>
  </div>
</header>

<!-- Sección de Sobre mí -->
<section id="sobre-mi" class="py-5">
  <div class="container">
    <div class="row">
      <div class="col-lg-6">
        <h2>Sobre mí</h2>
        <p>Soy un desarrollador apasionado por crear soluciones
web modernas, optimizadas y responsivas. Me encanta trabajar con
tecnologías como HTML, CSS, JavaScript y frameworks como Bootstrap
y React.</p>
      </div>
      <div class="col-lg-6">
        
      </div>
    </div>
  </div>
</section>

<!-- Sección de Proyectos -->
<section id="proyectos" class="bg-light py-5">
  <div class="container">
    <h2 class="text-center">Mis Proyectos</h2>
```



```
<div class="row">
  <div class="col-lg-4 mb-4">
    <div class="card">
      
      <div class="card-body">
        <h5 class="card-title">Proyecto 1</h5>
        <p class="card-text">Descripción del proyecto 1.</p>
        <a href="#" class="btn btn-primary">Ver más</a>
      </div>
    </div>
  </div>
  <div class="col-lg-4 mb-4">
    <div class="card">
      
      <div class="card-body">
        <h5 class="card-title">Proyecto 2</h5>
        <p class="card-text">Descripción del proyecto 2.</p>
        <a href="#" class="btn btn-primary">Ver más</a>
      </div>
    </div>
  </div>
  <div class="col-lg-4 mb-4">
    <div class="card">
      
      <div class="card-body">
        <h5 class="card-title">Proyecto 3</h5>
        <p class="card-text">Descripción del proyecto 3.</p>
        <a href="#" class="btn btn-primary">Ver más</a>
      </div>
    </div>
  </div>
</div>
```



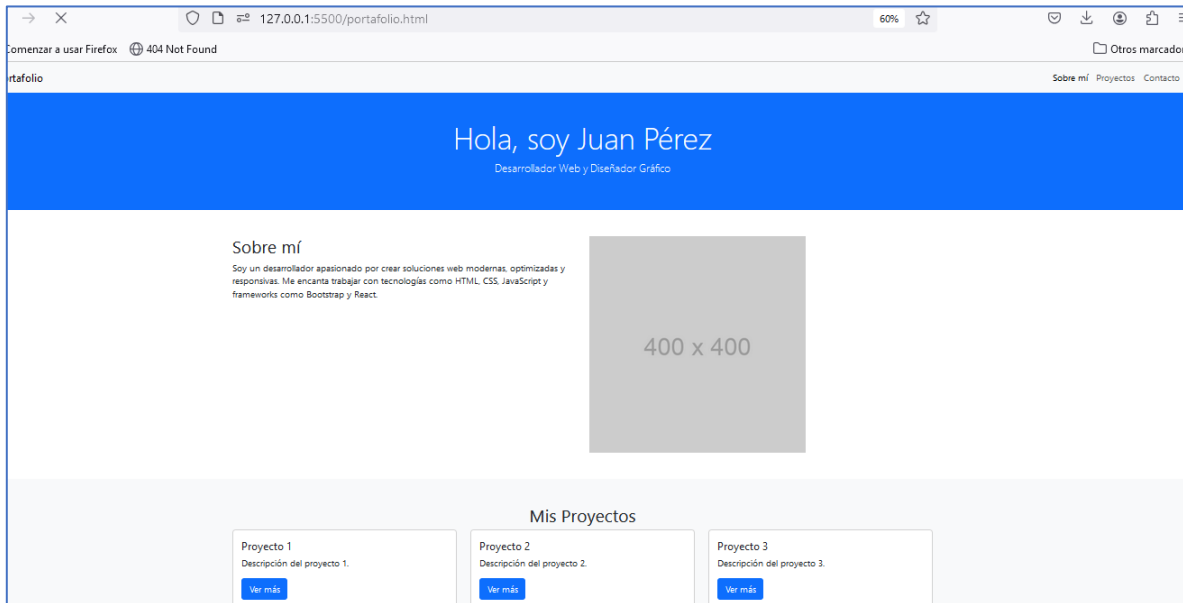

```
</div>
</div>
</section>

<!-- Sección de Contacto -->
<section id="contacto" class="py-5">
  <div class="container">
    <h2 class="text-center">Contáctame</h2>
    <form>
      <div class="row">
        <div class="col-lg-6 mb-3">
          <label for="nombre" class="form-label">Nombre</label>
          <input type="text" class="form-control" id="nombre"
placeholder="Tu nombre">
        </div>
        <div class="col-lg-6 mb-3">
          <label for="email" class="form-label">Correo
Electrónico</label>
          <input type="email" class="form-control" id="email"
placeholder="Tu correo">
        </div>
      </div>
      <div class="mb-3">
        <label for="mensaje" class="form-label">Mensaje</label>
        <textarea class="form-control" id="mensaje" rows="3"
placeholder="Escribe tu mensaje"></textarea>
      </div>
      <button type="submit" class="btn btn-
primary">Enviar</button>
    </form>
  </div>
</section>

<!-- Enlace a Bootstrap JS y dependencias -->
```



```
<script src="https://cdn.jsdelivr.net/npm/bootstrap@5.3.0-alpha1/dist/js/bootstrap.bundle.min.js"></script>
</body>
</html>
```



VI. Tailwind

Tailwind CSS es un framework de CSS basado en utilidades que permite a los desarrolladores construir interfaces de usuario rápidamente sin necesidad de escribir CSS personalizado. A diferencia de otros frameworks como Bootstrap, Tailwind no proporciona componentes predefinidos. En su lugar, ofrece clases utilitarias que se pueden combinar para crear diseños complejos de manera flexible y eficiente.

1. Instalación

1.1. A través de CDN

Dentro del head usamos la siguiente codificación

```
<link
href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwind.min.css" rel="stylesheet">
```

1.1. A través de npm

Primero se debe tener instalado Node.js.



```
npm init -y
```

Se instala Tailwind CSS:

```
npm install -D tailwindcss
```

Se crea el archivo de configuración:

```
npx tailwindcss init
```

Configura Tailwind para que procese tu CSS. En tu archivo input.css:

```
@tailwind base;
```

```
@tailwind components;
```

```
@tailwind utilities;
```

Compila tu CSS usando Tailwind:

```
npx tailwindcss -i ./input.css -o ./output.css -watch
```

2. Características generales:

2.1. Clases Utilitarias: Tailwind CSS se basa en clases pequeñas y atómicas, como `bg-blue-500`, `text-center`, `p-4`, que te permiten aplicar estilos directamente en el HTML.

2.2. Personalizable: Ofrece un sistema de configuración (archivo `tailwind.config.js`) donde puedes personalizar el diseño según las necesidades de tu proyecto, como colores, tamaños, tipografías, etc.

2.3. Responsive Design: Tailwind facilita la creación de diseños responsivos con su sistema de breakpoints (`sm`, `md`, `lg`, `xl`, etc.).

2.4. PurgeCSS Integrado: Tailwind incluye PurgeCSS para eliminar automáticamente el CSS no utilizado en producción, reduciendo el tamaño del archivo CSS final.

2.5. Plugins: Es extensible a través de plugins que añaden utilidades adicionales o características como formularios, tipografía, etc.

3. Ejemplos de Clases

3.1. Colores:

```
<div class="bg-blue-500 text-white p-4">  
  Esto es un fondo azul con texto blanco.  
</div>
```



3.2. Tamaño de Texto:

```
<p class="text-xl">Texto de tamaño grande. </p>
<p class="text-sm">Texto de tamaño pequeño. </p>
```

3.3. Márgenes y Padding:

```
<div class="m-4 p-6 bg-gray-100">Contenido con margen y
padding.</div>
```

3.4. Flexbox:

```
<div class="flex justify-center items-center h-screen">
  <div class="bg-green-500 p-10">Centrado horizontal y
  verticalmente.</div>
</div>
```

3.5. Diseño Responsivo:

```
<div class="bg-red-500 sm:bg-blue-500 md:bg-green-500 lg:bg-yellow-
500">
  Cambia de color según el tamaño de la pantalla.
</div>
```

3.6. Sombra y Bordes Redondeados:

```
<div class="shadow-lg rounded-lg p-6 bg-white">
  Caja con sombra y bordes redondeados.
</div>
```

4. Ejemplo de portafolio

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-
scale=1.0">
  <title>Mi Portafolio</title>
```



```
<link
href="https://cdn.jsdelivr.net/npm/tailwindcss@2.2.19/dist/tailwin
d.min.css" rel="stylesheet">
</head>
<body class="bg-gray-100 text-gray-800">
  <!-- Header -->
  <header class="bg-gray-900 text-white p-5 flex justify-between
items-center">
    <div class="flex items-center">
      
      <h1 class="text-3xl font-bold">Mi Portafolio</h1>
    </div>
  </header>

  <!-- Sección Acerca de Mí -->
  <section class="py-10">
    <div class="container mx-auto flex flex-col md:flex-row
items-center">
      <div class="md:w-1/2">
        <h2 class="text-2xl font-bold mb-5">Acerca de
Mí</h2>
        <p class="max-w-md mb-5">
          Soy un desarrollador web con experiencia en el
          uso de diversas tecnologías, incluyendo HTML, CSS, JavaScript y
          frameworks como Tailwind CSS. Me apasiona crear interfaces
          atractivas y funcionales.
        </p>
      </div>
      <div class="md:w-1/2">
        
      </div>
    </div>
```



```
</section>

<!-- Sección Proyectos -->
<section class="py-10 bg-white">
  <div class="container mx-auto">
    <h2 class="text-2xl font-bold text-center mb-5">Mis
Proyectos</h2>
    <div class="grid grid-cols-1 sm:grid-cols-2 lg:grid-
cols-3 gap-6">
      <!-- Proyecto 1 -->
      <div class="bg-gray-200 p-5 rounded-lg shadow-md">
        
        <h3 class="text-xl font-bold mb-2">Proyecto
1</h3>
        <p>Descripción breve del proyecto.</p>
      </div>
      <!-- Proyecto 2 -->
      <div class="bg-gray-200 p-5 rounded-lg shadow-md">
        
        <h3 class="text-xl font-bold mb-2">Proyecto
2</h3>
        <p>Descripción breve del proyecto.</p>
      </div>
      <!-- Proyecto 3 -->
      <div class="bg-gray-200 p-5 rounded-lg shadow-md">
        
        <h3 class="text-xl font-bold mb-2">Proyecto
3</h3>
        <p>Descripción breve del proyecto.</p>
      </div>
    </div>
  </div>
```

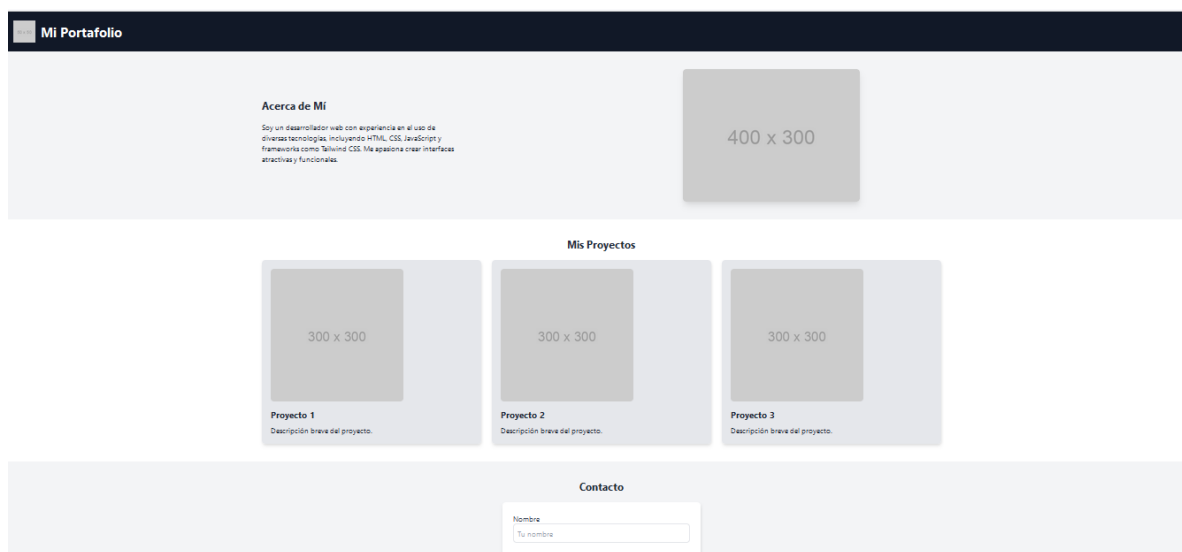


```
    </div>
</section>

<!-- Sección Contacto -->
<section class="py-10">
    <div class="container mx-auto">
        <h2 class="text-2xl font-bold text-center mb-5">Contacto</h2>
        <form class="max-w-md mx-auto bg-white p-6 rounded-lg shadow-md">
            <div class="mb-4">
                <label class="block text-gray-700"
for="name">Nombre</label>
                <input class="w-full border-2 p-2 rounded-lg"
type="text" id="name" placeholder="Tu nombre">
            </div>
            <div class="mb-4">
                <label class="block text-gray-700"
for="email">Correo Electrónico</label>
                <input class="w-full border-2 p-2 rounded-lg"
type="email" id="email" placeholder="Tu correo">
            </div>
            <div class="mb-4">
                <label class="block text-gray-700"
for="message">Mensaje</label>
                <textarea class="w-full border-2 p-2 rounded-
lg" id="message" rows="5" placeholder="Tu mensaje"></textarea>
            </div>
            <button class="bg-blue-500 text-white px-4 py-2
rounded-lg" type="submit">Enviar</button>
        </form>
    </div>
</section>
```



```
<!-- Footer -->
<footer class="bg-gray-900 text-white p-5 text-center">
  <p>&copy; 2024 Mi Portafolio. Todos los derechos
reservados.</p>
</footer>
</body>
</html>
```



VII. JavaScript

Es un lenguaje de programación de alto nivel y de propósito general. Ampliamente usado en el desarrollo web, permitiendo a los desarrolladores crear sitios interactivos y dinámicos.

1. Conceptos Básicos

1.1. Variables: Almacena datos temporalmente en la memoria RAM, se pueden declarar con var, let o const.

```
let nombre = "Juan"; // Variable que puede cambiar
const pi = 3.1416;   // Constante
```

1.2. Tipos de datos: number, string, boolean, null, undefined, object, y symbol.

```
let numero = 5;
```




```
let texto = "Hola";
```

```
let esVerdadero = true;
```

1.3. Operadores: Operadores básicos de aritmética, asignación, comparación y lógicos.

```
let suma = 5 + 3; // 8  
let esIgual = (5 == '5'); // true  
let esEstrictoIgual = (5 === '5'); // false
```

2. Funciones

2.1. Definición de funciones: Las funciones encapsulan bloques de código que pueden ser reutilizados.

```
function sumar(a, b) {  
  return a + b;  
}  
console.log(sumar(2, 3)); // 5
```

2.2. Funciones flecha: Forma más concisa de declarar funciones.

```
const multiplicar = (a, b) => a * b;
```

3. Estructuras de Control

3.1. Condicionales:

```
if (edad >= 18) {  
  console.log("Eres mayor de edad");  
} else {  
  console.log("Eres menor de edad");  
}
```

3.2. Bucles: for, while y do...while:



```
for (let i = 0; i < 5; i++) {  
    console.log(i); // Imprime 0 a 4 por consola  
}
```

```
//Ejemplo con while  
//Archivo index.html
```

```
<!DOCTYPE html>  
<html lang="en">  
<head>  
    <meta charset="UTF-8">  
    <meta name="viewport" content="width=device-width,  
initial-scale=1.0">  
    <title>while</title>  
</head>  
<body>  
    <script src="while.js"></script>  
</body>  
</html>
```

```
//Archivo while.js
```

```
Op=true;  
while (Op==true){  
    O=prompt("1--> Para continuar, 2-->salir");  
    if (O=='2'){  
        console.log("Fin del programa");  
        Op=false;  
    }  
    else{  
        console.log("El programa continua ejecutandose");  
    }  
}
```

4. Algoritmos



4.1. Algoritmo para encontrar el número mayor:

```
let numeros = [3, 6, 2, 8, 4];  
let mayor = Math.max(numeros);  
console.log(mayor); // 8
```

4.2. Algoritmo de búsqueda en un array:

```
let frutas = ["manzana", "naranja", "mango"];  
console.log(frutas.indexOf("mango")); // 2
```

4.3. Secuencia fibonacci: La secuencia de Fibonacci es una serie de números donde cada número es la suma de los dos anteriores.

```
function fibonacci(n) {  
  let secuencia = [0, 1];  
  for (let i = 2; i < n; i++) {  
    secuencia.push(secuencia[i - 1] + secuencia[i - 2]);  
  }  
  return secuencia;  
}  
console.log(fibonacci(10)); // [0, 1, 1, 2, 3, 5, 8, 13, 21,  
34]
```

4.4. Número perfecto: Número entero positivo que es igual a la suma de sus divisores enteros, excluyendo el mismo número.

```
function esNumeroPerfecto(numero) {  
  let suma = 0;  
  for (let i = 1; i < numero; i++) {  
    if (numero % i === 0) {  
      suma += i;  
    }  
  }  
  return suma === numero;  
}  
console.log(esNumeroPerfecto(6)); // true (6 es un número  
perfecto)  
console.log(esNumeroPerfecto(28)); // true (28 es un número  
perfecto)  
console.log(esNumeroPerfecto(10)); // false (10 no es un número  
perfecto)
```

4.5. Factorial: La factorial de un número es el producto de todos los números enteros desde 1 hasta ese número.



```
function factorial(n) {  
  if (n === 0 || n === 1) {  
    return 1;  
  }  
  return n * factorial(n - 1);  
}  
console.log(factorial(5)); // 120
```

4.6. Ordenamiento de array método de burbuja:

```
function ordenarBurbuja(arr) {  
  let n = arr.length;  
  for (let i = 0; i < n - 1; i++) {  
    for (let j = 0; j < n - i - 1; j++) {  
      if (arr[j] > arr[j + 1]) {  
        [arr[j], arr[j + 1]] = [arr[j + 1], arr[j]];  
      }  
    }  
  }  
  return arr;  
}  
  
let numeros = [5, 3, 8, 1, 2];  
console.log(ordenarBurbuja(numeros)); // [1, 2, 3, 5, 8]
```

4.7. Número Primo: Un número es primo si es mayor que 1 y no divisible por ningún otro número excepto por 1 y por sí mismo.

```
function esPrimo(numero) {  
  if (numero <= 1) return false;  
  for (let i = 2; i < numero; i++) {  
    if (numero % i === 0) {  
      return false;  
    }  
  }  
  return true;  
}  
console.log(esPrimo(7)); // true  
console.log(esPrimo(10)); // false
```

4.8. Máximo Común Divisor (MCD): El MCD de dos números es el número más grande que divide a ambos sin dejar residuo.



Algoritmo para calcular el MCD usando el método de Euclides:

```
function mcd(a, b) {  
  while (b !== 0) {  
    let temp = b;  
    b = a % b;  
    a = temp;  
  }  
  return a;  
}  
  
console.log(mcd(60, 48)); // 12
```

4.9. Ordenación por Selección: El algoritmo de ordenación por selección encuentra el valor más pequeño de una lista y lo coloca en la primera posición, repitiendo este proceso para los demás elementos.

```
function ordenarSeleccion(arr) {  
  for (let i = 0; i < arr.length; i++) {  
    let min = i;  
    for (let j = i + 1; j < arr.length; j++) {  
      if (arr[j] < arr[min]) {  
        min = j;  
      }  
    }  
    if (min !== i) {  
      [arr[i], arr[min]] = [arr[min], arr[i]];  
    }  
  }  
  return arr;  
}  
  
let numeros = [29, 10, 14, 37, 13];  
console.log(ordenarSeleccion(numeros)); // [10, 13, 14, 29, 37]
```

5. Manejo del DOM (Document Object Model)

El DOM es una representación estructurada de la página web que permite a JavaScript interactuar con los elementos HTML.

5.1. Seleccionar elementos:

```
const titulo = document.getElementById('titulo');
```



```
const parrafos = document.getElementsByTagName('p');  
const botones = document.querySelectorAll('.boton');
```

5.2. Modificar elementos:

```
const titulo = document.getElementById('titulo');  
titulo.textContent = "Nuevo título";  
titulo.style.color = "blue";
```

5.3. Crear y agregar elementos:

```
let nuevoParrafo = document.createElement('p');  
nuevoParrafo.textContent = "Este es un párrafo nuevo";  
document.body.appendChild(nuevoParrafo);
```

6. Eventos

Los eventos permiten a JavaScript responder a interacciones del usuario o del sistema.

```
const boton = document.querySelector('#miBoton');  
boton.addEventListener('click', () => {  
    alert('Botón clickeado');  
});
```

6.1. Eventos de Mouse: Estos eventos responden a la interacción del usuario con el ratón.

click: Se activa cuando el usuario hace clic en un elemento.

dblclick: Se activa cuando el usuario hace doble clic.

mousedown: Se dispara cuando se presiona el botón del mouse.

mouseup: Se dispara cuando se suelta el botón del mouse.

mousemove: Se activa cuando el usuario mueve el mouse.



mouseover: Se activa cuando el cursor pasa sobre un elemento.

mouseout: Se activa cuando el cursor sale de un elemento.

contextmenu: Se activa cuando se intenta abrir el menú contextual (botón derecho).

6.2. Eventos de Teclado: Estos eventos se activan cuando el usuario interactúa con el teclado.

keydown: Se activa cuando se presiona una tecla.

keypress: Se activa cuando se mantiene presionada una tecla (depreciado en versiones modernas).

keyup: Se activa cuando se suelta una tecla.

```
document.addEventListener('keydown', (evento) => {  
    console.log(`Tecla presionada: ${evento.key}`);  
    if (evento.key === 'Enter') {  
        alert('Presionaste Enter');  
    }  
});
```

6.3. Eventos de Formulario: Estos eventos ocurren cuando se interactúa con elementos de formulario.

submit: Se activa cuando un formulario es enviado.

focus: Se activa cuando un elemento recibe el foco.

blur: Se activa cuando un elemento pierde el foco.

change: Se activa cuando cambia el valor de un elemento de formulario (por ejemplo, <input> o <select>).

input: Se activa cada vez que el valor de un campo de entrada cambia.

6.4. Eventos de Ventana (Window): Estos eventos responden a cambios en la ventana o documento.



load: Se activa cuando la página y sus recursos (imágenes, estilos, etc.) han terminado de cargarse.

unload: Se activa antes de que la página se descargue (cuando el usuario cierra o recarga la página).

resize: Se activa cuando el usuario cambia el tamaño de la ventana.

scroll: Se activa cuando el usuario desplaza la página.

beforeunload: Permite mostrar un mensaje de advertencia antes de que el usuario abandone la página.

6.5. Eventos Multimedia: Estos eventos responden a la reproducción de contenido multimedia.

play: Se activa cuando la reproducción de un video/audio comienza.

pause: Se activa cuando la reproducción se pausa.

ended: Se activa cuando el video/audio ha terminado.

6.6. Otros Eventos:

error: Se activa cuando hay un error al cargar un recurso.

Ejemplo:

```
<!DOCTYPE html>
<html Lang="es">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width,
initial-scale=1.0">
  <title>Ejemplo de Eventos en JavaScript</title>
</head>
<body>

  <h1 id="titulo">Eventos en JavaScript</h1>
  <button id="botonClick">Haz clic aquí</button>
  <input type="text" id="inputTexto" placeholder="Escribe
algo aquí">
```




```
<form id="formulario">
  <label for="nombre">Nombre:</label>
  <input type="text" id="nombre" name="nombre">
  <button type="submit">Enviar</button>
</form>
<div id="areaMovimiento" style="border: 1px solid black;
padding: 10px; width: 200px; height: 200px; margin-top:
20px;">
  Área de Movimiento
</div>

<script>
  // Evento de Carga
  window.addEventListener('load', () => {
    console.log('La página ha cargado completamente');
  });

  // Evento de Clic
  document.getElementById('botonClick').addEventListener(
'click', () => {
    alert('Botón clickeado!');
  });

  // Evento de Doble Clic
  document.getElementById('botonClick').addEventListener(
'dblclick', () => {
    console.log('Botón doble clickeado!');
  });

  // Eventos de Teclado
  const inputTexto =
document.getElementById('inputTexto');
  inputTexto.addEventListener('keydown', (event) => {
    console.log(`Tecla presionada: ${event.key}`);
  });
```



```
});  
inputTexto.addEventListener('keyup', () => {  
    console.log(`Texto actual: ${inputTexto.value}`);  
});  
  
// Evento de Enviar Formulario  
document.getElementById('formulario').addEventListener(  
'submit', (event) => {  
    event.preventDefault(); // Evitar el envío real del  
formulario  
    alert('Formulario enviado!');  
});  
  
// Evento de Enfoque y Desenfoque  
inputTexto.addEventListener('focus', () => {  
    inputTexto.style.backgroundColor = 'lightblue';  
});  
inputTexto.addEventListener('blur', () => {  
    inputTexto.style.backgroundColor = '';  
});  
  
// Evento de Cambio en el Input  
inputTexto.addEventListener('change', () => {  
    console.log('El contenido del input ha cambiado');  
});  
  
// Eventos de Mouse en el Área de Movimiento  
const areaMovimiento =  
document.getElementById('areaMovimiento');  
areaMovimiento.addEventListener('mouseover', () => {  
    areaMovimiento.style.backgroundColor = 'lightgreen';  
});  
areaMovimiento.addEventListener('mouseout', () => {  
    areaMovimiento.style.backgroundColor = '';
```



```
});  
areaMovimiento.addEventListener('mousemove', (event) =>  
{  
    areaMovimiento.textContent = `Mouse en X:  
${event.offsetX}, Y: ${event.offsetY}`;  
});  
  
// Evento de Redimensionar La Ventana  
window.addEventListener('resize', () => {  
    console.log('La ventana ha cambiado de tamaño');  
});  
  
// Evento de Desplazamiento  
window.addEventListener('scroll', () => {  
    console.log('Desplazamiento en la página');  
});  
  
// Evento de Context Menu (Botón derecho del mouse)  
document.addEventListener('contextmenu', (event) => {  
    event.preventDefault(); // Evita el menú contextual  
    alert('Haz clic con el botón derecho!');  
});  
</script>  
</body>  
</html>
```

7. Consumo de APIs

JavaScript permite hacer peticiones HTTP para consumir datos de APIs externas utilizando `fetch()`.

```
fetch('https://api.example.com/data')  
    .then(response => response.json())  
    .then(data => console.log(data))  
    .catch(error => console.error('Error:', error));
```



7.1. Petición POST:

```
fetch('https://api.example.com/data', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({ nombre: 'Juan', edad: 25 }),
})
  .then(response => response.json())
  .then(data => console.log('Success:', data))
  .catch(error => console.error('Error:', error));
```

7.2. Consumo de la API Placeholder: La API de Placeholder es útil para generar datos ficticios, como posts o comentarios.

```
fetch('https://jsonplaceholder.typicode.com/posts/1')
  .then(response => response.json())
  .then(data => {
    console.log('Post:', data);
    // Aquí se realiza la programación de los datos del post
  })
  .catch(error => console.error('Error:', error));
```

7.3. Crear un nuevo post (método POST):

```
fetch('https://jsonplaceholder.typicode.com/posts', {
  method: 'POST',
  headers: {
    'Content-Type': 'application/json',
  },
  body: JSON.stringify({
    title: 'Nuevo post',
    body: 'Contenido del post.',
    userId: 1,
  }),
})
  .then(response => response.json())
  .then(data => {
    console.log('Post creado:', data);
  })
  .catch(error => console.error('Error:', error));
```



7.4. Consumo de la API del Clima (OpenWeather): La API de OpenWeather proporciona información meteorológica. Es Necesaria una clave de API que se obtiene registrándose en su sitio web.

```
const apiKey = 'CLAVE_API'; // Reemplaza con la clave de API
de OpenWeather

const ciudad = 'Madrid';
fetch('https://api.openweathermap.org/data/2.5/weather?q=${ciudad}
&appid=${apiKey}&units=metric&lang=es')
  .then(response => response.json())
  .then(data => {
    console.log('Clima en Madrid:', data);
    // Aquí puedes mostrar el clima en tu aplicación
    console.log(`Temperatura: ${data.main.temp}°C`);
    console.log(`Descripción:${data.weather[0].description}`);
  })
  .catch(error => console.error('Error al obtener el clima:',
error));
```

7.5. Consumo de la API de Pokémon

```
fetch('https://pokeapi.co/api/v2/pokemon/pikachu')
  .then(response => response.json())
  .then(data => {
    console.log('Datos de Pikachu:', data);
    console.log(`Nombre: ${data.name}`);
    console.log(`Altura: ${data.height}`);
    console.log(`Peso: ${data.weight}`);
    console.log(`Tipo:      ${data.types.map(tipo      =>
tipo.type.name).join(', ')}`);
  })
  .catch(error => console.error('Error al obtener los datos de
Pokémon:', error));
```

8. Canvas

El canvas permite dibujar gráficos y animaciones en el navegador.



```
<canvas id="miCanvas" width="500" height="500"></canvas>
```

8.1. Dibujo básico en canvas:

```
const canvas = document.getElementById('miCanvas');
const ctx = canvas.getContext('2d');
// Dibuja un rectángulo
ctx.fillStyle = 'red';
ctx.fillRect(50, 50, 100, 100);
// Dibuja un círculo
ctx.beginPath();
ctx.arc(150, 150, 50, 0, Math.PI * 2);
ctx.fillStyle = 'blue';
ctx.fill();
```

8.2. Animación en canvas:

```
function moverRectangulo() {
    ctx.clearRect(0, 0, canvas.width, canvas.height); // Limpiar
    canvas
    ctx.fillStyle = 'green';
    ctx.fillRect(x, 50, 100, 100);
    x += 1; // Mover el rectángulo
    requestAnimationFrame(moverRectangulo); // Repetir animación
}
let x = 0;
moverRectangulo();
```



VIII. Tutoriales YouTube recomendados

1. HTML Y CSS

1.1. Curso de HTML y CSS desde CERO (Completo)

<https://www.youtube.com/watch?v=ELSm-G201Ls>

1.2. ¡Aprende CSS ahora! curso completo GRATIS desde cero

<https://www.youtube.com/watch?v=wZniZEbPAzk>

1.3. HTML y CSS Curso Completo Español

https://www.youtube.com/playlist?list=PLPI81lqbj-4LKo66cEts5yC_AjOvqKptm

2. GIT y Git Hub

2.1. ¡Aprende GIT ahora! curso completo GRATIS desde cero

<https://www.youtube.com/watch?v=VdGzPZ31ts8>

2.2. ¿Qué es Git y cómo funciona?

<https://www.youtube.com/watch?v=jGehuhFhtnE>

2.3. Curso Git & GitHub

https://www.youtube.com/watch?v=ANF1X42_ae4&list=PLU8oAIHdN5BIyaPFiNQcV0xDqy0eR35aU

3. Javascript

3.1. Curso de JavaScript desde 0

<https://www.youtube.com/watch?v=m2nscBtQEIs&list=PLU8oAIHdN5BmpobVmjl1lneKIVLJ84TID>

3.2. ¡Aprende JavaScript Ahora! curso completo desde cero para principiantes

<https://www.youtube.com/watch?v=QoC4RxNls5M>

4. Frameworks de css

4.1. Materialize



<https://www.youtube.com/watch?v=F7D1ydKnA7E&list=PLYAyQauAPx8maaj2i1br7-il4AQLZkcqA>

https://www.youtube.com/watch?v=YJKWkUi2r5g&list=PLPI81lqbj-4J2Lbx1_qp7Yzo7wvjYiQ4E

4.2. Bootstrap

https://www.youtube.com/watch?v=p8Z_qSbN9zQ&list=PLZ2ovOgdl-kUmz8-r2DNigQ_qwpk1s5q4

<https://www.youtube.com/watch?v=JDs45GmZWUo>

<https://www.youtube.com/watch?v=F55FC7UheLA>

4.3. Tailwind css

<https://www.youtube.com/watch?v=3xIUAMXui2c&list=PLPI81lqbj-4JdoHDIERR2ptkw9zRggXAL>

5. Frameworks de javascript

5.1. Curso de Svelte

https://www.youtube.com/watch?v=pze2JJj82XA&list=PLTd5ehlJ0goM-5mQxXLmCr5nHZX_yc2QT

5.2. Curso gratuito de Svelte

https://www.youtube.com/watch?v=Xsxm8_BI63s&list=PLV8x_i1fqBw2QScggh0pw2ATSJg_WHqUN

5.3. Curso de Vue.js 3 en Español 2024

https://www.youtube.com/playlist?list=PLg-z1C9R1jutezByZ3WmH8NE_JqYBsDOA

5.4. Curso Gratuito de VUE desde Cero

<https://www.youtube.com/playlist?list=PLUdlARNXMVkk7VotfS3YVxcghSdJR7E2A>

5.5. Primeros pasos con Angular 18 [2024]

<https://www.youtube.com/watch?v=aiaKyhiY9TQ>

5.6. Angular clase definitiva

https://www.youtube.com/watch?v=xNfqXTENmY&list=PL42UNLc8e48RINrNGumxAKulG5CWgs_yv&index=2



5.7. Curso de React 2024

https://www.youtube.com/watch?v=7iobxzd_2wY&list=PLUofhDIg_38q4D0xNWp7FEHOTcZhjWJ29

5.8. ¡Aprende React ahora! curso completo para crear aplicaciones

https://www.youtube.com/watch?v=yI_r_1CasXkM

5.9. Curso de Alpine JS

https://www.youtube.com/watch?v=_0xJ4OxELw&list=PLs4YDKCLLrp-8DyeKNOZ4DJVQu-ue74JB

6. Desarrollo Móvil

6.1. Curso de ANDROID Studio con KOTLIN

<https://www.youtube.com/watch?v=DX-CIdg3jWY>

6.2. Curso Android con Kotlin

https://www.youtube.com/watch?v=k9NndvHyUvw&list=PLU8oAIHdN5BkdfBPpNv_IVCJxJgE87cr0

6.3. Apache Cordova

<https://www.youtube.com/watch?v=Y-kTehbMzT4&list=PLRM7PpbqqStLdavuvdZusWc17FXsLIY8k>

6.4. Flutter

<https://www.youtube.com/watch?v=hZ08stCV8n4&list=PLutr4gv1YI96buJoWiyG98qnBG26M9U9>

6.5. Flet

https://www.youtube.com/playlist?list=PL2PZw96yQChyD79_mtgY_oBdyiCdRJnJa

https://www.youtube.com/playlist?list=PLfSVB4Wge3DLFBx_RMCCCKeGW_hOuDeB3-



Ejercicios propuestos

1. Desarrollar un portafolio de hoja de vida utilizando herramientas HTML y css nativas.
2. Diseñar una página web representativa del centro de formación, incluyendo imágenes que reflejen su esencia y actividades principales. Seleccionar un tema visual adecuado que complemente y resalte los elementos distintivos del centro, creando una experiencia atractiva y coherente para los visitantes.

Referencias

Laboratoria. (n.d.). Front end vs. back end: ¿cuál es la diferencia? Laboratoria.
<https://hub.laboratoria.la/front-end-vs-back-end-cual-es-la-diferencia>

Tutoriales programación ya. (s. f.). <https://www.tutorialesprogramacionya.com/>

Control del documento

| | Nombre | Cargo | Dependencia | Fecha |
|-------------------|----------------------------|------------|----------------------------------|-----------------|
| Autor (es) | Jose David Montesino Hoyos | Instructor | Centro Biotecnológico del caribe | Octubre de 2024 |

Control de cambios

| | Nombre | Cargo | Dependencia | Fecha | Razón del Cambio |
|-------------------|--------|-------|-------------|-------|------------------|
| Autor (es) | | | | | |