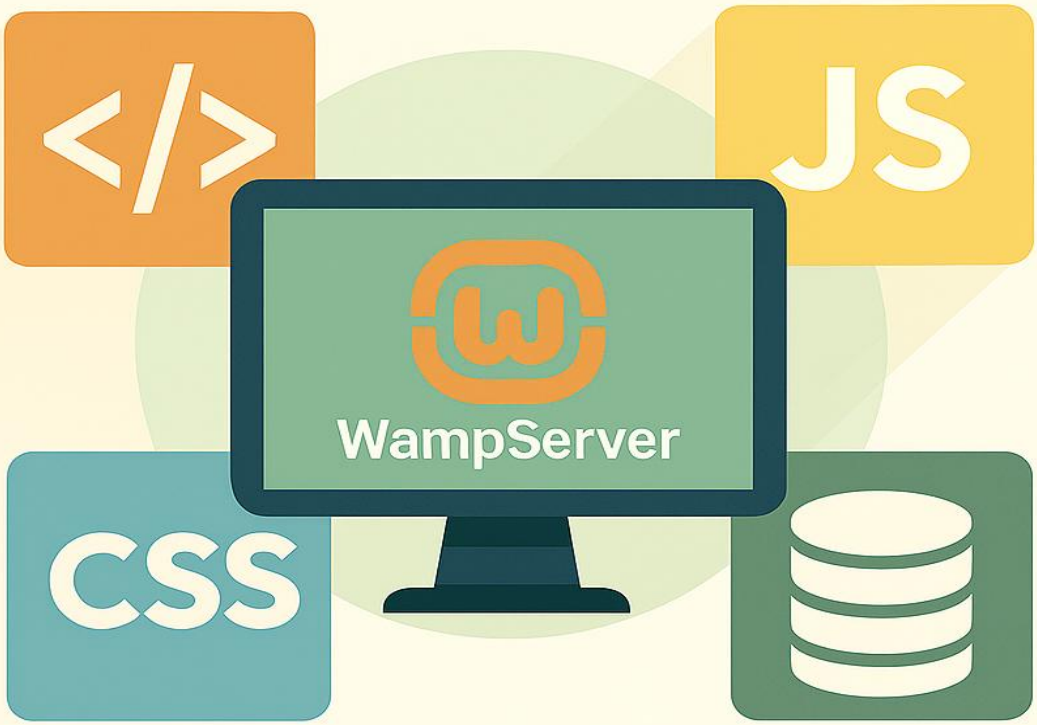
Programación de Páginas Web

Sexto semestre



Alfonso Torres Torres

📘 Introducción

Programación de Páginas Web

En la actualidad, gran parte de la información, los servicios y la comunicación se realizan a través de internet. Las páginas web ya no son simples documentos informativos, sino sistemas dinámicos capaces de interactuar con los usuarios, almacenar datos y ofrecer soluciones digitales a problemas reales. Por ello, el desarrollo web se ha convertido en una de las áreas más importantes y demandadas dentro del campo de la programación.

El presente taller tiene como propósito que el estudiante **diseñe, desarrolle y publique en entornos locales páginas web dinámicas con conectividad a base de datos mediante HTML, CSS3 y JavaScript**. A lo largo del curso, se fortalecerán habilidades técnicas y creativas que permitirán construir sitios web funcionales, estructurados y visualmente atractivos.

Durante el semestre, el alumno aprenderá a:

- Utilizar **HTML** para estructurar correctamente el contenido de una página web.
- Aplicar **CSS3** para diseñar interfaces modernas, responsivas y agradables al usuario.
- Implementar **JavaScript** para generar interactividad y dinamismo en los sitios web.
- Integrar bases de datos para almacenar y gestionar información de manera eficiente.
- Trabajar en entornos locales de desarrollo para probar y optimizar proyectos antes de su publicación.

Este material te guía para la programación de un pequeño proyecto, el cuál se te sugiere que desarrolles por completo.

Se sugiere que el estudiante lea y subraye lo más interesante.

Este material abarca todos los contenidos del programa de estudio

🧩 ¿Qué es WampServer?

WampServer es un programa que convierte tu computadora en un **servidor web local**.

👉 Esto significa que puedes crear y probar páginas web dinámicas (con base de datos) **sin necesidad de internet**.

La palabra **WAMP** significa:

- **W** → Windows
- **A** → Apache
- **M** → MySQL
- **P** → PHP

¿Qué integra WampServer?

- Cuando instalas WampServer, se instalan automáticamente tres componentes principales:

1️⃣ Apache HTTP Server

Es el **servidor web**.

◇ Su función es:

- Recibir las solicitudes del navegador.
- Mostrar las páginas web.

Ejemplo:
Cuando escribes http://localhost, Apache muestra tu página.

2️⃣ MySQL

Es el **gestor de base de datos**.

◇ Sirve para:

- Guardar información.
- Organizar datos en tablas.
- Consultar datos.

Ejemplo:
Guardar alumnos, productos, usuarios, etc.

3️⃣ PHP

Es el lenguaje de programación del lado del servidor.

- Sirve para:
  - Conectar páginas con bases de datos.
  - Procesar formularios.
  - Crear páginas dinámicas.

Ejemplo:
Tu archivo guardar.php usa PHP para guardar datos en MySQL.

📘 ¿Qué se puede hacer con WampServer?

Con WampServer puedes:

- ☑ Crear páginas web dinámicas
- ☑ Conectar formularios a bases de datos
- ☑ Crear sistemas escolares
- ☑ Hacer tiendas en línea
- ☑ Crear sistemas de usuarios (login y registro)
- ☑ Practicar desarrollo web sin pagar hosting

Es ideal para estudiantes porque todo funciona en tu propia computadora.

📂 Estructura jerárquica de carpetas en WampServer

Cuando instalas WampServer, se crea una carpeta principal, normalmente en:

```
C:\wamp64\www\
```

📁 Carpeta www

- Es la carpeta más importante.

Aquí debes guardar tus proyectos.

Ejemplo:

```
www
│
├── proyecto1
│     ├── index.html
│     ├── estilos.css
│     └── guardar.php
│
├── escuela
│     ├── index.html
│     └── conexion.php
```

- ◇

Si guardas tu proyecto en:

```
http://localhost/escuela
```

🌐 ¿Qué es localhost?

localhost significa:

👉 “Esta misma computadora”.

Cuando escribes:

```
http://localhost
```

Estás diciendo:

"Muéstrame el servidor que está en mi propia PC".

🔁 ¿Cómo funciona todo junto?

1. El usuario llena un formulario.
2. El formulario envía datos a un archivo .php.
3. PHP procesa los datos.
4. PHP los guarda en MySQL.
5. Apache muestra el resultado en el navegador.

Todo eso ocurre dentro de tu computadora gracias a WampServer.

🎯 Ventajas de WampServer

✓ Es gratis
✓ Fácil de instalar
✓ Ideal para aprender
✓ No necesitas internet
✓ Puedes trabajar sin hosting

🟠 Ejercicio 1 Instala WampServer en tu computadora solo sigue las indicaciones del siguiente video.

https://www.youtube.com/watch?v=k4lnf-7-sbE&t=10s

Una vez que has instaldo y probado el funcionamiento de tu WampServer deberás

1. Crea una carpeta llamada **proyecto** dentro de www.

Esta carpeta es importante, pues la vas a utilizar para desarrollar todo el proyecto

¿QUÉ ES HTML?

HTML (HyperText Markup Language) es el lenguaje de marcado que usamos para crear páginas web. No es un lenguaje de programación, sino un lenguaje que usa etiquetas para estructurar el contenido.

CONCEPTOS BÁSICOS

- **Etiquetas**: Palabras clave entre símbolos < >
- **Elemento**: Etiqueta + contenido + etiqueta de cierre
- **Atributos**: Información adicional dentro de las etiquetas

ESTRUCTURA BÁSICA DE UN DOCUMENTO HTML

```
<!DOCTYPE html>
<html>
<head>
    <!-- Información para el navegador -->
    <title>Mi Página</title>
</head>
<body>
    <!-- Contenido visible -->
    <h1>¡Hola Mundo!</h1>
</body>
</html>
```

ETIQUETAS FUNDAMENTALES

1. <!DOCTYPE html>

¿Qué hace?: Le dice al navegador que estamos usando HTML5

```
<!DOCTYPE html>
```

IMPORTANTE: Siempre va en la primera línea del documento.

2. <html>

¿Qué hace?: Es la raíz del documento, contiene todo el contenido HTML

```
<html lang="es">
  <!-- Todo el contenido aquí -->
</html>
```

Atributo común: lang="es" indica que el idioma es español.

3. <head>

¿Qué hace?: Contiene información sobre la página (metadatos)

```
<head>
    <title>Mi Página</title>
    <meta charset="UTF-8">
    <meta name="description" content="Mi primera página web">
</head>
```

Contiene: Título, caracteres especiales, descripción, enlaces a CSS.

4. <title>

¿Qué hace?: Define el título que aparece en la pestaña del navegador

```
<title>Mi Primera Página Web</title>
```

5. <body>

¿Qué hace?: Contiene todo el contenido visible de la página

```
<body>
    <h1>Título Principal</h1>
    <p>Este es un párrafo de texto.</p>
</body>
```

6. <h1> a <h6>

¿Qué hacen?: Encabezados o títulos (del más importante al menos importante)

```
<h1>Título Principal</h1>    <!-- Más importante -->
<h2>Subtítulo</h2>
<h3>Sección</h3>
<h4>Subsección</h4>
<h5>Apartado</h5>
<h6>Detalle</h6>             <!-- Menos importante -->
```

REGLA DE ORO: Solo debe haber **UN** <h1> por página.

7. <p>

¿Qué hace?: Define un párrafo de texto

```
<p>Este es mi primer párrafo en HTML.</p>
<p>Este es mi segundo párrafo.</p>
```

8. <a>

¿Qué hace?: Crea un enlace (hipervínculo)

Atributo principal: href indica la dirección del enlace.

```
<!-- Enlace a otra página -->
<a href="https://www.google.com">Ir a Google</a>

<!-- Enlace dentro de la misma página -->
<a href="#seccion2">Ir a Sección 2</a>
```

9. <img> 🖼️

```
<img src="imagen.jpg" alt="Descripción de la imagen" width="300">
```

¿Qué hace?: Inserta una imagen

Atributos importantes:

- src: Ruta de la imagen
- alt: Texto alternativo (para accesibilidad)
- width: Ancho en píxeles

10. 📦

¿Qué hace?: Crea un contenedor o división (NO tiene significado semántico)

USO COMÚN: Agrupar elementos para darles estilo con CSS.

```
<div class="contenedor">
    <h2>Contenido aquí</h2>
    <p>Texto dentro del div.</p>
</div>
```

11. y <li> 📝

¿Qué hacen?: Crean listas no ordenadas (con viñetas)

```
<ul>
    <li>Primer elemento</li>
    <li>Segundo elemento</li>
    <li>Tercer elemento</li>
</ul>
```

12. y <li> 1️⃣ 2️⃣ 3️⃣

¿Qué hacen?: Crean listas ordenadas (con números)

```
<ol>
    <li>Paso uno</li>
    <li>Paso dos</li>
    <li>Paso tres</li>
</ol>
```

13. <table>, <tr>, <td> 📊

¿Qué hacen?: Crean tablas

```
<table border="1">
    <tr> <!-- Fila 1 -->
        <td>Celda 1</td>
        <td>Celda 2</td>
    </tr>
    <tr> <!-- Fila 2 -->
        <td>Celda 3</td>
        <td>Celda 4</td>
    </tr>
</table>
```

14. <form> y <input>

¿Qué hacen?: Crean formularios

```
<form action="/procesar.php" method="POST">
    <input type="text" name="nombre" placeholder="Tu nombre">
    <input type="submit" value="Enviar">
</form>
```

15.

¿Qué hace?: Contenedor en línea (para texto o elementos pequeños)

```
<p>Texto <span style="color:red;">resaltado</span> en rojo.</p>
```

DIFERENCIA ENTRE ELEMENTOS DE BLOQUE Y LÍNEA

Elementos de Bloque	Elementos de Línea
Ocupan todo el ancho disponible	Ocupan solo el espacio necesario
Empiezan en una nueva línea	Están en la misma línea
Ejemplos: `<div>`, `<h1>`, `<p>`	Ejemplos: `<span>`, `<a>`, `<img>`

BUENAS PRÁCTICAS

1. Siempre cerrar las etiquetas

```
<!-- BIEN -->
<p>Texto</p>

<!-- MAL -->
<p>Texto
```

```
<!-- BIEN -->
<div>

<!-- MAL -->
<DIV>
```

2. **Usar minúsculas**

```
<!-- BIEN -->
<div><p>Texto</p></div>

<!-- MAL -->
<div><p>Texto</div></p>
```

3. **Anidar correctamente**

4. **Usar atributo** alt **en imágenes** ☑

```html
<!-- BIEN -->
<img src="foto.jpg" alt="Estudiante en clase">

<!-- MAL -->
<img src="foto.jpg">
```

🎯 RESUMEN VISUAL

```
<!DOCTYPE html>  → Define la versión de HTML
<html>           → Contenedor principal
  <head>         → Información para el navegador
    <title>      → Título en la pestaña
  </head>
  <body>         → Contenido visible
    <h1>         → Título principal
    <p>          → Párrafo de texto
    <img>        → Imagen
    <a>          → Enlace
  </body>
</html>
```

📚 APUNTES DE HTML SEMÁNTICO

🎯 ¿QUÉ ES HTML SEMÁNTICO?

HTML Semántico significa usar etiquetas HTML que **describen su contenido**. No solo organizan la página, sino que **dicen qué tipo de contenido contienen**.

📌 ¿POR QUÉ ES IMPORTANTE?

1. **Accesibilidad**: Los lectores de pantalla entienden mejor la estructura
2. **SEO**: Los buscadores indexan mejor tu contenido
3. **Mantenimiento**: Otros desarrolladores entienden tu código más fácilmente
4. **Organización**: La estructura es más clara y lógica

📋 ETIQUETAS SEMÁNTICAS PRINCIPALES

1. <header> 🏛️

¿Qué hace?: Representa la cabecera de una página o sección

```
<header>
    <h1>Mi Sitio Web</h1>
    <p>Bienvenidos a mi página personal</p>
</header>
```

CARACTERÍSTICAS:

- Puede contener logo, título principal, menú de navegación
- Se puede usar varias veces (en la página y en cada <article>)
- No es lo mismo que <head>

2. <nav> 🧭

¿Qué hace?: Define un bloque de enlaces de navegación

```
<nav>
    <ul>
        <li><a href="#inicio">Inicio</a></li>
        <li><a href="#servicios">Servicios</a></li>
        <li><a href="#contacto">Contacto</a></li>
    </ul>
</nav>
```

CARACTERÍSTICAS:

- Solo para navegación principal (no todos los grupos de enlaces)
- Generalmente va dentro de <header>

- Puede haber más de uno (navegación principal y secundaria)

3. <main> 🏛️

- **¿Qué hace?**: Contiene el contenido principal de la página

```
<main>
    <h1>Título del contenido principal</h1>
    <p>Aquí va el contenido más importante...</p>
</main>
```

CARACTERÍSTICAS:

- **SOLO UNO** por página
- Contenido único (no se repite en otras páginas)
- No debe contener elementos que se repiten (como menús)

4. <section> 📄

¿Qué hace?: Agrupa contenido temáticamente relacionado

```
<section>
    <h2>Sobre Mí</h2>
    <p>Texto sobre mi vida y experiencia...</p>
</section>

<section>
    <h2>Mis Proyectos</h2>
    <p>Descripción de mis trabajos...</p>
</section>
```

CARACTERÍSTICAS:

- Cada sección debe tener su propio encabezado
- Agrupa contenido que tiene un tema común
- Se puede anidar dentro de otras secciones

5. <article> 📰

¿Qué hace?: Contiene contenido independiente y autocontenido

CARACTERÍSTICAS:

- Tiene sentido por sí solo
- Ejemplos: post de blog, noticia, comentario, producto
- Se puede distribuir independientemente

```
<article>
    <h2>Cómo aprender HTML</h2>
    <p>Guía completa para principiantes...</p>
</article>

<article>
    <h2>Los beneficios del ejercicio</h2>
    <p>Por qué es importante mantenerse activo...</p>
</article>
```

6. <aside> 📌

- **¿Qué hace?**: Contiene contenido relacionado indirectamente

CARACTERÍSTICAS:

- Contenido secundario o complementario

```
<aside>
    <h3>Noticias relacionadas</h3>
    <ul>
        <li><a href="#">Artículo 1</a></li>
        <li><a href="#">Artículo 2</a></li>
    </ul>
</aside>
```

- Ejemplos: biografía del autor, enlaces relacionados, publicidad
- Puede ir dentro de un <article>

7. <footer> 👣

¿Qué hace?: Representa el pie de página de una página o sección

CARACTERÍSTICAS:

- Generalmente contiene: copyright, información de contacto, enlaces legales
- Se puede usar varias veces (pie de página principal y de artículos)

```
<footer>
    <p>© 2024 Mi Sitio Web</p>
    <p>Contacto: info@misitio.com</p>
    <nav>
        <a href="#">Política de privacidad</a>
        <a href="#">Términos de uso</a>
    </nav>
</footer>
```

🔍 DIFERENCIAS CLAVE

<section> vs <article>

`<section>`	`<article>`
Agrupa contenido temático	Contenido independiente
Puede no tener sentido solo	Tiene sentido por sí solo
Ejemplo: capítulo de libro	Ejemplo: post de blog
Más general	Más específico

<header> vs <head>

`<header>`	`<head>`
Visible en la página	**No visible** en la página
Va dentro de `<body>`	Va dentro de `<html>`
Contenido: título, logo, navegación	Contenido: metadatos, enlaces CSS
Puede haber varios	Solo uno por página

🎯 CUÁNDO USAR CADA ETIQUETA

¿<article> o <section>?

```html
<!-- USAR <article> -->
<article>
    <h2>Reseña: El último smartphone</h2>
    <p>Análisis completo del nuevo modelo...</p>
</article>

<!-- USAR <section> -->
<section>
    <h2>Características técnicas</h2>
    <p>Especificaciones del producto...</p>
</section>
```

¿<aside> dentro o fuera?

```html
<!-- Aside dentro de article (relacionado directamente) -->
<article>
    <h2>Título del artículo</h2>
    <p>Contenido principal...</p>
    <aside>
        <h3>Glosario de términos</h3>
        <!-- Definiciones relacionadas -->
    </aside>
</article>

<!-- Aside fuera (relacionado indirectamente) -->
<main>
    <article>...</article>
</main>
<aside>
    <h3>Artículos relacionados</h3>
    <!-- Enlaces a otros artículos -->
</aside>
```

BUENAS PRÁCTICAS DE HTML SEMÁNTICO

1. **Jerarquía correcta** ☑

```
<!-- BIEN -->
<main>
    <article>
        <section>...</section>
    </article>
</main>
```

```
<!-- BIEN -->
<section>
    <h2>Título de sección</h2>
    <!-- contenido -->
</section>
```

2. **Encabezados apropiados** ☑

3. **Un solo** <main> ☑

```
<!-- BIEN -->
<main>Contenido principal único</main>

<!-- MAL -->
<main>Primer contenido</main>
<main>Segundo contenido</main>
```

4. **No abusar de** ☑

```
<!-- BIEN (semántico) -->
<section>
    <article>...</article>
</section>

<!-- MAL (no semántico) -->
<div class="section">
    <div class="article">...</div>
</div>
```

💡 CONSEJOS PARA RECORDAR

Regla del "Sentido Solo"

🤔 **Pregúntate:** ¿Este contenido tiene sentido si lo saco de la página?

- **SÍ** ➔ <article>
- **NO** ➔ <section>

Regla del "Uno Principal"

🎯 **Recuerda:** Solo un <main> por página, pero puedes tener muchos <header>, <footer>, <nav>.

Regla del "Parentesco"

👨‍👩‍👧‍👦 **Visualiza:**

- <article> es como un "hijo" independiente
- <section> es como un "hermano" temático
- <aside> es como un "primo" relacionado

🌟 RESUMEN FINAL

HTML Semántico = HTML con Significado

Etiqueta	Significado	Ejemplo de uso
`<header>`	"Esta es la cabecera"	Logo, título, menú principal
`<nav>`	"Aquí hay navegación"	Menús principales o secundarios
`<main>`	"Esto es lo importante"	Contenido central de la página
`<article>`	"Esto es independiente"	Post de blog, noticia, comentario
`<section>`	"Esto va junto temáticamente"	Capítulos, grupos de contenido
`<aside>`	"Esto está relacionado"	Biografía, enlaces relacionados
`<footer>`	"Esto es el final"	Copyright, contacto, enlaces legales

Aquí inicia el proyecto: Crea una página web aplicando etiquetas semánticas, debe tener al menos 3 espacios (divisiones) para colocar diferente información en cada una.

Usa un editor de texto, en tu computadora puedes abrir el programa bloc de notas 

y escribe el siguiente código, una vez que lo has terminado, realiza los cambios que se te piden a continuación.

Archivo Editar Ver

```
<!DOCTYPE html>
<html>
<head>
    <title> Clase 2 </title>
    <meta charset="UTF-8">
    <link rel="stylesheet" href="estilos.css">
</head>
<body>

<div class="global">

 <header class="encabezado">
      <!--Aquí va una imágen o un texto-->
 </header>

 <nav class="navegacion">
      <!--Aquí van los enlaces o botones de navegación-->
 </nav>

 <main class="contenido">
      <!--Aquí se mostrará el contenido de cada boton o enlace-->
 </main>

</div>

</body>
</html>
```

1.- En el apartado de encabezado escribe: cambia <!—Aquí va una imagen o texto -- >

Por **Proyecto de programación de páginas Web**.

2.- En el apartado de navegación, cambia <!—Aquí van los enlaces o botones de navegación -- > por:

Corte 1
Corte 2
Corte 3
(puedes cambiarlos)

3.- En el apartado de contenido, pon una imagen con una frase motivadora

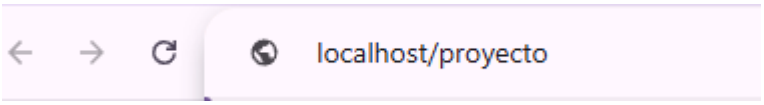
Para ello cambia <!—Aquí se mostrará el contenido de cada botón o enlace -- > por

<img src=”https://i.pinimg.com/236x/f7/88/28/f788283723578f564f6a4f42554e6fdb.jpg” alt=”Motivate” width=”300”>

4.- guárdalo con el nombre de index.html en la carpeta de trabajo, dentro de la carpeta www, en ella deberás todos los archivos del proyecto.

El resultado será algo parecido a…

Para ejecutar, abre tu navegador y en la barra de dirección escribe: localhost/proyecto



deberás tener activado tu wampserver

Tu proyecto tendrá un cambio radical… con solo unas ligeras modificaciones, después de trabajar la siguiente teoría.

📱 ¿QUÉ ES EL DISEÑO RESPONSIVO?

Diseño Responsivo es una técnica que hace que las páginas web se **vean bien en todos los dispositivos**: computadoras, tablets y teléfonos móviles.

📌 ¿POR QUÉ ES IMPORTANTE?

1. **Múltiples dispositivos**: La gente navega desde celulares, tablets y computadoras
2. **Mejor experiencia**: El contenido se adapta automáticamente
3. **SEO**: Google premia las páginas responsivas
4. **Mantenimiento**: Un solo sitio funciona para todos los dispositivos

🎯 PRINCIPIOS DEL DISEÑO RESPONSIVO

1. FLUIDO 💧

Los elementos se adaptan al ancho de la pantalla (usan %, no píxeles fijos)

2. FLEXIBLE 🧘

Las imágenes y medios se redimensionan automáticamente

3. MEDIA QUERIES 📏

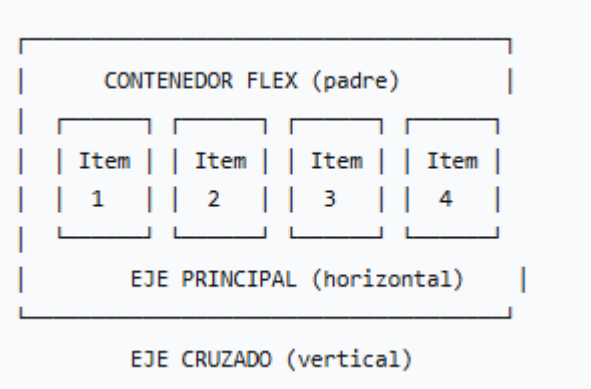
Cambian los estilos según el tamaño de la pantalla

📦 FLEXBOX: EL SUPERHÉROE DEL DISEÑO

¿QUÉ ES FLEXBOX?

Flexbox es un sistema de CSS que permite **distribuir elementos de forma flexible** dentro de un contenedor.

🔑 CONCEPTO CLAVE



🏠 PROPIEDADES DEL CONTENEDOR PADRE

1. display: flex; 🎮

Activa el modo Flexbox en un contenedor

2. flex-direction

Define la dirección de los elementos

VISUALMENTE:

```
row:        [A][B][C]
column:     [A]
            [B]
            [C]
```

```css
.contenedor {
    flex-direction: row;        /* Horizontal (default) → */
    flex-direction: row-reverse; /* Horizontal inversa ← */
    flex-direction: column;     /* Vertical ↓ */
    flex-direction: column-reverse; /* Vertical inversa ↑ */
}
```

3. justify-content ↔

Alineación horizontal (eje principal)

VISUALMENTE:

```
flex-start:    [A][B][C]_____
flex-end:      _____[A][B][C]
center:        ___[A][B][C]___
space-between: [A]___[B]___[C]
space-around:  _[A]__[B]__[C]_
```

```css
.contenedor {
    justify-content: flex-start;   /* A la izquierda (default) */
    justify-content: flex-end;     /* A la derecha */
    justify-content: center;       /* Centrado */
    justify-content: space-between;/* Espacio entre elementos */
    justify-content: space-around; /* Espacio alrededor */
    justify-content: space-evenly; /* Espacio igual */
}
```

4. align-items ↕

Alineación vertical (eje cruzado)

```css
.contenedor {
    align-items: stretch;    /* Estirar (default) */
    align-items: flex-start; /* Arriba */
    align-items: flex-end;   /* Abajo */
    align-items: center;     /* Centro */
    align-items: baseline;   /* Línea base del texto */
}
```

5. flex-wrap

Controla si los elementos saltan a la siguiente línea

VISUALMENTE:

```
nowrap:    [A][B][C][D][E][F]... (se encogen)
wrap:      [A][B][C]
           [D][E][F]
```

```css
.contenedor {
    flex-wrap: nowrap;  /* No saltan (default) */
    flex-wrap: wrap;    /* Saltan si no caben */
    flex-wrap: wrap-reverse; /* Saltan al revés */
}
```

6. align-content

Alineación de múltiples líneas (cuando hay wrap)

```
.contenedor {
    align-content: flex-start;  /* Arriba */
    align-content: flex-end;    /* Abajo */
    align-content: center;      /* Centro */
    align-content: space-between; /* Espacio entre líneas */
    align-content: space-around;  /* Espacio alrededor */
    align-content: stretch;     /* Estirar */
}
```

PROPIEDADES DE LOS HIJOS

1. flex-grow

Capacidad de crecer (0 = no crece, 1+ = sí crece)

```
.hijo {
    flex-grow: 1;  /* Puede crecer */
    flex-grow: 0;  /* No crece (default) */
}
```

2. flex-shrink

Capacidad de encogerse (1 = sí, 0 = no)

```
.hijo {
    flex-shrink: 1;  /* Puede encogerse (default) */
    flex-shrink: 0;  /* No se encoge */
}
```

3. flex-basis

Tamaño base antes de crecer o encogerse

```
.hijo {
    flex-basis: 200px;  /* Tamaño base de 200px */
    flex-basis: 50%;    /* Tamaño base del 50% */
    flex-basis: auto;   /* Automático (default) */
}
```

4. flex

Propiedad shorthand (grow + shrink + basis)

```
.hijo {
    flex: 1;           /* flex: 1 1 0% */
    flex: 1 0 200px;   /* grow:1, shrink:0, basis:200px */
    flex: 0 0 auto;    /* Tamaño fijo (no crece ni encoge) */
}
```

EJEMPLO DE CASO PRÁCTICO:

```css
/* Tres columnas iguales */
.columna {
    flex: 1;  /* Todas ocupan el mismo espacio */
}

/* Columna más grande */
.columna-principal {
    flex: 2;  /* Ocupa el doble que las demás */
}
.columna-lateral {
    flex: 1;  /* Ocupa la mitad */
}
```

5. align-self 🎮

Sobrescribe la alineación individual

```css
.hijo-especial {
    align-self: center;    /* Este se centra verticalmente */
    align-self: flex-end;  /* Este va abajo */
    align-self: stretch;   /* Este se estira */
}
```

6. order 🔢

Cambia el orden de los elementos

```css
.hijo {
    order: 0;  /* Valor por defecto */
    order: 1;  /* Va después */
    order: -1; /* Va antes */
}
```

VISUALMENTE:

```html
<div class="contenedor">
    <div style="order: 2;">Primero en HTML (aparece 2°)</div>
    <div style="order: 1;">Segundo en HTML (aparece 1°)</div>
    <div style="order: 3;">Tercero en HTML (aparece 3°)</div>
</div>
```

📱 MEDIA QUERIES

¿QUÉ SON LOS MEDIA QUERIES?

Media Queries son reglas CSS que se activan **según el tamaño de la pantalla**.

```
@media (condición) {
    /* Reglas CSS que se aplican cuando se cumple la condición */
    selector {
        propiedad: valor;
    }
}
```

🧩 SINTAXIS BÁSICA

📏 TAMAÑOS DE PANTALLA COMUNES

```
MÓVIL       TABLET      ESCRITORIO   GRANDE
└─────┘     └─────┘     └─────┘      └─────┘
0-600px     601-900px   901-1200px   1201px+
```

🎯 EJEMPLOS DE MEDIA QUERIES

1. Móviles 📱

```
/* Pantallas MENORES a 600px */
@media (max-width: 600px) {
    body {
        background-color: lightblue;
    }
    .menu {
        flex-direction: column;
    }
}
```

2. Tablets 📲

```
/* Pantallas ENTRE 601px y 900px */
@media (min-width: 601px) and (max-width: 900px) {
    .contenedor {
        grid-template-columns: repeat(2, 1fr);
    }
}
```

3. Escritorio 💻

```
/* Pantallas MAYORES a 901px */
@media (min-width: 901px) {
    .sidebar {
        display: block;
    }
}
```

🧠 ESTRATEGIAS DE DISEÑO RESPONSIVO

📌 MOBILE FIRST (Recomendado)

Diseñar primero para móviles y luego adaptar a pantallas grandes

```css
/* Estilos base para móvil (0px en adelante) */
.contenedor {
    flex-direction: column;
}

/* Tablet */
@media (min-width: 601px) {
    .contenedor {
        flex-direction: row;
        flex-wrap: wrap;
    }
}

/* Escritorio */
@media (min-width: 901px) {
    .contenedor {
        flex-wrap: nowrap;
    }
}
```

📌 DESKTOP FIRST

Diseñar primero para escritorio y luego adaptar a móviles

```css
/* Estilos base para escritorio */
.contenedor {
    flex-direction: row;
}

/* Tablet */
@media (max-width: 900px) {
    .contenedor {
        flex-wrap: wrap;
    }
}

/* Móvil */
@media (max-width: 600px) {
    .contenedor {
        flex-direction: column;
    }
}
```

🎨 ¿Qué es CSS?

CSS significa **Cascading Style Sheets** (Hojas de Estilo en Cascada).

Es un lenguaje que se usa para **dar estilo y diseño a las páginas web**.

Si lo explicamos de forma sencilla:

- 🧱 **HTML** crea la estructura (como el esqueleto).
- 🎨 **CSS** se encarga de la apariencia (colores, tamaños, posiciones, diseño).

🏠 Ejemplo sencillo

Imagina una casa:

- **HTML** = paredes, puertas y ventanas.
- **CSS** = pintura, decoración, muebles y estilo.

Sin CSS, una página web se vería muy simple, solo texto y elementos básicos.
Con CSS, la página puede tener:

- Colores
- Fondos
- Tipos de letra
- Tamaños diferentes
- Elementos centrados
- Diseños modernos

¿Para qué sirve CSS?

CSS permite:

- 🎨 Cambiar colores
- 🔤 Modificar el tipo y tamaño de letra
- 📦 Ajustar espacios (margen y relleno)
- 📐 Controlar tamaños (ancho y alto)
- 📱 Hacer páginas adaptables a celulares
- 📍 Organizar elementos en la pantalla

🧩 ¿Por qué se llama "en cascada"?

Se llama **cascada** porque los estilos se aplican siguiendo un orden y prioridad.

Si hay varias reglas que afectan al mismo elemento, CSS decide cuál aplicar según:

- Especificidad
- Orden
- Importancia

🧠 Resumen simple

CSS es el lenguaje que hace que una página web:

- Se vea bonita
- Tenga diseño
- Sea ordenada
- Sea profesional

📘 Teoría básica de CSS

1️⃣ * (Selector universal)

- El símbolo * se llama **selector universal**.
- Sirve para aplicar estilos a **todos los elementos** de la página.
- Ejemplo:

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

👉 Esto significa que **todos los elementos** tendrán margen y relleno en 0.

Se usa mucho para "reiniciar" estilos del navegador.

2️⃣ box-sizing

Controla cómo se calcula el tamaño total de un elemento.

```
* {
  box-sizing: border-box;
}
```

Hay dos valores principales:

- content-box (valor por defecto)
- border-box

◇ content-box

El ancho y alto solo incluyen el contenido.
El padding y border se suman aparte.

- **border-box**

El ancho y alto incluyen contenido + padding + border.

Ejemplo recomendado:

👉 Hace que los tamaños sean más fáciles de controlar.

3️⃣ margin

Es el **espacio exterior** de un elemento.

Separa un elemento de otros elementos.

Ejemplo: 👉 El div tendrá 20px de espacio alrededor.

```
div {
  margin: 20px;
}
```

4️⃣ padding

Es el **espacio interior** de un elemento.

Separa el contenido del borde.

Ejemplo: 👉 El texto dentro del div no estará pegado al borde.

```
div {
  padding: 15px;
}
```

5️⃣ height

Define la **altura** de un elemento.

Ejemplo: 👉 El elemento tendrá 200 píxeles de alto.

```
div {
  height: 200px;
}
```

6️⃣ min-height

Define la **altura mínima** que puede tener un elemento.

Ejemplo: 👉 Aunque el contenido sea pequeño, el elemento medirá mínimo 300px.

```
section {
  min-height: 300px;
}
```

7️⃣ max-width

Define el **ancho máximo** que puede tener un elemento.

Ejemplo: 👉 El elemento nunca será más ancho de 800px.

```
div {
  max-width: 800px;
}
```

8️⃣ font-family

Define el tipo de letra del texto.

Ejemplo: 👉 Si no encuentra Arial, usará otra fuente similar.

```
p {
  font-family: Arial, sans-serif;
}
```

9️⃣ font-size

Define el tamaño del texto.

Ejemplo: 👉 El texto tendrá tamaño de 18 píxeles.

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

🔟 background

Define el fondo de un elemento.

Puede cambiar:

- Color
- Imagen
- Degradado

Ejemplo: 👉 El fondo será azul claro.

```
div {
  background: lightblue;
}
```

1️⃣1️⃣ text-align

Alinea el texto horizontalmente.

Valores comunes:

- left
- center
- right
- justify

```
p {
  text-align: center;
}
```

Ejemplo: 👉 El texto se alinea al centro.

1️⃣2️⃣ justify-content

Se usa en **Flexbox** (cuando usamos display: flex).

Sirve para **alinear elementos horizontalmente** dentro de un contenedor.

```
.container {
  display: flex;
  justify-content: center;
}
```

Ejemplo: 👉 Los elementos se colocan en el centro horizontal.

Otros valores:

- flex-start
- flex-end
- space-between
- space-around

1️⃣3️⃣ align-items

También se usa en **Flexbox**.

Sirve para alinear elementos **verticalmente** dentro del contenedor.

```
.container {
  display: flex;
  align-items: center;
}
```

Ejemplo: 👉 Los elementos se centran verticalmente.

🧠 Resumen rápido

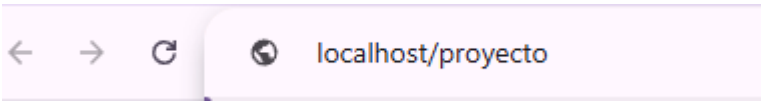
Propiedad	¿Qué controla?
*	Aplica estilo a todo
box-sizing	Cómo se calcula el tamaño
margin	Espacio exterior
padding	Espacio interior

Propiedad	¿Qué controla?
height	Altura
min-height	Altura mínima
max-width	Ancho máximo
font-family	Tipo de letra
font-size	Tamaño del texto
background	Fondo
text-align	Alineación del texto
justify-content	Alineación horizontal en Flexbox
align-items	Alineación vertical en Flexbox

Aplica el diseño responsivo y hojas de estilo en cascada a tu proyecto, para ello, deberás abrir otro bloc de notas y escribir el siguiente código.

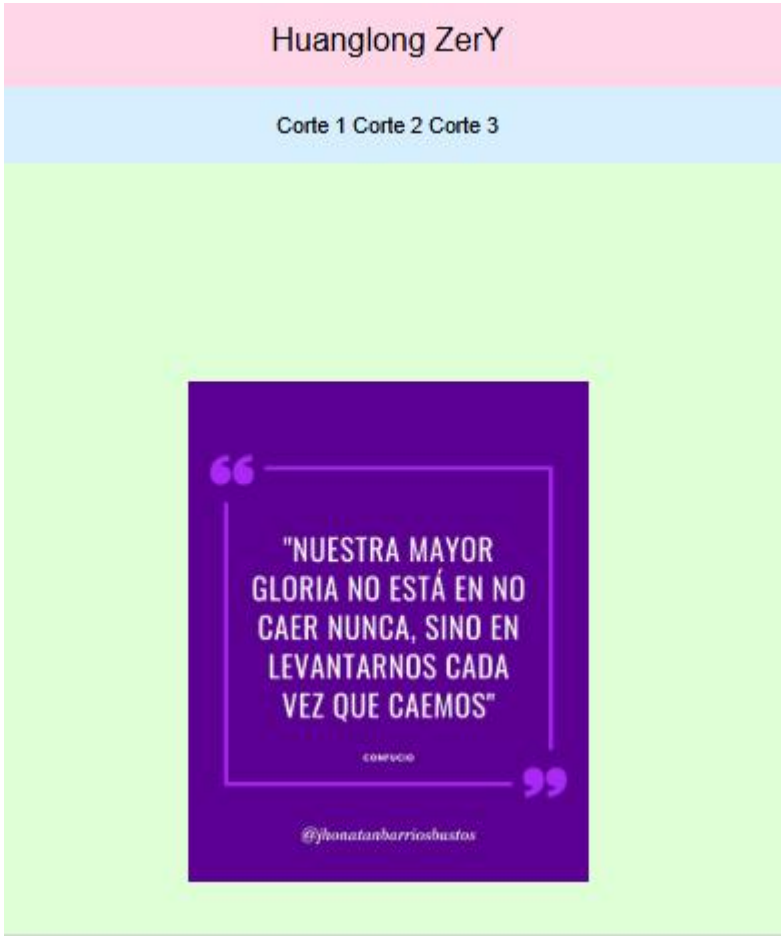
Archivo Editar Ver

```
* {                              /* Resetea toda configuración predeterminada en el navegador*/
    box-sizing: border-box;
    margin: 0;
    padding: 0;
}
html, body {
    height: 100%;           /*se ocupará la todo el alto de la pantalla */
    font-family: Arial, sans-serif;              /* Se establece el tipo de fuente */
}
.global {
    display: flex;
    flex-direction: column;
    min-height: 100vh;
}
/* HEADER */
.encabezado {
    background-color: #FFD6E8;
    padding: 20px;
    text-align: center;
    font-size: 24px;
}
/* CONTENEDOR NAV + MAIN */
.global {
    display: flex;
    flex-direction: column;
}
/* NAV */
.navegacion {
    background-color: #D6EFFF;
    padding: 20px;
    justify-content: center;      /* Centra horizontalmente*/
    display: flex;          /* Se activa el flexbox*/
}
/* MAIN */
.contenido {
    background-color: #DFFFD6;
    padding: 20px;
  /* flex: 1;*/
    min-height: 100vh;     /* Ocupa toda la altura de la ventana*/
    align-items: center;   /* Centra verticalmente*/
    justify-content: center;      /* Centra horizontalmente*/
    display: flex;          /* Se activa el flexbox*/
}
/* Imagen adaptable */
.contenido img {
    max-width: 100%;
    height: auto;
}
/* ---------- DISEÑO PARA PANTALLAS GRANDES ---------- */
@media (min-width: 768px) {
    .global {
        flex-direction: column;
    }
    .navegacion-contenido {
        display: flex;
    }
    .navegacion {
        width: 100%;
    }
    .contenido {
        width: 100%;
    }
}
```

Guarda el archivo anterior con el nombre: **estilos.css, para ver el resultado debes ejectura el index.html** en tu navegador, 

el resultado deberá parecerse como la siguiente imagen.

Recuerda que está enlazado con el archivo de estilos.css.



📘 Introducción a JavaScript

📘 ¿Qué es JavaScript?

JavaScript es un lenguaje de programación que permite agregar **interactividad** a las páginas web.

Si lo explicamos fácil:

- 🧱 **HTML** → estructura
- 🎨 **CSS** → diseño
- ⚡ **JavaScript** → comportamiento e interacción

Gracias a JavaScript podemos:

- Detectar cuando el usuario hace clic
- Cambiar texto o imágenes
- Mostrar mensajes
- Validar formularios
- Crear animaciones

🖱️ ¿Qué es un evento?

Un **evento** es una acción que ocurre en la página web.

Por ejemplo:

- Cuando el usuario hace clic
- Cuando mueve el ratón
- Cuando escribe en un campo
- Cuando pasa el cursor sobre una imagen

JavaScript puede "escuchar" esos eventos y ejecutar una acción.

⭕ Botones en JavaScript

Un botón en HTML se crea así:

```
<button>Haz clic aquí</button>
```

Pero el botón por sí solo no hace nada.
Necesitamos JavaScript para que realice una acción.

🖱️ Evento "click"

El evento más común es el **click**.

Se activa cuando el usuario presiona el botón con el ratón.

Ejemplo sencillo:

```
<button onclick="saludar()">Haz clic</button>

<script>
function saludar() {
  alert("Hola estudiante");
}
</script>
```

¿Qué pasa aquí?

1. El usuario hace clic.
2. Se ejecuta la función saludar().
3. Aparece un mensaje en pantalla.

👉 onclick significa: “cuando se haga clic”.

🖼️ Cambiar una imagen al hacer clic

Podemos cambiar una imagen usando JavaScript.

Ejemplo:

```
<img id="foto" src="imagen1.jpg" width="200">
<button onclick="cambiarImagen()">Cambiar imagen</button>

<script>
function cambiarImagen() {
  document.getElementById("foto").src = "imagen2.jpg";
}
</script>
```

¿Qué sucede?

- getElementById("foto") busca la imagen.
- .src cambia la imagen.
- Al hacer clic, la imagen cambia.
-

🖱️ Evento al pasar el ratón (hover)

También podemos detectar cuando el usuario pasa el cursor sobre una imagen.

Eventos importantes:

- onmouseover → cuando el ratón entra
- onmouseout → cuando el ratón sale

```
<img
  src="imagen1.jpg"
  width="200"
  onmouseover="this.src='imagen2.jpg'"
  onmouseout="this.src='imagen1.jpg'"
>
```

Ejemplo:

¿Qué ocurre?

- Cuando el cursor pasa por encima → cambia la imagen.
- Cuando el cursor sale → vuelve a la original.

👉 this se refiere al mismo elemento (la imagen).

Conceptos importantes

Concepto	¿Qué significa?
Evento	Acción del usuario
click	Cuando se presiona el ratón
función	Bloque de código que ejecuta una acción
getElementById	Busca un elemento por su id
src	Cambia la imagen

📌 ¿Por qué es importante JavaScript?

Porque permite que la página:

- Sea interactiva
- Responda al usuario
- No sea solo texto estático
- Se comporte como una aplicación

Vamos a realizar las siguientes modificaciones en los archivos que hemos venido trabajando que son: index.html y estilos.css, empecemos por el html.

Ábrelo en bloc de notas y agrega lo que indica a continuación en los recuadros:

```
 <nav class="navegacion">
      <!--Aquí van los enlaces o botones de navegación-->
      <button onClick="Corte1()"> Corte 1 </button>
      <button onClick="Corte2()"> Corte 2 </button>
      <button onClick="Corte3()"> Corte 3 </button>
 </nav>

 <main class="contenido">
      <!--Aquí se mostrará el contenido de cada boton o enlace-->


     <img id="foto" src="https://pbs.twimg.com/media/GrVzUztW0AAPn-N.jpg" alt="Motivate"
width="300" onmouseover="cambiarImagen()"    onmouseout="restaurarImagen()">

 </main>

</div>
```

```
<script>
function Corte1(){
   alert ("El Corte 1 está siendo muy interesante");
}
function Corte2(){
   alert ("El Corte 2 está de  lo mejor y vamos por más");
}
function Corte3(){
   alert ("El Corte 3 es el final y me pone triste ");
}
function cambiarImagen(){
    document.getElementById("foto").src = "https://capilea.com/wp-
content/uploads/2024/08/FRASES-2.jpg";
}
function restaurarImagen(){
    document.getElementById("foto").src = "https://pbs.twimg.com/media/GrVzUztW0AAPn-N.jpg";
}
</script>
</body>
</html>
```

Guardamos los cambios y ahora podrás ver que tu proyecto ya tiene interactividad cuando pulsas los botones, pero se te sugiere personalizar la información de los botones.

Al archivo de estilos.css vamos a agregar el siguiente bloque de código, escribelo al final

```
/*---------- Comportamiento de botones--------------- */
.navegacion button {
    position: relative;
    padding: 12px 25px;
    border: none;
    border-radius: 8px;
    font-size: 15px;
    font-weight: 600;
    cursor: pointer;
    color: white;
    background: linear-gradient(135deg, #1e3c72, #2a5298);
    transition: all 0.3s ease;
    box-shadow: 0 8px 15px rgba(0,0,0,0.2);
    overflow: hidden;
}

/* Hover elegante */
.navegacion button:hover {
    transform: translateY(-4px);
    box-shadow: 0 12px 20px rgba(0,0,0,0.3);
}

/* Efecto brillo */
.navegacion button::before {
    content: "";
    position: absolute;
    top: 0;
    left: -75%;
    width: 50%;
    height: 100%;
    background: rgba(255,255,255,0.2);
    transform: skewX(-25deg);
    transition: 0.6s;
}

.navegacion button:hover::before {
    left: 130%;
}

/* Efecto click */
.navegacion button:active {
    transform: scale(0.97);
}
/*---------- fin de comportamiento de botones ---------*/
```

Guarda los cambios del archivo estilos.css y ahora verás que los botones son más atractivos.

Sigamos aprendiendo más.

🗄️ ¿Qué es una Base de Datos?

Una **base de datos** es un lugar donde se guarda información organizada.

📌 Ejemplo en la vida real:

- La lista de alumnos de una escuela
- Los productos de una tienda
- Los contactos del celular

En informática, una base de datos permite:

- Guardar información
- Buscar información
- Modificar información
- Eliminar información

📋 ¿Qué es una Tabla?

Una **tabla** es como una hoja de Excel.

Tiene:

- **Filas** → Cada registro (ejemplo: un alumno)
- **Columnas** → Cada dato del registro (nombre, edad, grupo)

Ejemplo:

id	nombre	edad
1	Ana	16
2	Luis	17

🔑 ¿Qué es una Clave Primaria?

Una **clave primaria (Primary Key)** es un campo que:

- Identifica de forma única cada registro
- No se repite
- No puede estar vacío

Ejemplo:
El campo **id** es clave primaria porque cada alumno tiene un número diferente.

🔗 ¿Qué es una Clave Foránea?

Una **clave foránea (Foreign Key)** es un campo que:

- Conecta una tabla con otra
- Permite relacionar información

Ejemplo:

Tabla alumnos:

id_alumno	nombre
1	Ana

Tabla calificaciones:

id_calificacion	id_alumno	materia
1	1	Matemáticas

Aquí:

- id_alumno en calificaciones es clave foránea
- Se conecta con la clave primaria de alumnos

🔄 ¿Qué es una Relación?

Una **relación** es cuando dos tablas se conectan mediante claves.

Tipos:

- Uno a uno
- Uno a muchos
- Muchos a muchos

Ejemplo común:
Un alumno puede tener muchas calificaciones → Relación uno a muchos.

🛠️ Continua con el proyecto, ahora deberás crear una base de datos para enlazarla y que guarde información.

Ve el sguiente video: https://www.youtube.com/watch?v=zj1BjZK059U

Ahora Crea la base de datos con 1 tabla como se indica a continuación

```
CREATE DATABASE escuela;

USE escuela;

CREATE TABLE alumnos (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(50),
    edad INT,
    grupo VARCHAR(20)
);
```

Una vez que haz creado la base de datos, deberás agregar los siguientes cambios indicados en los cuadros, en el archivo index.html

Agrega un botón en el espacio de navegación

```
<nav class="navegacion">
     <!--Aquí van los enlaces o botones de navegación-->
     <button onClick="Corte1()"> Corte 1 </button>
     <button onClick="Corte2()"> Corte 2 </button>
     <button onClick="Corte3()"> Corte 3 </button>
     <button onclick="mostrarFormulario()"> Registrar Alumno </button>
</nav>
```

En el apartado de contenido realiza los siguientes cambios

```
 <main class="contenido">
      <!--Aquí se mostrará el contenido de cada boton o enlace-->

   <div id="imagenContainer">
      <img id="foto" src="https://pbs.twimg.com/media/GrVzUztW0AAPn-N.jpg" alt="Motivate"
width="300" onmouseover="cambiarImagen()"    onmouseout="restaurarImagen()">
   </div>
<div id="formulario" style="display:none;">
    <h2>Registro de Alumno</h2>
    <form action="guardar.php" method="POST">
        <input type="text" name="nombre" placeholder="Nombre" required><br><br>
        <input type="number" name="edad" placeholder="Edad" required><br><br>
        <input type="text" name="grupo" placeholder="Grupo" required><br><br>
        <input type="submit" value="Guardar">
    </form>
</div>
 </main>
```

En el apartado de los scripts realiza los siguientes cambios.

```
<script>
function Corte1(){
   alert ("El Corte 1 está siendo muy interesante");
   document.getElementById("formulario").style.display = "none";
   document.getElementById("imagenContainer").style.display = "block";
}
function Corte2(){
   alert ("El Corte 2 está de  lo mejor y vamos por más");
   document.getElementById("formulario").style.display = "none";
   document.getElementById("imagenContainer").style.display = "block";
}
function Corte3(){
   alert ("El Corte 3 es el final y me pone triste ");
   document.getElementById("formulario").style.display = "none";
   document.getElementById("imagenContainer").style.display = "block";
}
function cambiarImagen(){
    document.getElementById("foto").src = "https://capilea.com/wp-
content/uploads/2024/08/FRASES-2.jpg";
}
function restaurarImagen(){
    document.getElementById("foto").src = "https://pbs.twimg.com/media/GrVzUztW0AAPn-N.jpg";
}
function mostrarFormulario(){
    document.getElementById("imagenContainer").style.display = "none";
    document.getElementById("formulario").style.display = "block";
}

</script>
```

Deberás crear un nuevo archivo de bloc de notas y escribir lo siguiente:

```
<?php
$conexion = new mysqli("localhost", "root", "", "escuela");
if ($conexion->connect_error) {
    die("Error de conexión: " . $conexion->connect_error);
}
$nombre = $_POST['nombre'];
$edad = $_POST['edad'];
$grupo = $_POST['grupo'];
$sql = "INSERT INTO alumnos (nombre, edad, grupo)
        VALUES ('$nombre', '$edad', '$grupo')";
if ($conexion->query($sql) === TRUE) {
    // Mensaje atractivo con estilo
    echo '
    <html>
    <head>
        <title>Alumno Guardado</title>
        <link rel="stylesheet" href="estilos.css">
        <style>
            .mensaje {
                background-color: #DFFFD6;  /* igual que tu main */
                color: #333;
                border: 2px solid #2a5298;  /* borde azul oscuro estilo botón */
                padding: 30px 50px;
                border-radius: 12px;
                font-size: 20px;
                text-align: center;
                font-weight: bold;
                box-shadow: 0 8px 15px rgba(0,0,0,0.2);
                max-width: 500px;
                margin: 150px auto;
                animation: aparecer 0.5s ease;
            }

            @keyframes aparecer {
                from {opacity: 0; transform: scale(0.8);}
                to {opacity: 1; transform: scale(1);}
            }
        </style>
```

```
        <script>
            // Redirigir automáticamente después de 2 segundos
            setTimeout(function(){
                window.location.href = "index.html";
            }, 4000);
        </script>
    </head>
    <body>
        <div class="mensaje"> Alumno guardado correctamente.<br>Redirigiendo...</div>
    </body>
    </html>
    ';
} else {
    echo '
    <html>
    <head>
        <title>Error</title>
        <link rel="stylesheet" href="estilos.css">
        <style>
            .mensaje-error {
                background-color: #FFD6D6;  /* fondo rojo suave */
                color: #b71c1c;
                border: 2px solid #b71c1c;
                padding: 30px 50px;
                border-radius: 12px;
                font-size: 20px;
                text-align: center;
                font-weight: bold;
                max-width: 500px;
                margin: 150px auto;
                box-shadow: 0 8px 15px rgba(0,0,0,0.2);
            }
        </style>
    </head>
    <body>
        <div class="mensaje-error"> Error al guardar el alumno: ' . $conexion->error .
'</div>
    </body>
    </html>
    ';
}
$conexion->close();
?>
```

Guardalo con el nombre: guardar.php tambien en la carpeta: proyecto

🎯 ¿Qué hace éste archivo aquí?

1. El formulario envía datos a guardar.php
2. PHP se conecta a MySQL
3. Inserta los datos en la tabla
4. Muestra mensaje de confirmación
5. Después de unos segundos Regresa al archivo inicial index.html

Una alternativa al archivo anterior es...

```php
<?php
$conexion = new mysqli("localhost", "root", "", "escuela");

if ($conexion->connect_error) {
    die("Error de conexión: " . $conexion->connect_error);
}

$nombre = $_POST['nombre'];
$edad = $_POST['edad'];
$grupo = $_POST['grupo'];

$sql = "INSERT INTO alumnos (nombre, edad, grupo)
        VALUES ('$nombre', '$edad', '$grupo')";

if ($conexion->query($sql) === TRUE) {
    header("Location: index.html?mensaje=ok");
    exit();
    echo "<button><a href=index.html> Regresar </a></button>";
} else {
    header("Location: index.html?mensaje=error");
    exit();
}

$conexion->close();
?>
```

¿Qué hace este archivo, pues al igual que el anterior, guarda la información en la base de datos, solo que no nos avisa de tener el registro guardado, por lo que te sugiero que pruebes ambos, solo para conocerlos y probarlos?

Es hora de probar tu proyecto, que no tenga errores

En un documento de Word se te sugiere que elabores un reporte de trabajo de este proyecto, que agregues capturas de pantalla como evidencia de tu trabajo y expresa que te ha parecido esta actividad, agrega una portada con tus datos, imprimelo, lo entregarás en jefaturas.