ESCUELA PROFESIONAL DE INGENIERÍA DE SISTEMAS
INTRODUCCIÓN AL DESARROLLO WEB
2025 II
LABORATORIO 09 – CSS: MODELO DE CAJAS Y POSICIONAMIENTO

I

OBJETIVOS

- Comprender la importancia de CSS en el desarrollo web
- Aplicar estilos relacionados al modelo de cajas y posicionamiento
- Solucionar ejercicios aplicando los conocimientos obtenidos

TIEMPO ESTIMADO: 2 horas

II

CONSIDERACIONES DE EVALUACIÓN

- Se deberán utilizar los conocimientos impartidos en las clases teóricas
- Deberá utilizar nombre de variables significativos
- Deberá realizar pruebas adicionales
- El alumno deberá indicar en su código con quien colaboró, así sea la IA
- El alumno será requerido de realizar modificaciones en su código y responder a preguntas sobre el mismo
- Los ejercicios deberán realizarse en el laboratorio, a su ritmo y subirlos al aula virtual como AVANCE antes de finalizar la clase
- Todos los ejercicios completos, deberán ser subidos al aula virtual como TAREA, para lo cual se les dará un tiempo adecuado y deben cumplir el deadline estipulado. Esto se deberá cumplir, aunque en el laboratorio ya se hayan terminado todos los ejercicios
- El formato a usar para los avances y tareas será .zip, que contenga todos los archivos requeridos
- Utilizar Git con GitHub para el manejo de versiones de su trabajo, ocasionalmente se le solicitará que compartan sus repositorios

III

POLITICA DE COLABORACION

La política del curso es simple, a menos que se exprese lo contrario en el laboratorio, siéntase libre de colaborar con sus compañeros en todos los laboratorios, pero debe notificar expresamente con quien ha colaborado. La colaboración con alumnos, que no están matriculados en el curso está prohibida. Los laboratorios y asignaciones han sido desarrollados para ayudarlo a comprender el material. Conozca su código y esté preparado para revisiones individuales de código. Durante las revisiones es probable que se le pida realizar modificaciones y justificar sus decisiones de programación. Cada uno de sus ejercicios debe iniciar de la siguiente forma:

```
// Laboratorio  Nro x - Ejerciciox
// Autor: mi nombre
// Colaboró : el nombre
// Tiempo :
```

INDICACIONES GENERALES

a. En cada sesión de laboratorio, los ejercicios propuestos deberán ser guardados en la misma carpeta/directorio
b. La carpeta deberá tener el nombre del Laboratorio y el nombre del alumno, así por ejemplo:
Laboratorio 11 – Juan Perez
c. Utilice nombres significativos
d. Su código deberá estar correctamente indentado y preferentemente documentado
e. Deberá ser debidamente probado

MARCO TEORICO

1. Modelo de cajas

Cada elemento HTML se representa como una caja rectangular. Esta caja está compuesta por varias capas que determinan su tamaño y separación con respecto a otros elementos.
El modelo de cajas define:

- Contenido (content) → el texto o imagen.
- Relleno (padding) → espacio interno entre el contenido y el borde.
- Borde (border) → línea alrededor del elemento.
- Margen (margin) → espacio externo hacia otros elementos.

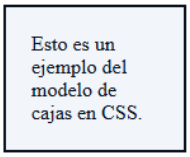
La propiedad box-sizing define **cómo se calcula el ancho (**width**) y alto (**height**) de un elemento**, es decir, si esos valores incluyen **padding y borde** o sólo el contenido

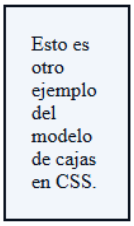
```
box-sizing: content-box; /* por defecto, el width sólo incluye el contenido */
box-sizing: border-box;  /* el width incluye padding y border en width y height */
```

```
<div class="caja con-content">
    Esto es un ejemplo del modelo de cajas en CSS.
</div>
<div class="caja con-border">
    Esto es otro ejemplo del modelo de cajas en CSS.
</div>
```

```
.caja {
    width: 100px; /* ancho del contenido */
    padding: 20px; /* espacio interno */
    border: 2px solid #0b1320; /* borde */
    margin: 30px; /* espacio externo */
    background: #f2f6fb; /* color de fondo */
    color: #0b1320; /* color del texto */
}

.con-content {
    box-sizing: content-box; /* valor por defecto, sólo para contenido */
}
.con-border {
    box-sizing: border-box; /* incluye padding y border en el ancho */
}
```





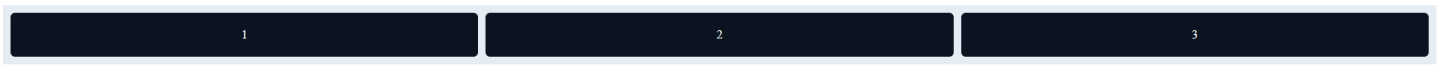
2. Flexbox

Flexbox (Flexible Box Layout) es un modelo de diseño en CSS que permite organizar elementos dentro de un contenedor de manera flexible, ya sea horizontal o vertical (unidimensional), ajustando automáticamente su tamaño, alineación y distribución del espacio.
El contenedor flex controla la disposición de los elementos internos (flex items).

```
<h2>Usando Flexbox</h2>
<div class="flex-container">
    <div class="flex-item">1</div>
    <div class="flex-item">2</div>
    <div class="flex-item">3</div>
</div>
```

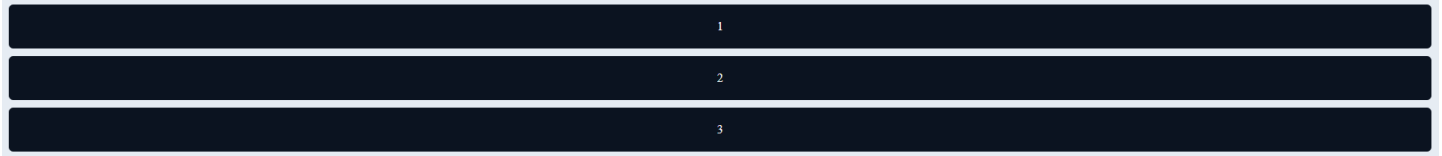
```
.flex-container {
    display: flex;
    gap: 10px; /* espacio entre ítems */
    background: #e6ecf3;
    padding: 10px;
}
.flex-item {
    flex: 1; /* todos crecen igual */
    background: #0b1320;
    color: #fff;
    text-align: center;
    padding: 20px;
    border-radius: 6px;
}
```

Usando Flexbox



Si se aumenta en el contenedor: `flex-direction: column;`

Usando Flexbox

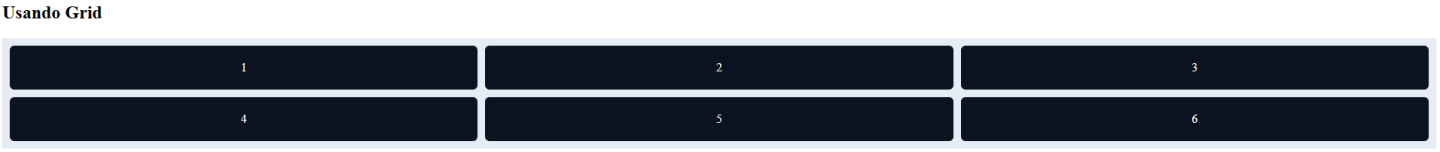


3. Grid

CSS Grid Layout es un sistema de diseño bidimensional en CSS que permite organizar los elementos en filas y columnas. A diferencia de Flexbox (que trabaja mejor en una sola dirección, fila o columna), Grid está pensado para manejar ambas dimensiones al mismo tiempo.

```html
<h2>Usando Grid</h2>
<div class="grid-container">
    <div class="grid-item">1</div>
    <div class="grid-item">2</div>
    <div class="grid-item">3</div>
    <div class="grid-item">4</div>
    <div class="grid-item">5</div>
    <div class="grid-item">6</div>
</div>
```

```css
.grid-container {
    display: grid;
    grid-template-columns: repeat(3, 1fr); /* 3 columnas iguales */
    gap: 10px;
    background: #e6ecf3;
    padding: 10px;
}
.grid-item {
    background: #0b1320;
    color: #fff;
    text-align: center;
    padding: 20px;
    border-radius: 6px;
}
```



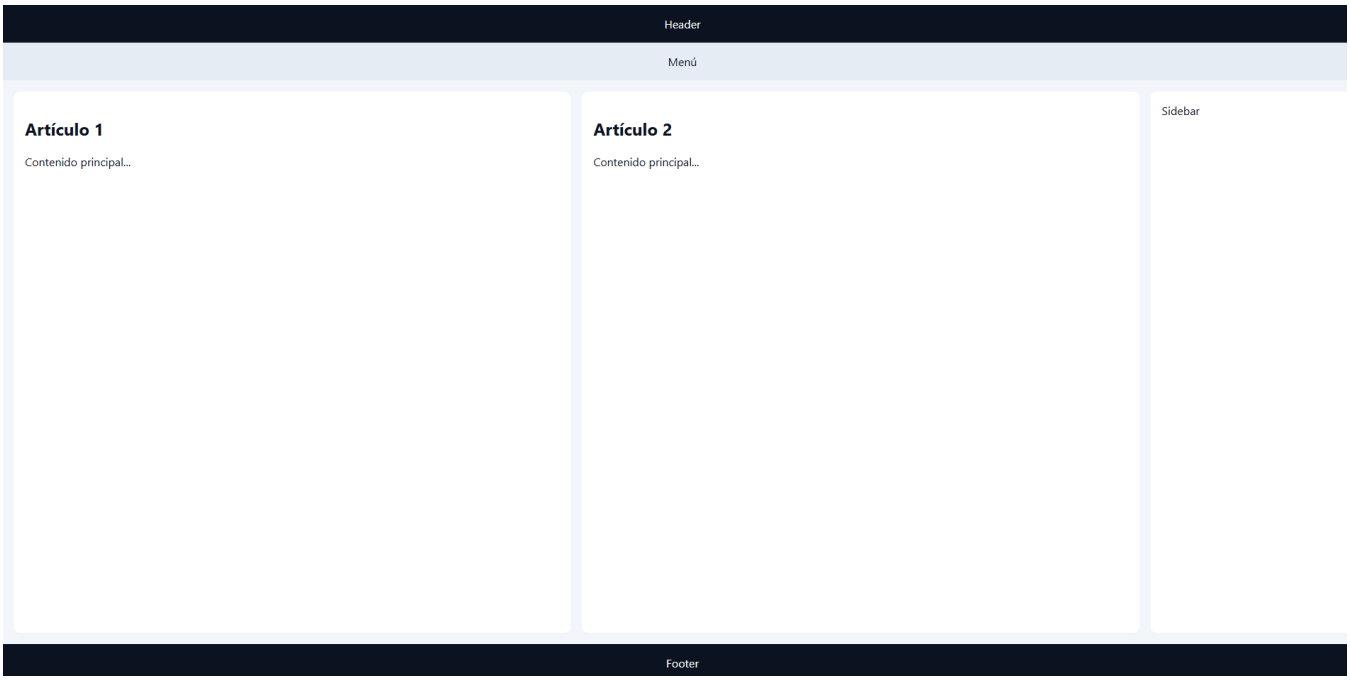
Probar cambiando el tamaño de la ventana del browser.
Cambiar 3 por 4 en `grid-template-columns`

4. **Posicionamiento avanzado**

Se pueden combinar flex y grid. Grid para el layout general y Flexbox para alinear los ítems internos.

```
<header>Header</header>
<nav>Menú</nav>
<main>
    <article>
        <h2>Artículo 1</h2>
        <p>Contenido principal...</p>
    </article>
    <article>
        <h2>Artículo 2</h2>
        <p>Contenido principal...</p>
    </article>
    <aside>Sidebar</aside>
</main>
<footer>Footer</footer>
```

```
body {
  margin: 0;
  font-family: system-ui, sans-serif;
  color: #0b1320;
  background: #f2f6fb;
  min-height: 100vh;
  /* GRID para el layout principal */
  display: grid;
  grid-template-rows: auto auto 1fr auto; /* header, nav, main, footer */
}
header, nav, footer {
  padding: 16px;
  text-align: center;
}
header, footer {
  background: #0b1320;
  color: #fff;
}
nav {
  background: #e6ecf3;
}
/* MAIN con Flex */
main {
  display: flex;         /* aquí entra Flexbox */
  gap: 16px;
  padding: 16px;
}
/* Article y aside dentro de main */
article {
  flex: 3;
  background: #fff;
  padding: 16px;
  border-radius: 8px;
}
aside {
  flex: 1;
  background: #fff;
  padding: 16px;
  border-radius: 8px;
}
```



VI

EJERCICIOS PROPUESTOS

1. Crear un directorio que tenga su nombre y un subdirectorio laboratorio08. Recomendación: usar minúsculas, sin espacios, sin tildes ni “ñ” y con guiones medios o bajos
2. Utilizar los atajos de teclado o combinación de teclas para agilizar su trabajo
3. Crea una página index.html que tenga enlaces a las siguientes 5 páginas
4. boxmodel.html
   Usa tarjetas considerando el modelo de cajas y Flex
   Objetivo: Crear un contenedor con varias tarjetas (div), aplicando márgenes, bordes, padding y usando Flexbox para alinearlas.
   - Cada tarjeta debe tener un borde, padding y margen entre sí
   - Usa display: flex; justify-content: space-around;

5. gridflex.html
   Layout de Blog con Grid + Flex
   Objetivo: Crear la estructura de un blog.
   - Usa CSS Grid para dividir en: header, nav, main, article, aside, footer
   - Dentro de main, usa Flexbox para organizar artículos y darles el mismo ancho
   - Aplica el modelo de cajas con padding y border a cada artículo

6. Imágenes.html
   Galería de Imágenes Responsive
   Objetivo: Hacer una galería de imágenes.
   - Usa Grid para generar una cuadrícula responsive (ej. grid-template-columns: repeat(auto-fit, minmax(200px, 1fr));)
   - Cada imagen dentro de la cuadrícula debe estar dentro de un contenedor con padding, border y margin
   - Usa box-sizing: border-box; para que los bordes y paddings no rompan el diseño

7. dashboard.html
   Dashboard con Sidebar
   Objetivo: Crear un diseño tipo panel de administración.
   - Usa Grid para dividir en dos columnas: sidebar y contenido
   - Dentro del contenido, usa Flexbox para organizar tarjetas de estadísticas
   - Cada tarjeta debe mostrar el modelo de cajas: border, padding, margin

Una idea base:

8. Landing Page Simple
   Objetivo: Hacer una landing page básica.
   - Usa Grid para organizar: header, hero, main, footer
   - En la sección hero, usa Flexbox para centrar vertical y horizontalmente un título y un botón
   - Cada sección debe mostrar uso de modelo de cajas (padding interno, márgenes externos, bordes redondeados)

9. Crear un repositorio remoto en GitHub y subir tu repositorio local. Compartir URL
10. Continuar con su Proyecto Personal. Utilizar todo lo realizado en este laboratorio. Utilizar su repositorio remoto en GitHub. Compartir URL

I. TAREA PARA LA CASA: Complete todos los ejercicios.

Crear un documento docx/pdf con la solución de los ejercicios.

Subir el documento a la tarea **Tarea 09** del Aula Virtual respetando las fechas indicadas.