Explicación del Código

1. **HEADER (<header>)**
   - Representa la cabecera de la página.
   - Se usa background-color: #ffcc00; para que tenga un color amarillo.
   - Se centra el texto con text-align: center;.
2. **NAV (<nav>)**
   - Contiene enlaces de navegación (Inicio, Carreras, Admisiones, Contacto).
   - Se usa background-color: #ff9900; para destacar el menú.
   - Los enlaces tienen margin: 0 15px; para separarlos.
3. **ASIDE (<aside>)**
   - Es una barra lateral con noticias.
   - Se coloca a la izquierda usando width: 20% dentro de .content.
4. **MAIN (<main>)**
   - Es la parte principal del contenido.
   - Usa flex: 1; para ocupar el espacio restante.
5. **SECTION (<section>)**
   - Agrupa información relacionada dentro del <main>.
   - Tiene un color de fondo diferente #ff6666.
6. **ARTICLE (<article>)**
   - Contiene información específica dentro de una <section>.
   - Tiene un borde con border: 1px solid #ddd;.
7. **FOOTER (<footer>)**
   - Representa el pie de página.
   - Se coloca abajo con text-align: center; y un fondo oscuro #333.

¿Cómo se posiciona en pantalla?

- **display: flex;** en .content organiza el aside y main en filas.
- **flex-direction: column;** en .container coloca los elementos en columna.
- **El aside ocupa el 20%** y el main el **80% restante** con flex: 1;.

EXPLICACIÓN GENERAL

1. * { margin: 0; padding: 0; box-sizing: border-box; }

Esta regla CSS se llama **"reset CSS"** y se aplica a **todos los elementos** de la página (* es un selector universal).

- margin: 0; → Elimina los márgenes predeterminados de los elementos.
- padding: 0; → Elimina el relleno interno predeterminado.
- box-sizing: border-box; → Hace que el tamaño total del elemento **incluya** el padding y el borde, en lugar de sumar el ancho y alto.

□ Otras opciones para box-sizing:

- content-box (por defecto) → El ancho y alto **no** incluyen el padding ni el borde.
- inherit → Hereda el valor del elemento padre.

---

2. .container { display: flex; flex-direction: column; min-height: 100vh; }

Este es el **contenedor principal** que organiza el diseño en una columna usando flexbox.

- display: flex; → Activa el modelo **Flexbox**, permitiendo organizar los elementos internos.
- flex-direction: column; → Coloca los elementos hijos **en columna** (uno debajo del otro).
- min-height: 100vh; → Asegura que la página ocupe al menos el **100% del alto de la pantalla**.

□ Otras opciones para flex-direction:

- row (por defecto) → Los elementos se colocan en fila.
- row-reverse → Invierte el orden de la fila.
- column-reverse → Invierte el orden de la columna.

---

3. header { background-color: #ffcc00; text-align: center; padding: 20px; font-size: 24px; }

Define el estilo del **encabezado** (<header>).

- background-color: #ffcc00; → Fondo amarillo (#ffcc00).
- text-align: center; → Centra el texto dentro del <header>.
- padding: 20px; → Agrega un espacio interno de 20px.
- font-size: 24px; → Aumenta el tamaño del texto a 24px.

□ Otras opciones para text-align:

- left → Alinea el texto a la izquierda.
- right → Alinea el texto a la derecha.
- justify → Justifica el texto (alineado en ambos lados).

---

4. nav { background-color: #ff9900; padding: 10px; text-align: center; }

Define el estilo del **menú de navegación** (<nav>).

- background-color: #ff9900; → Fondo naranja.
- padding: 10px; → Espacio interno de 10px.
- text-align: center; → Centra los enlaces dentro del <nav>.

---

5. nav a { color: white; text-decoration: none; margin: 0 15px; font-size: 18px; }

Estiliza los enlaces (<a>) dentro del <nav>.

- color: white; → Color de texto blanco.
- text-decoration: none; → Elimina el subrayado predeterminado de los enlaces.
- margin: 0 15px; → Añade un **espacio horizontal** de 15px entre enlaces.
- font-size: 18px; → Hace el texto más grande (18px).

□ Otras opciones para text-decoration:

- underline → Subraya el texto.
- overline → Línea sobre el texto.
- line-through → Texto tachado.

---

6. .content { display: flex; flex: 1; }

Define la disposición de <aside> y <main>.

- display: flex; → Activa **Flexbox** para organizar los elementos en fila.
- flex: 1; → Permite que el contenedor ocupe todo el espacio disponible.

---

7. aside { background-color: #66ccff; width: 20%; padding: 20px; text-align: center; }

Define la **barra lateral** (<aside>).

- background-color: #66ccff; → Fondo azul claro.
- width: 20%; → Ocupa el **20%** del ancho de la pantalla.
- padding: 20px; → Añade un **espacio interno** de 20px.
- text-align: center; → Centra el texto dentro del <aside>.

---

8. main { background-color: #ccff99; flex: 1; padding: 20px; }

Define el área **principal** (<main>).

- background-color: #ccff99; → Fondo verde claro.
- flex: 1; → Ocupa el **80% restante** del ancho disponible.
- padding: 20px; → Añade un **espacio interno** de 20px.

---

9. section { background-color: #ff6666; padding: 15px; margin-bottom: 20px; }

Define las **secciones** (<section>) dentro del <main>.

- background-color: #ff6666; → Fondo rojo claro.
- padding: 15px; → Añade un **espacio interno** de 15px.
- margin-bottom: 20px; → Añade **separación** de 20px entre secciones.

---

10. article { background-color: #ffffff; padding: 10px; border: 1px solid #ddd; margin-top: 10px; }

Define los **artículos** (<article>) dentro de <section>.

- background-color: #ffffff; → Fondo blanco.
- padding: 10px; → Añade un **espacio interno** de 10px.
- border: 1px solid #ddd; → Borde gris claro.
- margin-top: 10px; → Añade **espacio superior** de 10px.

---

11. footer { background-color: #333; color: white; text-align: center; padding: 15px; }

Define el **pie de página** (<footer>).

- background-color: #333; → Fondo negro grisáceo.
- color: white; → Texto blanco.
- text-align: center; → Centra el texto dentro del <footer>.
- padding: 15px; → Añade un **espacio interno** de 15px.

⍰ Para entenderlo, tenemos que:

- Se usa **Flexbox** para organizar aside y main en **filas**.
- Se asigna width: 20% a aside y flex: 1; a main para dividir el espacio correctamente.
- Se usan colores distintos para visualizar mejor cada sección.
- Se eliminan márgenes y paddings predeterminados con box-sizing: border-box;.

INTRUCCIONES DETALLADAS

margin en CSS

¿Qué hace?

La propiedad margin en CSS **define el espacio exterior** de un elemento, es decir, la distancia entre ese elemento y otros elementos adyacentes en la página.

Es diferente de padding, que define el espacio interno dentro del borde del elemento.

Valores que puede tomar margin

La propiedad margin puede tomar varios tipos de valores:

Valores en píxeles, porcentajes u otras unidades

Se puede especificar en:

- **Píxeles** → margin: 20px; (20 píxeles de margen en todas las direcciones)
- **Porcentajes** → margin: 10%; (10% del ancho del contenedor padre)
- **Unidades relativas**
  - em (relativo al tamaño de la fuente) → margin: 2em;
  - rem (relativo a la raíz html) → margin: 1rem;
  - vw (porcentaje del ancho de la ventana) → margin: 5vw;
  - vh (porcentaje del alto de la ventana) → margin: 10vh;

Valores específicos para cada lado

Puedes definir los márgenes de cada lado **por separado**:

```
margin-top: 10px;    /* Margen superior */
margin-right: 20px;  /* Margen derecho */
margin-bottom: 30px; /* Margen inferior */
margin-left: 40px;   /* Margen izquierdo */
```

También puedes combinar los valores en una sola línea:

```
 margin: 10px 20px 30px 40px;
/* Margen superior 10px, derecho 20px, inferior 30px, izquierdo 40px */
```

 Orden en que se aplican los valores en margin:

margin: top right bottom left;

Ejemplos:

```
margin: 10px 20px;
/* 10px arriba/abajo, 20px izquierda/derecha */

margin: 10px 20px 30px;
/* 10px arriba, 20px izquierda/derecha, 30px abajo */
```

---

auto (centrado horizontal)

El valor auto se usa principalmente para **centrar un elemento horizontalmente** cuando tiene un ancho definido:

```
div {
  width: 50%;
  margin: 0 auto;  /* Centra horizontalmente */
}
```

---

inherit, initial, unset

- inherit → Hereda el valor del margin de su elemento padre.
- initial → Restaura el margen a su valor por defecto (0).
- unset → Si el padre tiene margin, lo hereda; si no, lo pone en initial (0).

Ejemplo:

```
p {
  margin: inherit;  /* Tomará el margen de su contenedor padre */
}
```

Ejemplo visual de margin

```
.box {
  width: 200px;
  height: 100px;
  background: lightblue;
  margin: 20px auto 40px;
}
```

Explicación

- Margen **superior** → 20px
- Margen **inferior** → 40px
- Margen **izquierda/derecha** → auto (lo centra)

Diferencias entre margin y padding

Propiedad	¿Qué ajusta?	Afecta el tamaño total del elemento
margin	Espacio **externo**	No afecta el tamaño total
padding	Espacio **interno**	Hace que el elemento sea más grande

---

Conclusión:

- margin **crea espacio fuera del elemento**.
- Se puede usar en **píxeles, %, auto, em, etc.**
- Se puede definir en **todos los lados o de forma individual**.
- margin: auto; **centra el elemento horizontalmente** cuando tiene un ancho definido.

margin: 2em; en

La propiedad margin: 2em; establece un margen de **2 veces el tamaño de la fuente actual** en **todos los lados** del elemento (arriba, abajo, izquierda y derecha).

---

em

- **em** es una unidad relativa en CSS que se **basa en el tamaño de la fuente del elemento o de su contenedor padre**.
- **1em** equivale al tamaño de la fuente (font-size) del elemento en cuestión.
- **2em** significa **2 veces** el tamaño de la fuente.

Ejemplo práctico de margin: 2em;

```
p {
  font-size: 16px; /* Tamaño de fuente */
  margin: 2em;  /* Margen = 2 × 16px = 32px */
}
```

Explicación:

- Si la fuente (font-size) del <p> es **16px**, entonces **2em equivale a 32px de margen en todos los lados**.

Ejemplo visual

```
div {
  font-size: 20px;
  background: lightblue;
  margin: 2em;
```

}

Si el tamaño de fuente es 20px:

- 2em será 2 × 20px = 40px de margen en cada lado.

¿Por qué usar em en lugar de px?

 Ventaja: Se adapta automáticamente si el tamaño de la fuente cambia.
 Desventaja: Puede ser complicado de calcular cuando hay elementos anidados.

Alternativa: Si necesitas un margen basado en la raíz del documento, usa rem en lugar de em:

margin: 2rem;

2rem siempre será 2 × font-size del <html> (generalmente 16px en los navegadores).

padding

La propiedad padding **controla el espacio interno** de un elemento, es decir, el espacio **entre el contenido y el borde del elemento**.

Diferencia entre margin y padding

Propiedad	¿Dónde afecta?
margin	**Espacio externo** (fuera del borde del elemento)
padding	**Espacio interno** (dentro del borde, alrededor del contenido)

Ejemplo visual

```
div {
  background: lightblue;
  padding: 20px;
}
```

Esto crea **20px de espacio dentro del div**, separando su contenido del borde.

Valores que puede tomar padding

Un solo valor

padding: 20px;

Aplica **20px en todos los lados** (arriba, abajo, izquierda y derecha).

Cuatro valores (arriba, derecha, abajo, izquierda)

padding: 10px 20px 30px 40px;

Se aplican en este orden:

□ **Arriba** → □ **Derecha** → ◀□ **Abajo** → ▶□ **Izquierda**
Ejemplo visual:

- 10px arriba
- 20px derecha
- 30px abajo
- 40px izquierda

Tres valores (arriba, lados, abajo)

padding: 10px 20px 30px;

Se aplican así:

- 10px **arriba**
- 20px **izquierda y derecha**
- 30px **abajo**

Dos valores (vertical, horizontal)

padding: 10px 20px;

Se aplican así:

- 10px **arriba y abajo**
- 20px **izquierda y derecha**

---

padding individual para cada lado

```
padding-top: 10px;
padding-right: 20px;
padding-bottom: 30px;
padding-left: 40px;
```

Cada lado se configura por separado.

Unidades que puede usar padding

padding acepta diferentes unidades:

Unidad	Descripción
px	Píxeles (tamaño fijo)
em	Relativo al tamaño de fuente del elemento
rem	Relativo al tamaño de fuente del <html>
%	Relativo al ancho del elemento padre
vh/vw	Relativo al alto (vh) o ancho (vw) de la pantalla

Ejemplo con em

```
div {
  font-size: 16px;
  padding: 2em; /* 2 × 16px = 32px */
}
```

2em se ajustará automáticamente si cambia el tamaño de fuente.

Ejemplo práctico

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Ejemplo de Padding</title>
   <style>
      .box {
         background-color: lightblue;
         padding: 20px; /* Espacio interno de 20px */
         border: 2px solid blue;
      }
   </style>
</head>
<body>
```

```
  <div class="box">
    Esto es un contenedor con `padding: 20px;`
  </div>
</body>
</html>
```

Visualmente, el texto dentro del div no toca los bordes gracias al padding de 20px.

Entonces:

- padding controla **el espacio interno** de un elemento.
- Puede tomar valores en **píxeles (px), unidades relativas (em, %, etc.)**.
- Se puede aplicar **individualmente** (padding-top, padding-right, etc.) o en **una sola línea** (padding: 10px 20px;).

border en CSS

Define **el borde de un elemento**.

Valores posibles

border: <ancho> <tipo> <color>;

- **Ancho:** px, em, %, thin, medium, thick
- **Tipo:** solid, dotted, dashed, double, groove, ridge, inset, outset, none
- **Color:** red, #ff0000, rgb(255, 0, 0), transparent

Ejemplo

```
div {
  border: 2px solid blue;
}
```

---

background-color en CSS

Establece **el color de fondo** de un elemento.

Valores posibles

- Nombres de color: red, blue, green
- Código hexadecimal: #ff0000
- rgb(): rgb(255, 0, 0)
- rgba(): rgba(255, 0, 0, 0.5) (con opacidad)
- hsl(): hsl(0, 100%, 50%)
- transparent: Sin color de fondo

Ejemplo

```
div {
  background-color: lightgray;
}
```

---

margin-top en CSS

Define **el espacio externo arriba** de un elemento.

Valores posibles

- px, %, em, rem, vh
- auto: Se ajusta automáticamente
- inherit: Hereda el valor del elemento padre

Ejemplo

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

---

text-align en CSS

Controla **la alineación del texto** dentro de un elemento.

Valores posibles

- left: Alineado a la izquierda
- right: Alineado a la derecha
- center: Centrado
- justify: Justificado

Ejemplo

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

---

margin-bottom en CSS

Define **el espacio externo debajo** de un elemento.

Valores posibles

Mismos que margin-top (px, %, auto, inherit).

Ejemplo

```
div {
  margin-bottom: 30px;
}
```

---

flex en CSS

Define cómo un **elemento hijo** crece o se encoge dentro de un contenedor display: flex;.

Valores posibles

flex: <grow> <shrink> <basis>;

- grow: Factor de crecimiento (1, 2, 3, etc.)
- shrink: Factor de reducción (0, 1, 2, etc.)
- basis: Tamaño base (px, %, auto)

Ejemplo

```
.item {
  flex: 1; /* Todos los elementos ocupan el mismo espacio */
}
```

---

display: flex en CSS

Convierte un elemento en un **contenedor flexible**, permitiendo organizar sus hijos.

Valores posibles

- flex: Activa el modelo flexible
- inline-flex: Como flex, pero en línea

Ejemplo

```
.container {
  display: flex;
}
```

---

text-decoration en CSS

Controla **el estilo del texto** (subrayado, tachado, etc.).

Valores posibles

- none: Sin decoración
- underline: Subrayado

- overline: Línea arriba del texto
- line-through: Texto tachado
- blink: (obsoleto)

Ejemplo

```
a {
  text-decoration: none;
}
```

---

font-size en CSS

Controla **el tamaño del texto**.

Valores posibles

- px, %, em, rem, vh, vw
- larger, smaller
- xx-small, x-small, small, medium, large, x-large, xx-large

Ejemplo

```
p {
  font-size: 20px;
}
```

---

flex-direction en CSS

Controla **la dirección de los elementos flexibles**.

Valores posibles

- row: De izquierda a derecha (▯▯▯)
- row-reverse: De derecha a izquierda (▯▯▯)
- column: De arriba a abajo (⬆⬆⬆)
- column-reverse: De abajo a arriba (⬇⬇⬇)

Ejemplo

```
.container {
  display: flex;
  flex-direction: column;
}
```

---

min-height en CSS

Define **la altura mínima** de un elemento.

Valores posibles

- px, %, vh, vw
- auto: Se ajusta automáticamente

Ejemplo

```
div {
  min-height: 100px;
}
```

---

box-sizing en CSS

Controla **cómo se calculan las dimensiones** de un elemento.

Valores posibles

- content-box (por defecto): No incluye padding ni border en el tamaño total.
- border-box: Incluye padding y border en el tamaño total.

Ejemplo

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

box-sizing en CSS: Explicación detallada

La propiedad **box-sizing** en CSS **define cómo se calculan las dimensiones de un elemento HTML** (ancho y alto), teniendo en cuenta o no el padding y el border dentro de esas dimensiones.

Problema sin box-sizing

Por defecto, cuando estableces un width y height en CSS, **solo se aplica al contenido** del elemento.
Si luego agregas padding o border, estos **se suman al tamaño total**, lo que puede romper diseños y causar inconsistencias.

Ejemplo sin box-sizing:

```
div {
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 5px solid black;
}
```

Cálculo real del tamaño del elemento:

- Ancho total = 200px (contenido) + 20px*2 (padding) + 5px*2 (borde) = 250px
- Alto total = 100px (contenido) + 20px*2 (padding) + 5px*2 (borde) = 150px

Solución con box-sizing: border-box

Si usamos box-sizing: border-box;, **el width y height incluyen el padding y el border dentro del tamaño total**.

Ejemplo con box-sizing: border-box:

```
div {
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 5px solid black;
  box-sizing: border-box;
}
```

Cálculo real del tamaño del elemento:

- Ancho total = 200px (el contenido se ajusta para incluir padding y border).
- Alto total = 100px.

Ventaja: Mantiene el diseño uniforme y evita que los elementos se expandan inesperadamente.

Valores de box-sizing y su comportamiento

content-box (valor predeterminado)

Solo el contenido respeta el width y height, pero **el padding y border se agregan al tamaño total**.

```
div {
  width: 200px;
  height: 100px;
  padding: 20px;
  border: 5px solid black;
  box-sizing: content-box; /* El tamaño final será mayor */
}
```

border-box (recomendado)

width y height **incluyen el padding y border**, asegurando que el tamaño total no cambie.

```
div {
  width: 200px;
```

```
  height: 100px;
  padding: 20px;
  border: 5px solid black;
  box-sizing: border-box; /* Tamaño fijo */
}
```

Ejemplo práctico con ambas opciones

```
<!DOCTYPE html>
<html lang="es">
<head>
   <meta charset="UTF-8">
   <meta name="viewport" content="width=device-width, initial-scale=1.0">
   <title>Ejemplo box-sizing</title>
   <style>
      .content-box {
         width: 200px;
         height: 100px;
         padding: 20px;
         border: 5px solid black;
         box-sizing: content-box;
         background-color: lightblue;
      }

      .border-box {
         width: 200px;
         height: 100px;
         padding: 20px;
         border: 5px solid black;
         box-sizing: border-box;
         background-color: lightcoral;
      }
   </style>
</head>
<body>
   <h2>Ejemplo de box-sizing</h2>
   <div class="content-box">content-box (tamaño más grande)</div>
   <div class="border-box">border-box (tamaño exacto)</div>
</body>
</html>
```

Observación:

- El primer `div` (azul) es más grande porque usa `content-box`.
- El segundo `div` (rojo) respeta el tamaño exacto porque usa `border-box`.

Buenas prácticas con `box-sizing`

Recomendación: Usa `box-sizing: border-box;` globalmente para mantener el diseño predecible.

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

Esto hace que **todos los elementos** sigan este comportamiento sin necesidad de definirlo uno por uno.

Resumen

Valor	¿Qué hace?
content-box (por defecto)	**NO incluye** padding ni border en el tamaño total (puede desbordarse).
border-box	**Incluye** padding y border en el tamaño total (mantiene el diseño estable).

Resumen de las instrucciones Propiedad	¿Qué hace?
border	Define bordes
background-color	Color de fondo
margin-top	Espacio arriba
text-align	Alineación del texto
margin-bottom	Espacio abajo
flex	Controla el tamaño flexible
display: flex	Activa flexbox
text-decoration	Estilos de texto
font-size	Tamaño del texto
flex-direction	Dirección de flexbox
min-height	Altura mínima
box-sizing	Cómo se mide el tamaño