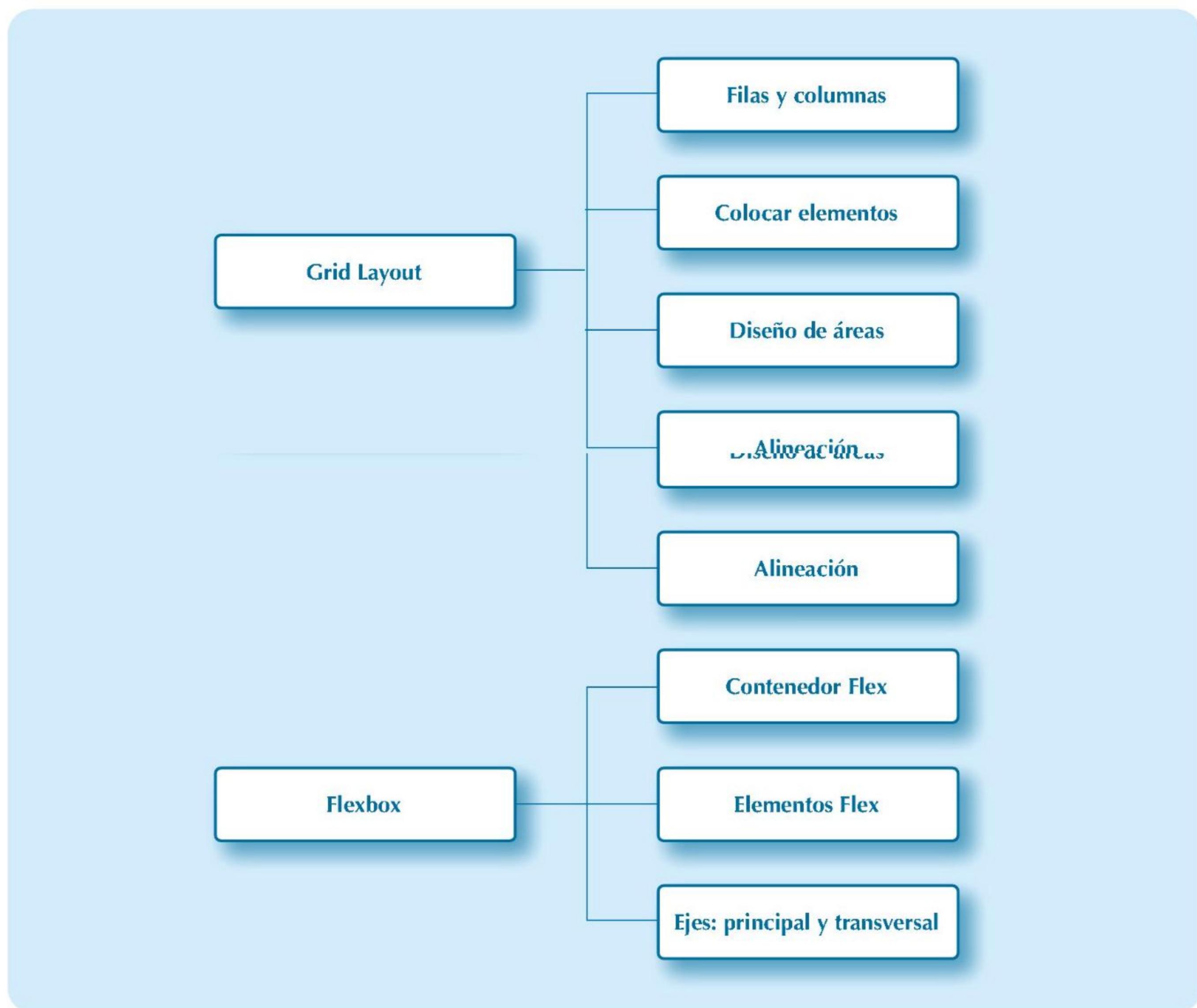


# Grid Layout y Flexbox

## Objetivos

- ✓ Comprender los conceptos básicos de Grid Layout y Flexbox, y su utilidad en el diseño web.
- ✓ Familiarizarse con las propiedades clave de Grid Layout y Flexbox, y cómo se aplican.
- ✓ Ser capaz de crear diseños flexibles y responsivos utilizando Grid Layout y Flexbox.
- ✓ Realizar diseños complejos de páginas web utilizando Grid Layout y Flexbox.

## Mapa conceptual



## Glosario

**Flexbox.** Técnica de diseño unidimensional en CSS que proporciona un método flexible para organizar y distribuir elementos. Es habitual utilizarla para la creación de barras de navegación y galerías de imágenes, entre otros elementos.

**Grid Area.** Espacio rectangular formado por un conjunto de celdas adyacentes en la cuadrícula. Puede contener uno o más elementos de la cuadrícula, y se define mediante la especificación de las líneas de inicio y fin, tanto para las filas como para las columnas.

**Grid Cell.** Área rectangular formada por la intersección de una fila y una columna en la cuadrícula. Cada elemento de la cuadrícula reside en una celda específica.

**Grid Container.** Elemento principal al que se aplica la propiedad `display: grid`. Permite definir el contexto de la cuadrícula, y contiene los elementos secundarios que forman parte de la disposición en la cuadrícula.

**Grid Item.** Elementos secundarios del contenedor de la cuadrícula. Se sitúan dentro de las celdas de la cuadrícula, y pueden ocupar una o más celdas, según la configuración.

**Grid Layout.** Es una técnica en CSS que permite crear diseños de página mediante la estructuración del contenido en filas y columnas, con un sistema bidimensional, lo que facilita la elaboración de diseños estructurados y responsivos.

**Responsivo.** En cuanto al diseño, se refiere a la capacidad de un sitio web o una aplicación para adaptarse y presentarse de manera óptima en una variedad de dispositivos y tamaños de pantalla, actuando de forma dinámica en función del tamaño de la pantalla del dispositivo en el que se visualice.

## 8.1. Introducción

El diseño de páginas con Grid Layout es una técnica en CSS que permite crear diseños de páginas complejos y flexibles, mediante la estructuración del contenido en filas y columnas, lo que facilita la confección de diseños estructurados y responsivos. Este tipo de sistema de organización se caracteriza por definir una cuadrícula bidimensional.

El sistema de rejilla Grid Layout presenta algunas características clave, que es conveniente conocer (cuadro 8.1).

**CUADRO 8.1**  
**Características Grid Layout**

<b>Cuadrícula bidimensional</b>	Se trata de un sistema bidimensional, lo que permite organizar tanto filas como columnas (a diferencia de Flexbox, que se verá más adelante)
<b>Columnas y filas</b>	El sistema Grid va a permitir definir tanto el número de filas como el de columnas que contenga la cuadrícula
<b>Posicionamiento</b>	El sistema Grid hace posible definir la ubicación exacta de los elementos en la cuadrícula
<b>Alineación</b>	Incorpora propiedades para la alineación de los elementos en vertical y en horizontal
<b>Cuadrículas en cuadrículas</b>	Permite introducir cuadrículas dentro de otros elementos de cuadrícula

## 8.2. Sistema Grid Layout

Para la creación del sistema de cuadrícula, en primer lugar habrá que definir en el fichero CSS un contenedor, que será el elemento padre que contendrá el resto de los elementos de la cuadrícula.

Este elemento deberá incluir `display:grid`. Esta propiedad establece que el elemento con la clase `.contenedor` utilizará un sistema de cuadrícula (`grid`) para organizar sus elementos secundarios, lo que implica que sus elementos hijos pueden ser colocados en filas y columnas.

```
.contenedor {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr; /* Define 3 columnas de tamaño igual */
    grid-template-rows: 100px 200px; /* Define 2 filas con alturas específicas */
}
```



#### TOMA NOTA

Para activar el sistema de cuadrícula, utiliza la propiedad `display: grid`. Recuerda que, en primer lugar, debes aplicar esta regla al elemento principal que deseas que actúe como contenedor de la cuadrícula. Cuando se establece `display: grid` en un elemento, se está indicando al navegador que el contenido de ese elemento se organizará en una cuadrícula bidimensional.

Una vez que el contenedor principal tiene `display: grid`, es posible definir cómo se distribuirán y alinearán los elementos hijos dentro de esa cuadrícula, utilizando el resto de las propiedades (`grid-template-columns`, `grid-template-rows`, `grid-gap`, `justify-items`, `align-items`).

### 8.2.1. Filas y columnas

El sistema de cuadrícula Grid permite definir de forma precisa la estructura y la organización de los elementos. Para esto, es necesario fijar tanto el número de columnas como el de filas. Se utilizarán las propiedades `grid-template-columns` y `grid-template-rows`.

- `grid-template-columns`: define la distribución de columnas en la cuadrícula. Es habitual utilizar la unidad “fr,” lo que significa “1 fracción” o “un espacio igual”. Es decir, en caso de usarse `1fr 1fr 1fr`, se estarán creando tres columnas que tendrán el mismo ancho. Los valores se separan mediante un espacio.

```
.contenedor {
    grid-template-columns: 1fr 1fr 2fr;
}
```

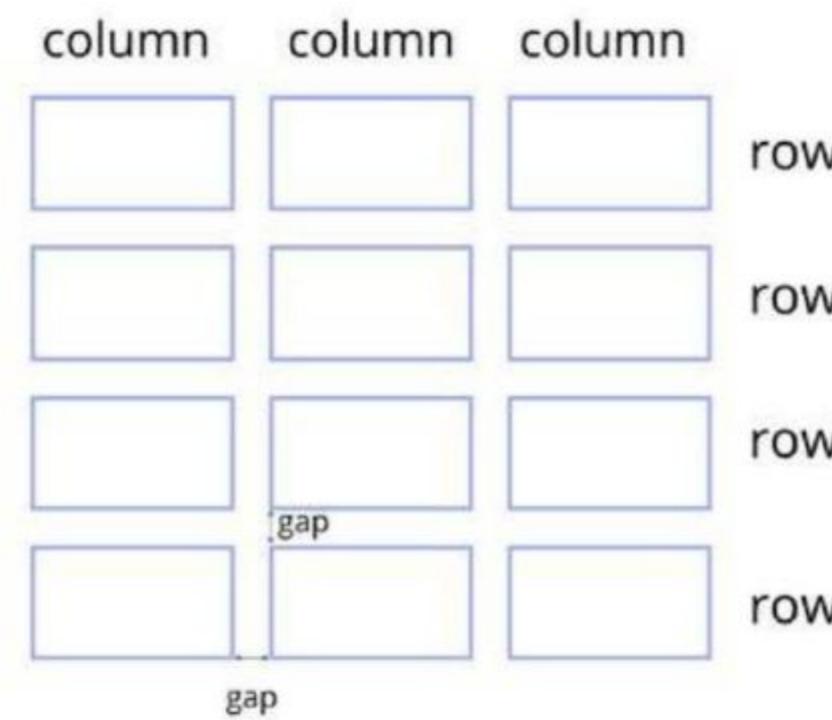
- `grid-template-rows`: define la altura y el número de las filas. Por ejemplo, en el caso de `grid-template-rows: 100px 200px`, la primera fila tendrá una altura de 100 px, y la segunda fila, de 200 px. Los valores se separan mediante un espacio.

```
.contenedor {
    grid-template-rows: 100px 200px;
}
```

Además de estas propiedades básicas, se incluyen otras para fijar el espacio que debe quedar entre los elementos de la cuadrícula y otras para indicar en qué posición exacta se va a colocar un elemento (figura 8.1).

- grid-gap: esta propiedad se utilizará para definir el espacio entre las filas y columnas de una cuadrícula. Los valores que recibe esta propiedad podrán ser en píxeles, porcentajes, etc.

```
.contenedor {
    grid-gap: 10px;
}
```



**Figura 8.1**  
Diagrama Grid Layout.

#### RECUERDA

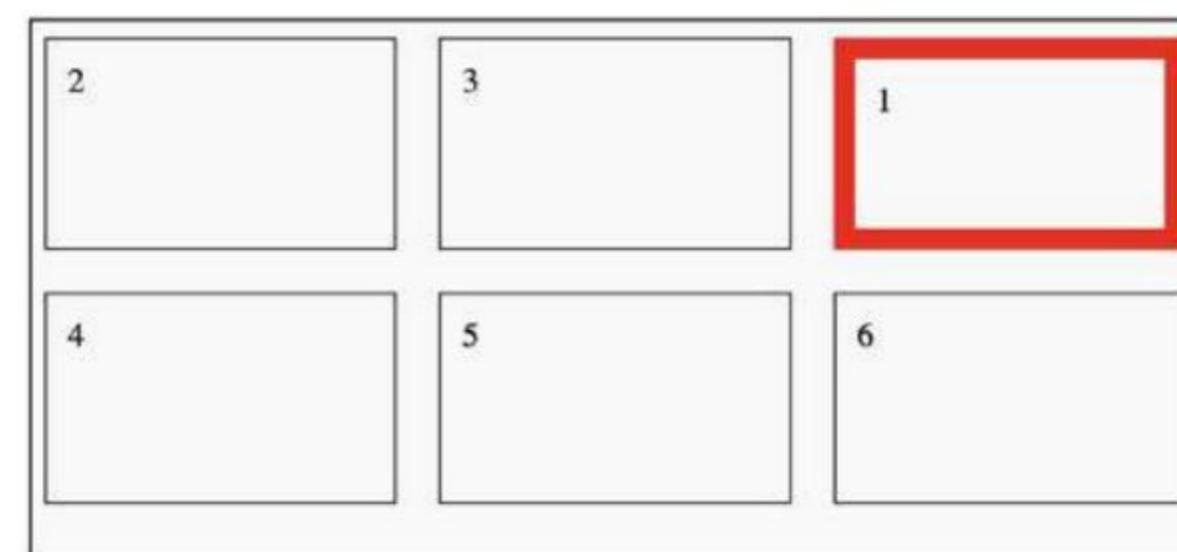
- ✓ La unidad "fr" se utiliza en Grid Layout para distribuir el espacio disponible en fracciones, organizando de esta forma el espacio de una cuadrícula entre columnas o filas, de manera proporcional.

- grid-column y grid-row: permiten especificar en qué columnas y filas debe estar ubicado un elemento. Puedes usar números para señalar la línea, *span* para que se extienda varias líneas, y más.

```
.elemento {
    grid-column: 2 / span 2;
    grid-row: 1 / 3;
}
```

### 8.2.2. Práctica guiada. Colocando elementos en un Grid Layout

Se modela una cuadrícula formado por dos filas de 100 px y tres columnas de igual tamaño (se distribuyen de forma proporcional en el espacio disponible, 1fr 1fr 1fr). Hay una separación entre los elementos de la cuadrícula de 20 px. (figura 8.2).



**Figura 8.2**  
Diagrama de ejemplo  
de Grid Layout

El elemento 1 se coloca en la columna 3 y en la fila 1. De esta forma se reestructuran el resto de los elementos, colocándose el 2 en la primera posición (que ha dejado libre el elemento 1 al situarse en la nueva posición).

Para definir el sistema de cuadrícula sobre el contenedor, se utiliza la propiedad `display: grid`, así como las dimensiones y el espaciado de los elementos de dicha cuadrícula. Después, se define la posición exacta del elemento identificado como `#a1` con `grid-column` y `grid-row` (figura 8.3).

```
.contenedor {
    display: grid;
    grid-template-columns: 1fr 1fr 1fr;
    grid-template-rows: 100px 100px;
    grid-gap: 20px;
}

.item {
    border: 1px solid black;
    padding: 10px;
}

#a1{
    border: 10px solid red;
    padding: 10px;
    grid-column: 3;
    grid-row: 1;
}
```

**Figura 8.3**  
Sistema de cuadrícula sobre el contenedor.

### Actividad propuesta 8.1



Implementa un sistema de cuadrícula como el de la práctica guiada, pero añadiendo una nueva columna que ocupe la mitad del espacio disponible en el contenedor.

El elemento 1 se colocará en la fila inferior y en la primera columna.

Para facilitar la ubicación de los elementos en la cuadrícula, se utiliza la propiedad `display: template-areas`, que permite colocar áreas, previamente nombradas, en la cuadrícula. Esta propiedad recibe los nombres usando un tipo String.

Los nombre utilizados en la propiedad `grid-template-areas` se deben haber definido previamente en cada una de las áreas que colocar, usando `grid-area` seguido del nombre que se vaya a asignar a cada área (figura 8.4).

Si se necesita que un área se extienda por más de una columna o una fila, se deberá repetir el nombre del área tantas veces como huecos se quieran ocupar. En la siguiente práctica guiada, se ilustra este caso.

```
.container {
    display: grid;
    grid-template-areas:
        "header header header"
        "main content sidebar"
        "footer footer footer";
    grid-template-rows: 100px 1fr 100px;
    grid-template-columns: 1fr 2fr 1fr;
    gap: 10px;
    height: 100vh;
}

.header {
    grid-area: header;
    background-color: lightblue;
}

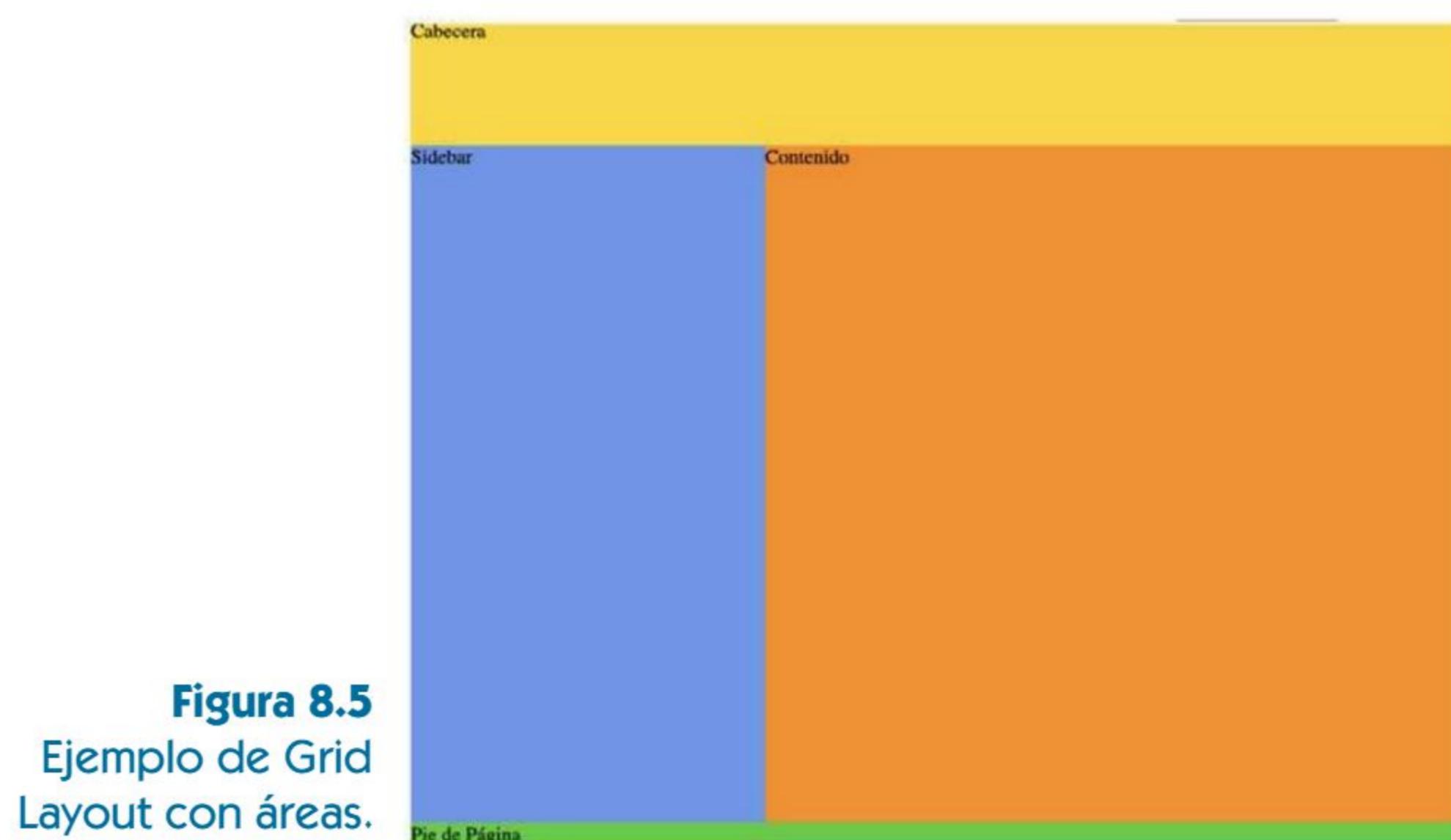
.main {
    grid-area: main;
    background-color: lightgreen;
}
```

**Figura 8.4**  
Código de distribución de áreas con `grid-templates.areas`.

### 8.2.3. Práctica guiada. Diseño de áreas en Grid Layout

Se modela un sistema de cuadrícula como el de la imagen, en el que la cabecera ocupará todo el ancho de la cuadrícula, el área de contenido tendrá un tamaño del doble que la de sidebar, y el pie de página solo ocupará el espacio que requiera su contenido.

En el elemento contenedor se definirán la distribución de los elementos de la cuadrícula (filas y columnas), así como la ubicación concreta de cada una de las áreas. En este caso, la cabecera ocupará toda la fila compuesta por dos columnas (“cabecera cabecera”), y en la zona central se situarán el sidebar y el contenido, ocupando la segunda columna el doble que la primera (1fr 2fr) (figura 8.5).



Para poder utilizar la distribución de áreas con `grid-template-columns`, en cada uno de los elementos se ha definido previamente el nombre asignado (por ejemplo, `grid-area:cabecera`) (figura 8.6).

```
.contenedor {
    display: grid;
    grid-template-columns: 1fr 2fr; /* 1 columna ancha, 2 columnas más anchas */
    grid-template-rows: 100px 1fr; /* 1 fila fija, 1 fila flexible */
    grid-template-areas:
        "cabecera cabecera"
        "sidebar contenido"
        "pie pie";
    height: 100vh; /* Establece la altura del contenedor al 100% del viewport height */
}

.cabecera {
    grid-area: cabecera; /* Asigna esta div al área llamada "cabecera" */
    background-color: #FFD700; /* Amarillo */
}

.sidebar {
    grid-area: sidebar; /* Asigna esta div al área llamada "sidebar" */
    background-color: #6495ED; /* Azul */
}

.contenido {
    grid-area: contenido; /* Asigna esta div al área llamada "contenido" */
    background-color: #FF8C00; /* Naranja */
}

.pie {
    grid-area: pie; /* Asigna esta div al área llamada "pie" */
    background-color: #32CD32; /* Verde */
}
```

**Figura 8.6**  
Código de desarrollo del diseño de la figura 8.5.

## Actividad propuesta 8.2



Implementa un sistema de cuadrícula como el de la práctica guiada “Diseño de áreas en Grid Layout”. Modifica los colores, y añade una columna nueva y un apartado de sugerencias para situarlo de forma paralela al sidebar y al contenido. Las columnas deberán aumentar el tamaño de manera progresiva desde la izquierda hasta la derecha.

### 8.2.4. Alineación

La alineación en Grid Layout permite definir cómo los elementos de una cuadrícula van a ser colocados y distribuidos con respecto a los ejes horizontal y vertical. Existen varias propiedades que permiten controlar la alineación de los elementos en la cuadrícula: `justify-items`, `align-items`, `justify-content` y `align-content`.

- `justify-items` y `align-items`: estas propiedades permiten alinear elementos dentro de las celdas de la cuadrícula. Pueden tomar diferentes valores: `start`, `end`, `center`, `stretch`... En el siguiente ejemplo, los elementos quedarían centrados en el hueco de la cuadrícula donde están colocados, con respecto tanto al eje horizontal como al vertical.

```
.contenedor {
    justify-items: center;
    align-items: center;
}
```

- `justify-content` y `align-content`: estas propiedades permiten alinear el contenido de la cuadrícula con respecto al contenedor, tanto en el eje horizontal (`justify-content`) como en el eje vertical (`align-content`).
  - `justify-content`: define la alineación del contenido de la cuadrícula con respecto al eje horizontal (cuadro 8.2).

#### CUADRO 8.2

#### Opciones de `justify-content`

<b>start</b>	El contenido se alinea al inicio del contenedor
<b>end</b>	El contenido se alinea al final del contenedor
<b>center</b>	El contenido se alinea en el centro del contenedor
<b>space-between</b>	Los elementos se distribuyen manteniendo un espacio uniforme entre ellos
<b>space-around</b>	Los elementos se distribuyen con un espacio alrededor uniforme. La diferencia entre esta opción y la anterior es que aquí se tiene en cuenta que la distancia que quedaría entre el primer elemento y el margen izquierdo, sumando la distancia entre el último elemento y el margen derecho, es igual que el espacio que queda entre el resto de los elementos

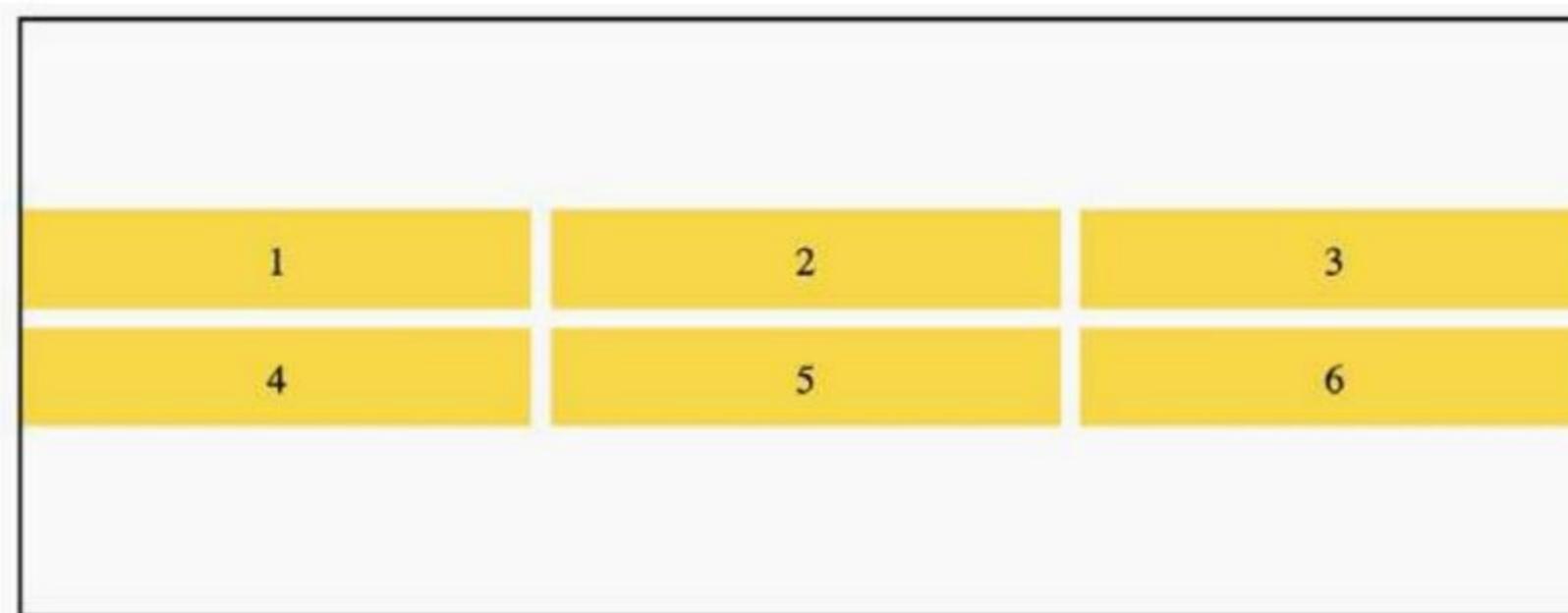
- `align-content`: define la alineación del contenido de la cuadrícula con respecto al eje vertical (cuadro 8.3).

**CUADRO 8.3**  
Opciones de *align-content*

<b>start</b>	El contenido se alinea al inicio del contenedor
<b>end</b>	El contenido se alinea al final del contenedor
<b>center</b>	El contenido se alinea en el centro del contenedor
<b>stretch</b>	El contenido se “estira” hasta llenar todo el contenedor de forma vertical
<b>space-between</b>	Los elementos se distribuyen manteniendo un espacio uniforme entre ellos.
<b>space-around</b>	Los elementos se distribuyen con un espacio alrededor uniforme. La diferencia entre esta opción y la anterior es que aquí se tiene en cuenta que la distancia que quedaría entre el primer elemento y el margen izquierdo, sumando la distancia entre el último elemento y el margen derecho, es igual que el espacio entre el resto de los elementos

### 8.2.5. Práctica guiada. Alineación de elementos en Grid Layout

Se modela un sistema de cuadrícula como el de la imagen, en el que se colocan seis elementos distribuidos en dos filas y tres columnas. Se define el mismo tamaño para las columnas (`1fr 1fr 1fr`) y también para las filas (`50px 50px`). El espacio entre los elementos será de 10 px (definido en `grid-gap`) (figura 8.7).



**Figura 8.7**  
Ejemplo de Grid Layout.

En cuanto a la alineación horizontal, se utilizará `justify-content: space-around`. Es decir, la distancia que existe entre 1 y 2 (o entre 2 y 3, 4 y 5, y 5 y 6) será igual a la suma de la distancia que queda a la izquierda del 1 y a la derecha del 3 (se ha coloreado el borde con dos colores para que se observe esta suma) (figura 8.8).

```

.contenedor {
  display: grid;
  grid-template-columns: 1fr 1fr 1fr;
  grid-template-rows: 50px 50px;
  grid-gap: 10px;
  height: 300px;
  border: 2px solid black;
  justify-content: space-around;
  align-content: center;
}

.item {
  background-color: #FFD700; /* Amarillo */
  text-align: center;
  padding: 15px;
  font-size: 20px;
}

```

**Figura 8.8**  
Código desarrollo del diseño de la figura 8.7.

Para alinear el contenido en la zona central –es decir, centrado con respecto al eje vertical– se utilizará `align-content: center`.

### Actividad propuesta 8.3



Implementa un sistema de cuadrícula como el anterior, pero ahora coloca 10 elementos, distribuidos en tres filas y dos columnas. Las columnas tendrán mismo tamaño y las filas también.

#### RECUERDA

- ✓ Para alinear el contenido de texto de cada elemento utiliza `text-align: center;`

## 8.3. Flexbox

Flexbox (*Flexible Box Layout*) es un método de diseño unidimensional en CSS que resulta flexible para organizar y distribuir elementos. A diferencia de Grid Layout, que permite la organización bidimensional, Flexbox se centra en la distribución unidimensional. Por esta razón, es habitual utilizarlo para la creación de barras de navegación y galerías de imágenes, entre otros componentes, puesto que facilita la alineación vertical y horizontal de elementos. Algunas de las ventajas del uso de Flexbox son las que siguen:

- *Simplifica la maquetación*: elimina la necesidad de flotar elementos.
- *Control del espaciado*: incorpora propiedades que permiten un control preciso del espacio entre los elementos, tanto en el nivel horizontal como en el vertical, lo que facilita la creación de diseños limpios y agradables.
- *Responsivo*: está especialmente destinado a la realización de diseños que se adapten fácilmente a diferentes tamaños de pantalla y dispositivos, mejorando la experiencia del usuario.
- *Ordenación de elementos*: permite cambiar el orden visual de los elementos sin variar el orden en el HTML, lo que es útil para presentar la información de manera más efectiva.

#### RECUERDA

- ✓ Usa Flexbox para diseño unidimensional y para organizar elementos en filas o columnas.
- ✓ Usa Grid Layout para diseño bidimensional, es decir, para organizar elementos en filas y columnas al mismo tiempo. Es perfecto para diseños más complejos y estructurados.

A menudo, la mejor estrategia es combinar Flexbox y Grid Layout en la misma página. Flexbox puede ser usado en el nivel de los elementos individuales dentro de un contenedor Grid.

### 8.3.1. Contenedor Flex y elementos Flex

Un contenedor Flex es un elemento HTML que tiene sus hijos directos configurados para usar Flexbox. Al habilitar un contenedor como “contenedor Flex”, este se convierte en un contexto de diseño flexible sobre el que distribuir el resto de los elementos que dependen de él.

Este elemento deberá incluir `display:flex`. Esta propiedad establece que el elemento con la clase `.contenedor` utilizará un sistema Flexbox para organizar sus elementos secundarios, lo que implica que sus elementos hijos pueden ser colocados en filas o columnas.

```
.contenedor-flex {
    display: flex;
}
```

Los elementos que son hijos directos de un contenedor Flex serán los elementos flexibles que participen en el diseño.

```
.elemento-flex {
    // Propiedades
}
```

Las propiedades de los elementos Flexbox serán definidas mediante la propiedad abreviada `flex` o las propiedades individuales `flex-grow`, `flex-shrink` y `flex-basis` (cuadro 8.4).

```
.item {
    flex: <flex-grow> <flex-shrink> <flex-basis>;
}
```

**CUADRO 8.4**  
Propiedades `flex`

<b>flex-grow</b>	Define el espacio adicional que debe tomar el elemento en relación con los otros elementos dentro del mismo contenedor. Un valor de 1 significa que tomará todo el espacio adicional disponible
<b>flex-shrink</b>	Define la capacidad del elemento para reducir su tamaño, en caso de ser necesario para evitar el desbordamiento del contenedor
<b>flex-basis</b>	Define el tamaño inicial del elemento antes de que se distribuya el espacio adicional o se apliquen los factores de reducción

### 8.3.2. Práctica guiada. Contenedor de elementos en Flexbox

Se modela un sistema Flexbox como el de la imagen, en el que se colocan tres elementos distribuidos en una fila, de tal forma que estos se expandan y ocupen todo el espacio disponible (`flex-grow:1`). Si el tamaño del contenedor disminuye, estos elementos también deberán disminuir en tamaño (`flex-shrink:1`) (figura 8.9).



**Figura 8.9**  
Ejemplo de distribución de elementos Flexbox.

En primer lugar, se creará el contenedor y se activará la opción de Flexbox (figura 8.10).

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

**Figura 8.10**  
Activación de Flexbox.

En cada elemento, en este caso en el que comparten la etiqueta de clase `item`, se aplican las características especificadas (figura 8.11).

```
.item {  
    flex-grow: 1; /* Ocupa todo el espacio disponible */  
    flex-shrink: 1; /* Puede encogerse si es necesario */  
    flex-basis: 200px; /* Tamaño inicial */  
    background-color: #f0f0f0;  
    border: 1px solid #ccc;  
    margin: 5px;  
}  
.item:nth-child(2) {  
    flex-basis: 150px; /* Este elemento tendrá un tamaño inicial mayor */  
}
```

**Figura 8.11**  
Código de desarrollo  
diseño de la figura 8.9.

En el elemento segundo de los que se colocan, el tamaño en condiciones iniciales será de 150 px, mientras que en el resto será de 200 px.

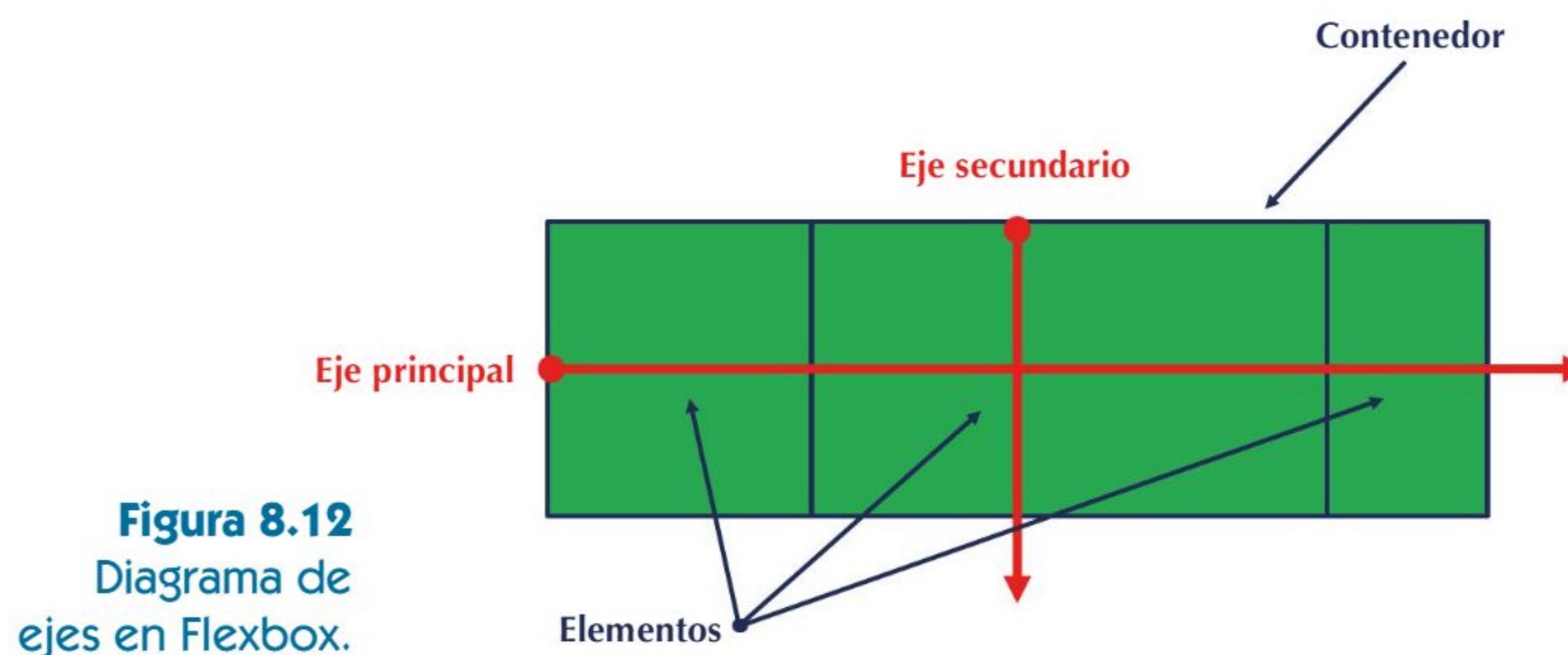
#### Actividad propuesta 8.4



Implementa un sitio web como el anterior, pero añadiendo dos nuevos elementos y modificando el tamaño inicial de los mismos. Además, estos no deben ajustarse al tamaño adicional, como los restantes del contenedor.

### 8.3.3. Ejes: principal y transversal

Los ejes son los elementos fundamentales en Flexbox. Se refieren a las direcciones sobre las cuales se distribuirán y alinearán los elementos flexibles. Hay dos ejes esenciales: el eje principal (*main axis*) y el eje secundario (*cross axis*). Estos dos ejes son importantes para saber cómo se comportan y distribuyen los elementos flexibles dentro de un contenedor Flex (figura 8.12).



**Figura 8.12**  
Diagrama de  
ejes en Flexbox.

El eje principal es la dirección en la que se colocan los elementos flexibles dentro del contenedor. Para indicar cuál de los dos ejes es el principal se utiliza la propiedad `flex-direction`, seguida de `row` para el caso de la dirección horizontal (como una fila) o `column` para el caso vertical (como una columna). Si no se define ninguno, por defecto será horizontal.

```
.contenedor-flex {
    flex-direction: row; /* Eje principal horizontal (por defecto) */
    flex-direction: column; /* Eje principal vertical */
}
```

El transversal será, en cada caso, el eje perpendicular al eje principal. Es decir, si el eje principal es horizontal, el transversal es vertical, y viceversa.

### A) Dirección

En función de los ejes definidos, se aplican las propiedades de dirección o alineación para los elementos que dependen del contenedor flexible Flexbox. En este apartado, se describen las propiedades fundamentales sobre la dirección.

- `direction`: define la dirección de escritura sobre el eje principal, es decir, la dirección en la que se escribe el texto. Para definirla, se utiliza la propiedad `direction`. Esta dirección será de izquierda a derecha (`ltr`) o de derecha a izquierda (`rtl`). Si no se define ninguna dirección, se toma por defecto la dirección de izquierda a derecha.

```
.contenedor-flex {
    direction: ltr; /* De izquierda a derecha (por defecto) */
    direction: rtl; /* De derecha a izquierda */
}
```

- `flex-direction`: define la dirección en la que se colocan los elementos dentro del contenedor flexible. Puede tomar los siguientes valores: `row` (fila en sentido izquierda-derecha), `row-reverse` (fila en sentido inverso: derecha-izquierda), `column` (columna de arriba abajo) y `column-reverse` (columna sentido inverso, de abajo arriba).

```
.contenedor-flex {
  flex-direction: row | row-reverse | column | column-reverse;
}
```

- **flex-wrap:** esta propiedad permite definir si los elementos se van a distribuir en varias líneas, en caso de que el espacio sea insuficiente. Puede tomar los siguientes valores: `wrap` (se podrán distribuir en varias líneas), `no-wrap` (no se podrán distribuir en varias líneas) y `wrap-reverse` (se distribuirán en varias líneas, pero en sentido inverso).

```
.contenedor-flex {
  flex-wrap: wrap | nowrap | wrap-reverse;
}
```

- **flex-flow:** Se trata de una propiedad abreviada que combina `flex-direction` y `flex-wrap`, por lo que estas propiedades toman los valores descritos anteriormente.

```
.contenedor-flex {
  flex-flow: row wrap;
}
```

## B) Alineación

La alineación permitirá definir cómo se posicionan y ajustan los elementos en el interior de un contenedor flexible, tanto en el eje principal como en el secundario. Se utilizan varias propiedades para controlar la alineación de los elementos:

- **justify-content:** permite definir la alineación de los elementos a lo largo del eje principal del contenedor (cuadro 8.5).

```
.contenedor-flex {
  justify-content: opciones;
}
```

**CUADRO 8.5**  
Opciones de *justify-content*

<b>flex-start</b>	Los elementos se alinean hacia el inicio del contenedor. El primer elemento estará en la esquina inicial, y los demás seguirán la dirección del eje principal
<b>flex-end</b>	Los elementos se alinean hacia el final del contenedor. El último elemento estará en la esquina final, y los demás se colocarán antes
<b>center</b>	Los elementos se centran dentro del contenedor a lo largo del eje principal
<b>space-between</b>	Los elementos se distribuyen uniformemente a lo largo del eje principal. El primer elemento se coloca al inicio; el último, al final, y el espacio restante se distribuye igualmente entre los elementos
<b>space-around</b>	Similar a <code>space-between</code> , pero con espacio adicional antes del primer elemento y después del último elemento

- `align-items`: permite definir la alineación de los elementos a lo largo del eje transversal del contenedor (eje perpendicular al principal) (cuadro 8.6).

```
.contenedor-flex {
  align-items: opciones;
}
```

**CUADRO 8.6**  
Opciones de `align-items`

<b>flex-start</b>	Los elementos se alinean hacia el inicio del contenedor
<b>flex-end</b>	Los elementos se alinean hacia el final del contenedor
<b>center</b>	Los elementos se centran dentro del contenedor a lo largo del eje transversal
<b>baseline</b>	Los elementos se alinean en las líneas base del eje transversal
<b>stretch</b>	Los elementos se estiran para llenar todo el espacio disponible en el eje transversal

- `align-content`: define cómo se distribuye el espacio adicional en el eje transversal si hay varias líneas de elementos. Esta propiedad se aplica cuando hay `wrap` y afecta al espacio entre las filas (cuadro 8.7).

```
.contenedor-flex {
  align-content: opciones;
}
```

**CUADRO 8.7**  
Opciones de `align-content`

<b>flex-start</b>	Las filas se alinean hacia el inicio del contenedor
<b>flex-end</b>	Las filas se alinean hacia el final del contenedor
<b>center</b>	Las filas se centran verticalmente dentro del contenedor
<b>space-between</b>	Las filas se distribuyen uniformemente a lo largo del eje transversal
<b>space-around</b>	Las filas se distribuyen uniformemente a lo largo del eje transversal, con espacio adicional alrededor de cada fila
<b>stretch</b>	Las filas se estiran para llenar todo el espacio disponible en el eje transversal

### C) Order

Los elementos organizados en un contenedor Flexbox podrán cambiar su orden de visualización. Para ello se utiliza la propiedad `order`. De forma predeterminada, todos los elementos tienen un valor de 0, lo que significa que se mostrarán en el orden en el que aparecen en el documento HTML. A través de `order`, se podrá modificar el orden visual de los elementos sin cambiar la estructura del DOM.

La propiedad `order` recibe como valor la posición en la que se va a colocar el elemento con respecto al resto. Puede usar valores decimales (figura 8.13). Si varios elementos tienen el mismo valor de `order`, se mostrarán en el mismo orden en el que aparecen en la estructura HTML.

```
elemento {  
    order: <número>;  
}
```



#### TOMA NOTA

Si se asigna a `order` un valor negativo, este elemento se colocará al principio del orden visual, incluso antes que los elementos con valor `order` de 0.

```
.contenedor-flex {  
    display: flex;  
}  
.elemento-1 {  
    order: 2; /* Se mostraría en la cuarta posición */  
}  
.elemento-2 {  
    order: 1; /* Se mostraría en la tercera posición */  
}  
.elemento-3 {  
    order: 3; /* Se mostraría en la quinta posición */  
}  
.elemento-4 {  
    order: -1; /* Se mostraría en la primera posición */  
}  
.elemento-5 {  
    /* Por defecto, el valor de order es 0 */  
    /* Se mostraría en la segunda posición */  
}
```

**Figura 8.13**  
Código distribución de elementos en Flexbox.

En el ejemplo, bajo un mismo contenedor se han definido cinco elementos. El primer elemento será aquel que tenga en `order` un número más abajo; en este caso, -1. El siguiente elemento será el que tenga 0, es decir, el que utilice el valor de `order` por defecto. Los demás se colocarán siguiendo un orden ascendente.

#### Actividad propuesta 8.5



Reescribe el código del ejemplo con `order` para que los elementos se coloquen en orden contrario al que se muestra.

## Resumen

- Para la creación del sistema de cuadrícula, en el fichero CSS será necesario definir un contenedor, que será el elemento padre y que contendrá el resto de los elementos de la cuadrícula. Este elemento deberá incluir `display:grid`. El sistema de cuadrícula Grid permite definir de forma precisa la estructura de los elementos que la componen. Para esto es necesario definir tanto el número de columnas como el de filas. Se utilizarán las propiedades `grid-template-columns` y `grid-template-rows`.
- Para definir la ubicación de los elementos en el sistema de cuadrícula, se utiliza la propiedad `display: template-areas`. Los nombre utilizados en la propiedad `grid-template-areas` se deben haber definido previamente en cada una de las áreas que colocar, utilizando `grid-area` seguido del nombre que se vaya a asignar a cada área.
- Existen varias propiedades que permiten controlar la alineación de los elementos en la cuadrícula: `justify-items`, `align-items`, `justify-content` y `align-content`.
- Flexbox es un método de diseño unidimensional en CSS que proporciona flexibilidad para organizar y distribuir elementos. Flexbox se centra en la distribución unidimensional de elementos.
- Un contenedor Flex es un elemento HTML que tiene sus hijos directos configurados para usar Flexbox. Este elemento deberá incluir `display:flex`.
- Los ejes son los elementos fundamentales en Flexbox. Se refieren a las direcciones sobre las cuales se distribuirán y alinearán los elementos flexibles. Hay dos ejes esenciales: el eje principal y el eje transversal.
- El eje principal es la dirección en la que se colocan los elementos flexibles dentro del contenedor. Para indicar cuál de los dos ejes es el principal se utiliza la propiedad `flex-direction`, seguida de `row` para la dirección horizontal o `column` para la vertical. El transversal será, en cada caso, el eje perpendicular al principal.
- En función de los ejes definidos, se aplican las propiedades de dirección o alineación: `direction`, `flex-direction`, `flex-wrap`, `flex-flow`, `justify-content` o `align-items`.

## Ejercicios propuestos



1. Diseña un sitio web utilizando Grid Layout en CSS para el siguiente código HTML. Crea tres cajas de igual tamaño y color de fondo azul, alineadas horizontalmente y con un espacio uniforme entre ellas. El texto dentro de cada caja debe estar centrado tanto horizontal como verticalmente.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <metaname="viewport"content="width=device-width, initial-scale=1.0">
    <title>Grid Layout Exercise</title>
    <link rel="stylesheet" href="styles.css">
</head>
```

```
<body>
  <div class="container">
    <div class="box">Box 1</div>
    <div class="box">Box 2</div>
    <div class="box">Box 3</div>
  </div>
</body>
</html>
```

2. Utilizando el siguiente código HTML como referencia, diseña el CSS correspondiente para que, utilizando Flexbox, las cajas tengan un ancho y un alto fijos, un color de fondo azul, y estén alineadas horizontalmente, con un espacio uniforme entre ellas. Además, el texto dentro de cada caja debe estar centrado tanto horizontal como verticalmente.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Flexbox Exercise</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <div class="box">Box 1</div>
    <div class="box">Box 2</div>
    <div class="box">Box 3</div>
  </div>
</body>
</html>
```

3. Diseña, utilizando tanto Grid Layout como Flexbox, para el siguiente código HTML, cuatro cajas de diferentes tamaños y colores de fondo, distribuidas en dos filas. Las dos primeras deben estar alineadas horizontalmente usando Flexbox, mientras que las dos últimas deben estar distribuidas en una cuadrícula de dos columnas empleando Grid Layout.

```
<!DOCTYPE html>
<html lang="en">
<head>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <title>Grid and Flexbox Exercise</title>
  <link rel="stylesheet" href="styles.css">
</head>
<body>
  <div class="container">
    <div class="flex-box">
      <div class="box1">Box 1</div>
      <div class="box2">Box 2</div>
    </div>
    <div class="grid-box">
      <div class="box3">Box 3</div>
      <div class="box4">Box 4</div>
    </div>
  </div>
</body>
</html>
```

```

</div>
<div class="grid-container">
    <div class="box3">Box 3</div>
    <div class="box4">Box 4</div>
</div>
</div>
</body>
</html>

```

4. Tras realizar los ejercicios anteriores, ¿qué diferencias consideras que existen entre Grid Layout y Flexbox? ¿En qué casos es más adecuado utilizar uno u otro?
5. Con respecto a Grid Layout y FlexBox, contacta a las siguiente cuestiones: ¿Cómo describirías las principales diferencias en la estructura y enfoque de diseño entre CSS Grid Layout y Flexbox? ¿En qué situaciones es más ventajoso utilizar la dirección unidimensional de Flexbox? ¿Y cuándo es preferible la dirección bidimensional de Grid Layout? ¿En qué contextos podría el rendimiento ser un factor determinante para elegir entre Flexbox y Grid Layout?
6. Crea una página que muestre tres contenedores flexibles, cada uno con una configuración diferente de `flex-wrap`. Cada contenedor debe contener nueve elementos en contenedores cuadrados numerados del 1 al 9.

- El primer contenedor debe tener la configuración `flex-wrap: nowrap;`, lo que significa que los elementos no se envolverán y seguirán en la misma línea incluso si no hay suficiente espacio horizontal.
- El segundo contenedor debe tener la configuración `flex-wrap: wrap;`, lo que permitirá que los elementos se envuelvan a la siguiente línea si no hay suficiente espacio horizontal.
- El tercer contenedor debe tener la configuración `flex-wrap: wrap-reverse;`, haciendo que los elementos se envuelvan en orden inverso.

7. Diseña una página web de noticias con Grid Layout. Utiliza CSS Grid Layout para organizar los elementos de la página de manera efectiva.

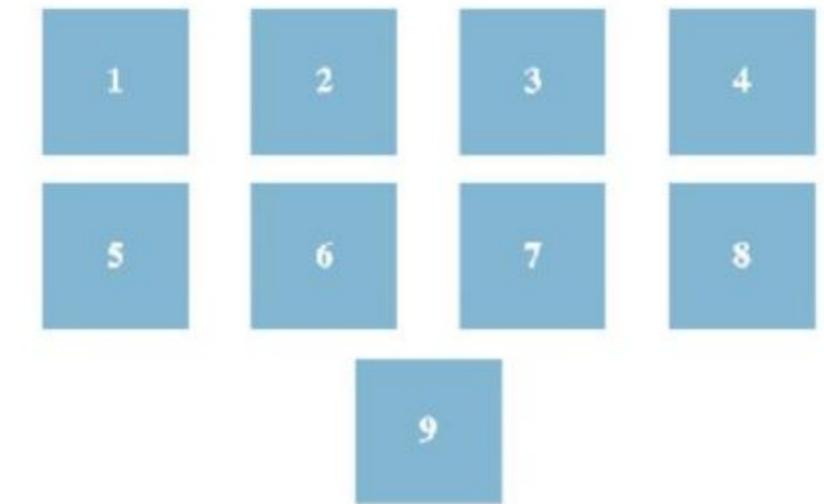
- Crea un documento HTML con la estructura básica de una página web que incluya lo siguiente: un encabezado, una barra de navegación, una sección de noticias principales, una barra lateral con noticias destacadas y un pie de página.
- Utiliza CSS Grid Layout para organizar los elementos en el diseño de la página. Define áreas y asigna elementos a esas áreas según se describe en el enunciado.
- El tamaño de la caja de noticias principales se debe adaptar al tamaño de la pantalla.

#### Ejercicio con Todas las Opciones de Flex-Wrap

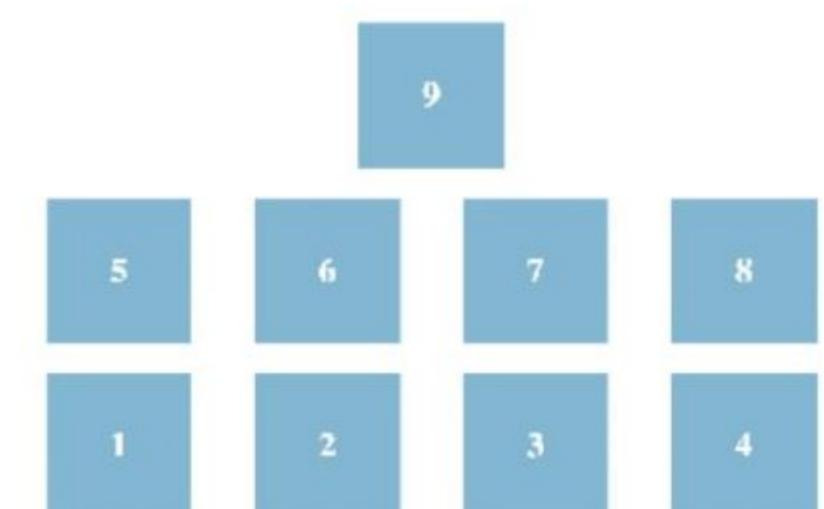
##### No Wrap



##### Wrap



##### Wrap Reverse

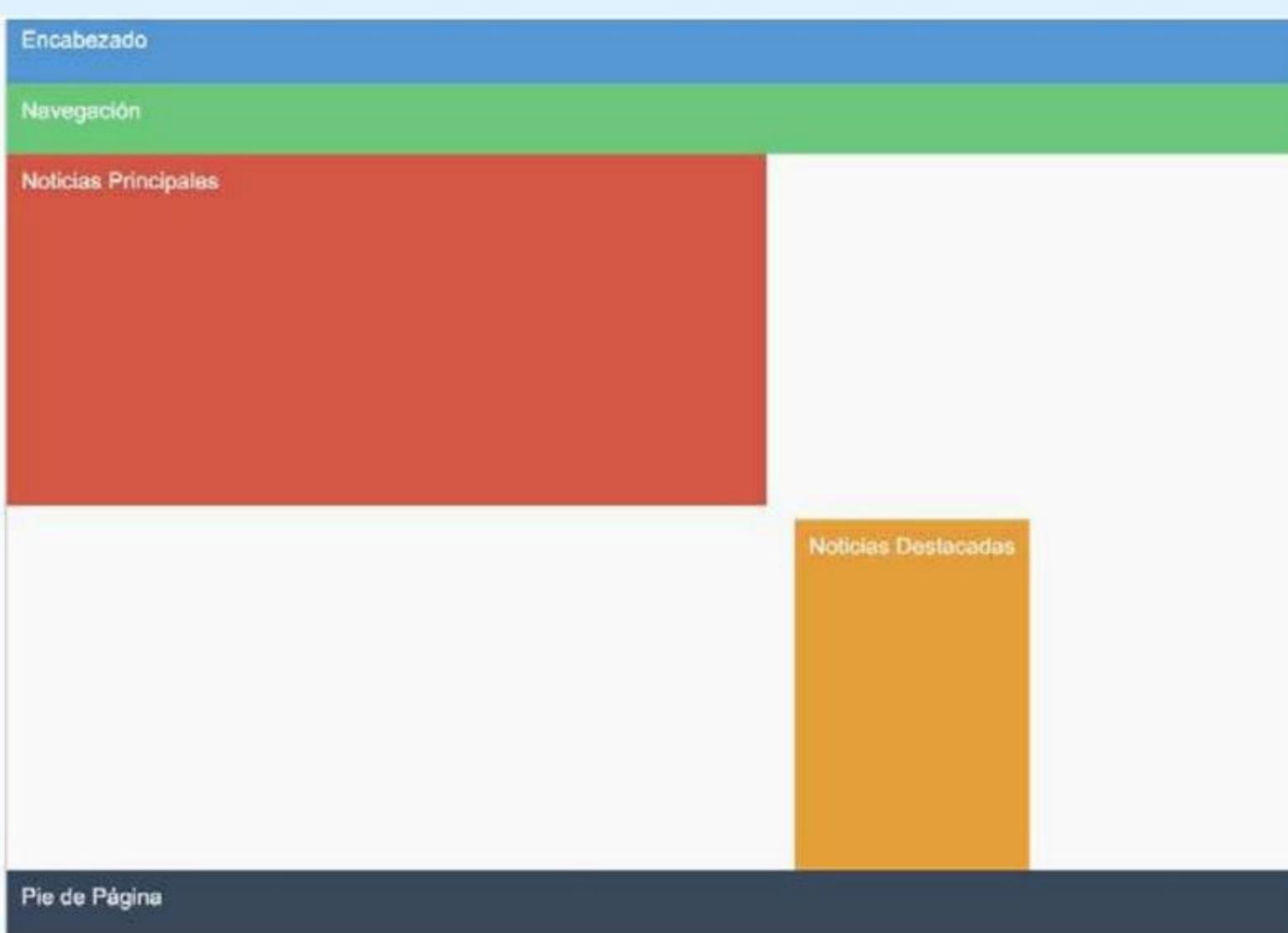


**8.** Diseña un panel de control. Crea un sitio web que contenga un panel de control, con los elementos distribuidos en filas y columnas, utilizando Flexbox.

- Debe haber un encabezado fijo en la parte superior.
- El panel debe tener tres secciones: izquierda, centro y derecha.
- La sección izquierda debe contener un menú vertical con, al menos, cinco elementos.
- La sección central debe contener un gráfico de barras horizontal.
- La sección derecha debe contener un formulario con campos para ingresar datos.



**9.** Crea una página web con galería de imágenes, utilizando la técnica de Flexbox para organizar y distribuir los elementos de manera eficiente. El contenedor principal (contenedor-flex) utiliza Flexbox para organizar los elementos hijos (las imágenes). Cada imagen tiene un tamaño de 200 x 200 px, y entre ellas hay un espacio de 10 píxeles.



- 10.** Crea un sitio web que tenga la estructura anterior. Personalízalo como quieras (elaborando una temática de sitio web que te pueda servir para el proyecto de final de trimestre: elige los colores, la distribución de elementos, etc.). La zona superior está modelada con Flex, mientras que la zona central se modela con Grid.



## ACTIVIDADES DE AUTOEVALUACIÓN

1. ¿Cuál es el propósito de la propiedad `grid-template-areas` en Grid Layout?
  - a) Especificar el número de filas y columnas en una cuadrícula.
  - b) Organizar áreas de la cuadrícula y asignar elementos a esas áreas.
  - c) Establecer el tamaño de las celdas de la cuadrícula.
  - d) Controlar el espaciado entre filas y columnas en la cuadrícula.
2. Si deseas centrar un elemento en un diseño Flexbox horizontal y verticalmente, ¿cuál es la propiedad de CSS que se debe usar en el contenedor?
  - a) `justify-content: center;`
  - b) `align-content: center;`
  - c) `flex-align: center;`
  - d) `align-items: center;`
3. En Grid Layout, ¿qué propiedad se usa para indicar cuánto espacio se asigna a una fila o columna en una cuadrícula?
  - a) `grid-space`
  - b) `grid-gap`
  - c) `grid-row-gap`
  - d) `grid-template-columns`

4. En Flexbox, ¿cuál es el valor predeterminado de la propiedad `flex-direction`?
- a) column
  - b) row
  - c) row-reverse
  - d) column-reverse
5. ¿Qué propiedad de Flexbox se utiliza para distribuir el espacio adicional o faltante entre los elementos flexibles dentro de un contenedor?
- a) `flex-basis`
  - b) `justify-content`
  - c) `align-items`
  - d) `flex-grow`
6. En Grid Layout, ¿cuál es la propiedad que se utiliza para especificar el espacio entre las filas y columnas de una cuadrícula?
- a) `grid-gap`
  - b) `grid-template-gap`
  - c) `grid-row-gap`
  - d) `grid-column-gap`
7. En Flexbox, ¿cuál es la propiedad que se usa para especificar el orden en que los elementos flexibles son colocados dentro de su contenedor?
- a) `flex-order`
  - b) `flex-direction`
  - c) `flex-wrap`
  - d) `order`
8. En Grid Layout, ¿cuál es la propiedad que se utiliza para definir una cuadrícula con cuatro columnas de igual tamaño?
- a) `grid-template-columns: (4, 1fr);`
  - b) `grid-template-columns: 1fr 2fr 3fr 4fr;`
  - c) `grid-template-columns: 1fr 1fr 1fr 1fr;`
  - d) Ninguna de las anteriores.
9. ¿Cuál de las siguientes propiedades se emplea para distribuir el espacio entre elementos en una columna, de manera que haya un espacio igual alrededor de cada elemento?
- a) `justify-content: space-around;`
  - b) `align-content: space-around;`
  - c) `justify-content: space-between;`
  - d) `align-content: space-between;`
10. ¿Cuál de las siguientes afirmaciones es correcta acerca de la propiedad `grid-area` en Grid Layout?
- a) Define el tamaño de una celda en la cuadrícula.
  - b) Asigna elementos a áreas de la cuadrícula definidas por `grid-template-areas`.
  - c) Controla el espaciado entre las filas de la cuadrícula.
  - d) Establece el color de fondo de las celdas de la cuadrícula.

**SOLUCIONES:**1. **a** **b** **c** **d**2. **a** **b** **c** **d**3. **a** **b** **c** **d**4. **a** **b** **c** **d**5. **a** **b** **c** **d**6. **a** **b** **c** **d**7. **a** **b** **c** **d**8. **a** **b** **c** **d**9. **a** **b** **c** **d**10. **a** **b** **c** **d**