Tema 2.11 - Generación de Interfaces I - CSS (Grid)

Módulo: Desarrollo de Interfaces

Curso 2º DAM

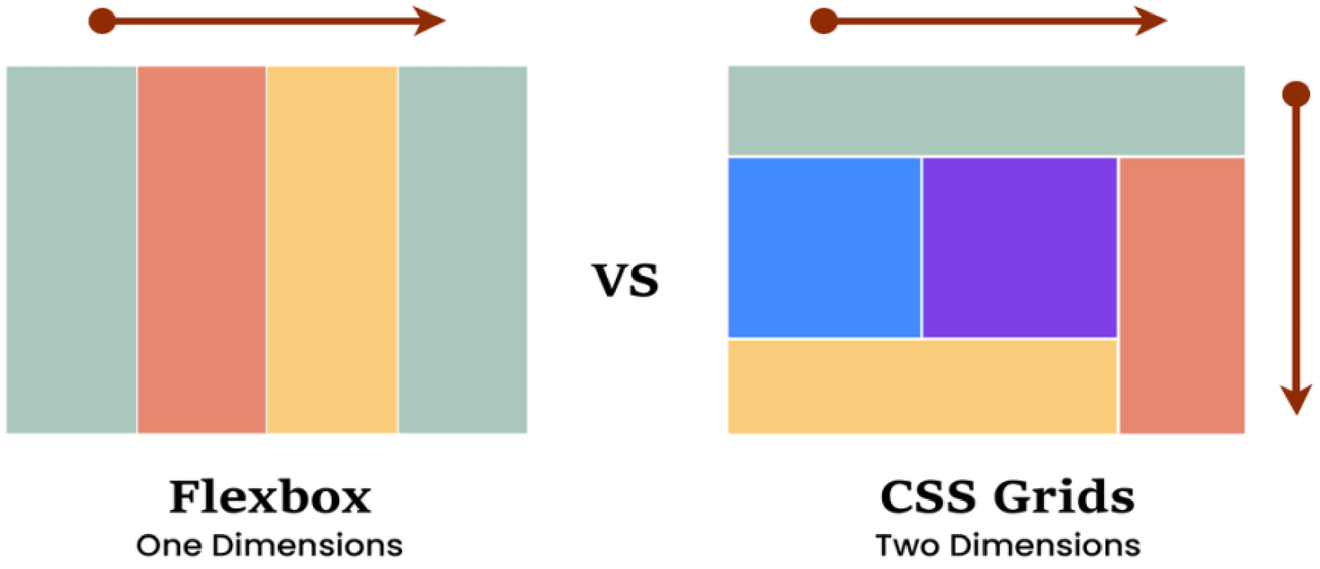
CSS Grid

Herramienta de maquetación **bidimensional** de la estructura de la interfaz.

CSS Grid permite crear estructuras de filas y columnas proporcionando una cuadrícula que para alinear elementos y distribuirlos en el espacio.

CSS Grid vs CSS Flexbox

La diferencia clave es que **Grid** está diseñado para controlar tanto filas como columnas en un **diseño bidimensional**, mientras que **Flexbox** está pensado para distribuir elementos en **una sola dimensión** (horizontal o vertical).



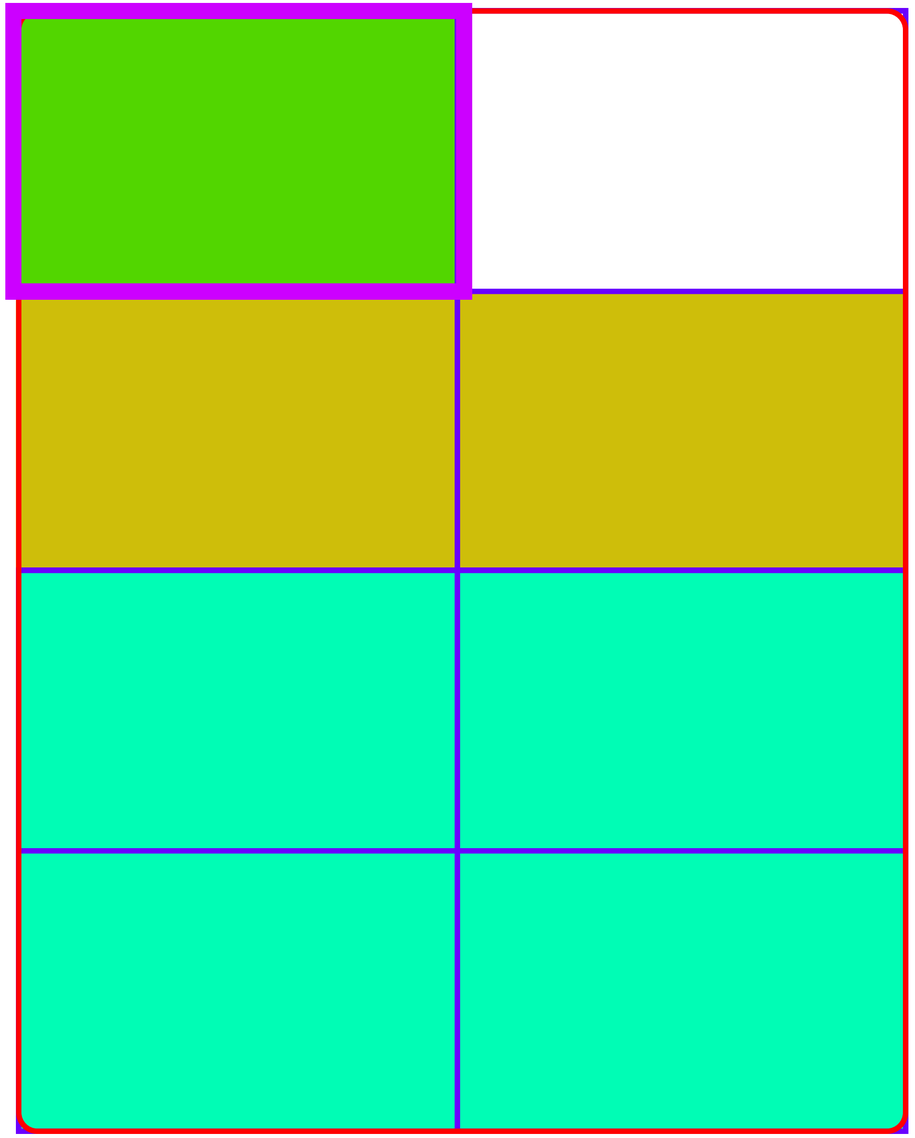
- **Grid es preferible** cuando el diseño requiere control sobre ambas dimensiones (filas y columnas) simultáneamente:
  - Galería de imágenes
  - Tabla de contenido.

- **Flexbox se usa mejor** para alinear elementos en una sola dimensión:
  - Menús y navegadores
  - Botones en línea.

Conceptos Básicos y Terminología de Grid

Para crear diseños basados en Grid CSS necesitaremos tener en cuenta una serie de conceptos que utilizaremos a partir de ahora y que definiremos a continuación:

- **Contenedor**: El elemento padre contenedor que definirá la cuadrícula o rejilla.
- **Ítem**: Cada uno de los hijos que contiene la cuadrícula (elemento contenedor).
- **Celda (grid cell)**: Cada uno de los cuadritos (unidad mínima) de la cuadrícula.
- **Línea (grid line)**: Separador horizontal o vertical de las celdas de la cuadrícula.
- **Area (grid area)**: Región o conjunto de celdas de la cuadrícula rodeada de 4 grid lines
- **Banda (grid track)**: Banda horizontal o vertical de celdas de la cuadrícula.



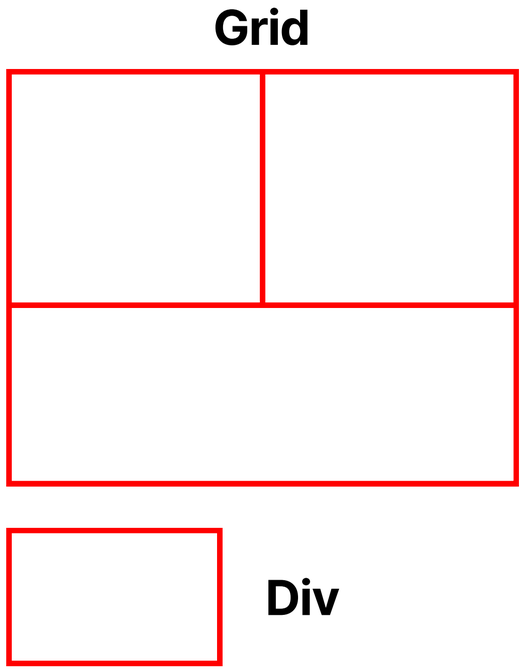
Propieadades del padre

Grid Container

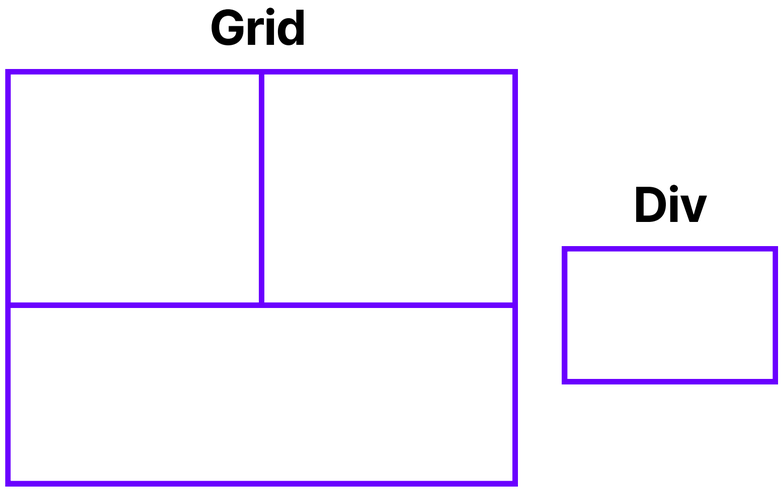
Disposicion del grid (Display: grid)

Para activar la cuadrícula grid en un elemento html, hay que utilizar sobre el elemento contenedor la propiedad **display** y especificar uno de los dos valores que queramos utilizar: **grid o inline-grid.**

```
.container {
  display: grid | inline-grid;
}
```



Display: grid → Genera una cuadrícula con comportamiento de bloque.



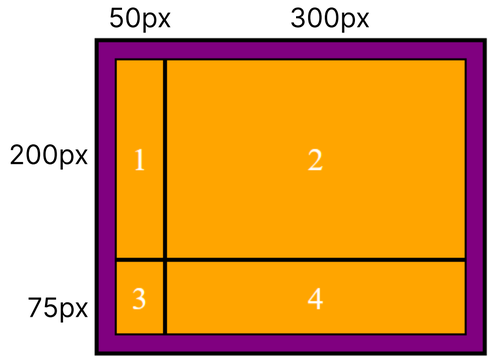
Display: inline-grid → Genera una cuadrícula con comportamiento de bloque en línea

Definir filas y columnas (grid-template-rows/columns)

En Grid CSS, la forma principal de definir una cuadrícula es indicar el tamaño de sus filas y sus columnas de forma explícita. Para ello, sólo tenemos que usar las propiedades CSS **grid-template-columns y grid-template-rows**

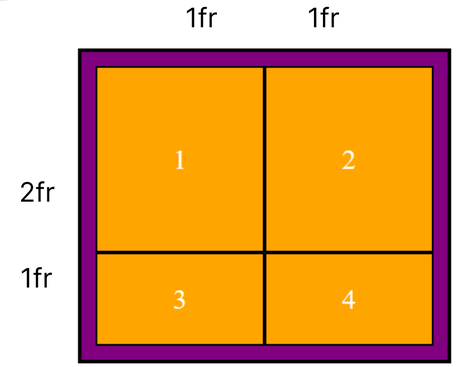
Filas y columnas fijas

```css
.container {
  display: grid;
  grid-template-columns: 50px 300px;
  grid-template-rows: 200px 75px;
}
```



Unidad fracción restante (fr)

```css
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 2fr 1fr;
}
```



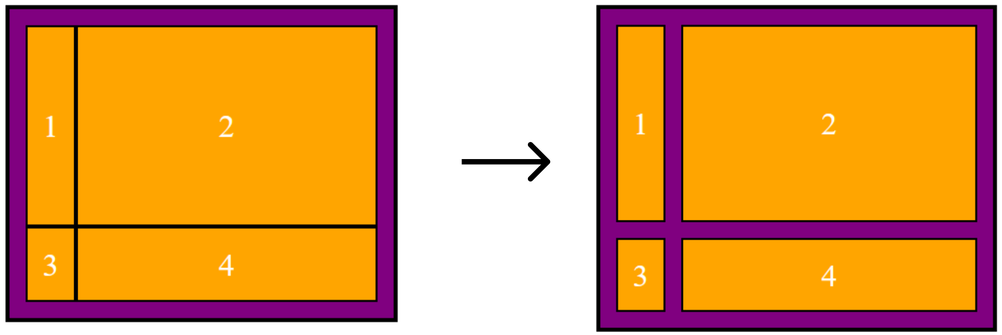
Podemos utilizar unidades relativas (o absolutas): **porcentajes**, la palabra clave **auto** (que obtiene el tamaño restante) o la unidad especial de grid **fr** (fracción restante)

Huecos en grid (row/column-gap)

Estas propiedades nos sirve para generar una separación entre los distintos grid-rows y grid-columns. Los valores que se pueden tomar son todas la medida absolutas y relativas, pero no los fr.

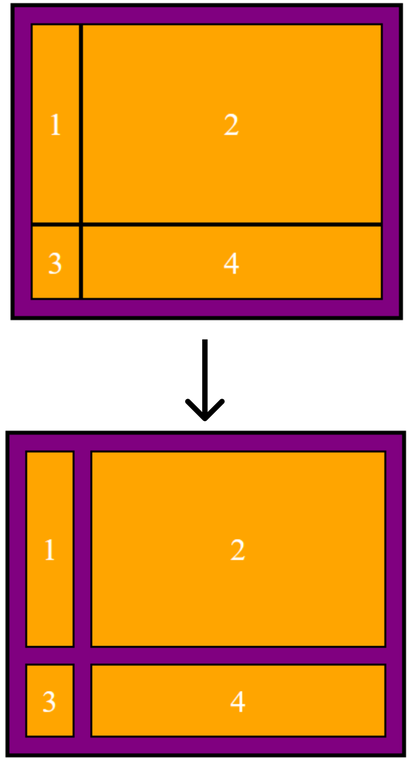
row-gap y column-gap

```css
.container {
  display: grid;
  grid-template-columns: 50px 300px;
  grid-template-rows: 200px 75px;
  row-gap: 1rem;
  column-gap: 1rem;
}
```



gap

```css
.container {
  display: grid;
  grid-template-columns: 50px 300px;
  grid-template-rows: 200px 75px;
  gap: 1rem;
}
```



Esta propiedad es un atajo para agrupar el column-gap y el row-gap, y se puede tomar 1 o 2 valores. Eso dependerá del efecto que queremos:

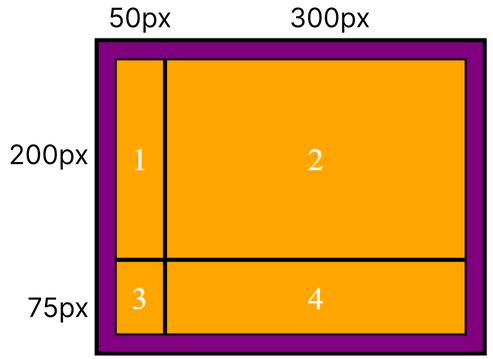
- **1 valor**: Define row y column gap a la vez
- **2 valores**: Define row y column gap por separado

Seleccionar filas y columnas (grid-rows/columns)

En Grid CSS, la forma principal de definir una cuadrícula es indicar el tamaño de sus filas y sus columnas de forma explícita. Para ello, sólo tenemos que usar las propiedades CSS **grid-template-columns y grid-template-rows**

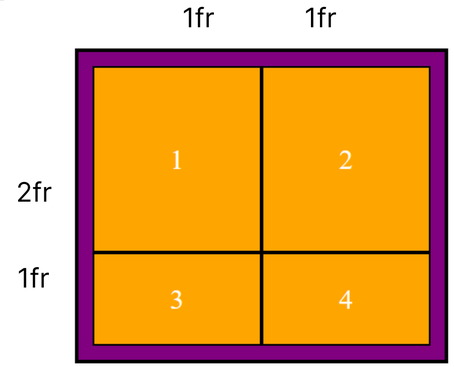
Filas y columnas fijas

```css
.container {
  display: grid;
  grid-template-columns: 50px 300px;
  grid-template-rows: 200px 75px;
}
```



Unidad fracción restante (fr)

```css
.container {
  display: grid;
  grid-template-columns: 1fr 1fr;
  grid-template-rows: 2fr 1fr;
}
```



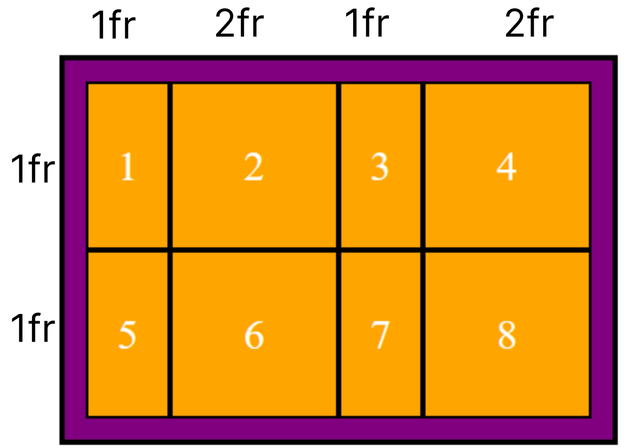
Podemos utilizar unidades relativas (o absolutas): **porcentajes**, la palabra clave **auto** (que obtiene el tamaño restante) o la unidad especial de grid **fr** (fracción restante)

Filas y columnas repetitivas (repeat)

En algunos casos, en las propiedades **grid-template-columns** y **grid-template-rows** podemos necesitar indicar las mismas cantidades un número alto de veces.

Se puede utilizar la función **repeat()** para indicar repetición de valores, señalando el número de veces que se repiten y el tamaño en cuestión.

```css
.container {
  display: grid;
  grid-template-columns: repeat(2,1fr 2fr);
  grid-template-rows: repeat(2,1fr);
}
```



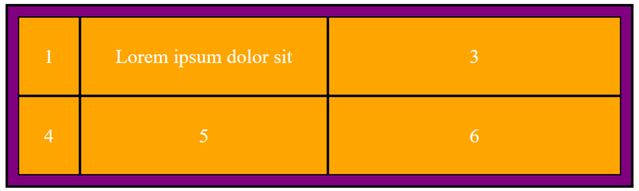
Definir tamaño dinámicamente: minmax, auto-fill, auto-fit

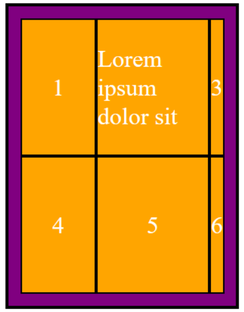
Esta medidas nos permiten ajustar de manera dinámica el tamaño de nuestros grid-tracks (filas y/o columnas) según el contenido que tengan los grid-items

minmax()

Es una medida que sirve para definir un valor mínimo que puede tomar el track y un valor máximo

```css
.container {
  display: grid;
  grid-template-columns: 100px minmax(150px, 400px) 1fr;
  grid-template-rows: repeat(2, 1fr);
}
```

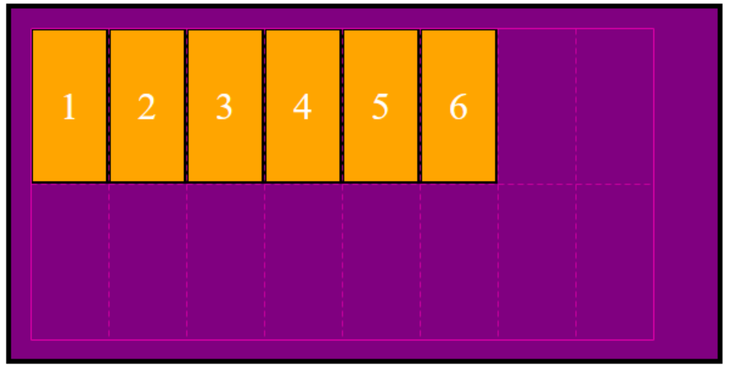




auto-fill

Junto a repeat. Nos permite generar columnas dinámicamente según el valor que le coloquemos al repeat y el espacio del contenedor. Genera hasta ocupar todo el espacio.

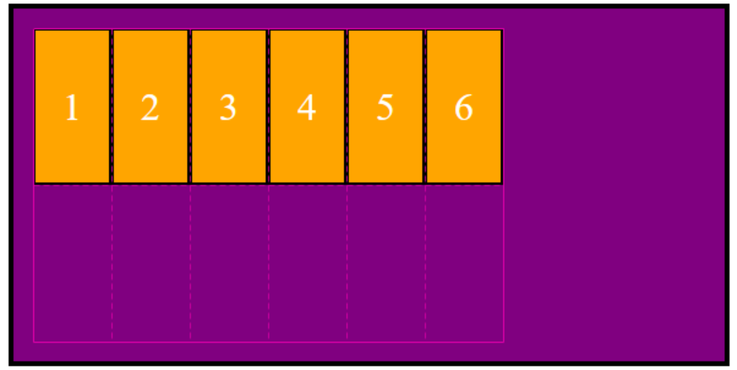
```css
.container {
  display: grid;
  grid-template-columns: repeat(auto-fill, 4rem);
  grid-template-rows: repeat(2, 1fr);
}
```



auto-fit

Este método ocupa los dos conceptos anteriores. Permite pasar un valor máximo al que llega el elemento y como mínimo toma el min-content. Genera hasta lo necesario

```css
.container {
  display: grid;
  grid-template-columns: repeat(auto-fit, 4rem);
  grid-template-rows: repeat(2, 1fr);
}
```



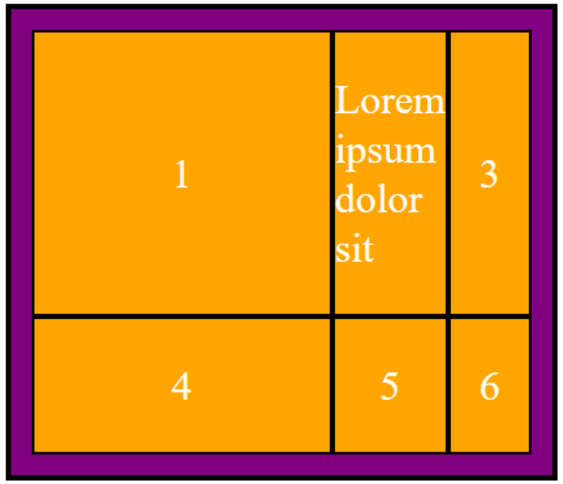
Definir tamaño dinámicamente: min-content, max-content, fit-content

Esta medidas nos permiten ajustar de manera dinámica el tamaño de nuestros grid-tracks (filas y/o columnas) según el contenido que tengan los grid-items

min-content

Se adapta el mínimo tamaño que pueda tomar el track en función del grid-item

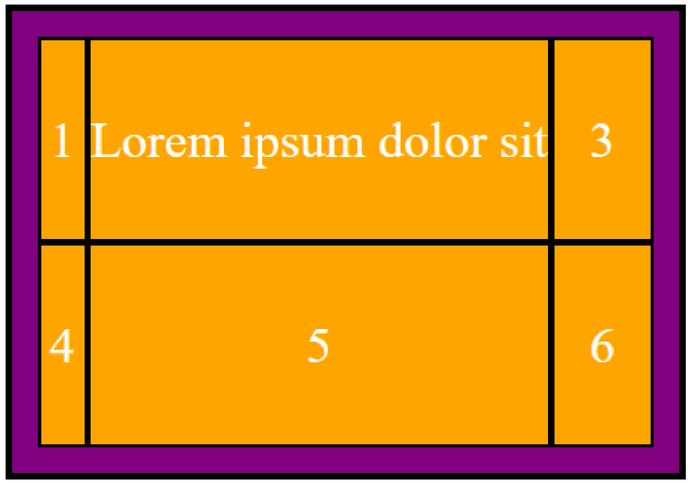
```css
.container {
  display: grid;
  grid-template-columns: 1fr min-content 4rem;
  grid-template-rows: repeat(2,1fr);
}
```



max-content

Se adapta al máximo tamaño que pueda tener el contenido

```css
.container {
  display: grid;
  grid-template-columns: 1fr max-content 4rem;
  grid-template-rows: repeat(2,1fr);
}
```



fit-content

Este método ocupa los dos conceptos anteriores. Permite pasar un valor máximo al que llega el elemento y como mínimo toma el min-content.

```css
.container {
  display: grid;
  grid-template-columns: 1fr fit-content(10rem) 4rem;
  grid-template-rows: repeat(2,fit-content(0.1rem));
}
```