Curso de Tailwind CSS

# ¿Qué es Tailwind CSS?

**¿**[**Qué es TailWind CSS**](https://tailwindcss.com/)**?** 🧐  
Es un *[framework](https://en.wikipedia.org/wiki/Software_framework" \t "_blank)* que permite la contrucción de diseños altamente personalizados y de bajo nivel.

**¿**[**Por qué usar TailWind CSS**](https://tailwindcss.com/#what-is-tailwind)**?** 🤨  
A diferencia de otros frameworks css, como lo es [bootstrap](https://getbootstrap.com/" \t "_blank), que otorgan componentes prediseñados, Tailwind provee clases a bajo nivel de css que nos permiten construir diseños completamente personalizados por nostros y sin tener que pelear con el framework en sí.

Nos externa una [responsividad](https://tailwindcss.com/docs/responsive-design/" \t "_blank) desde la misma sintaxis.

Trabaja de manera amigable en el crecimiento de nuestro proyecto al proveernos herramientas para [extraer clases](https://tailwindcss.com/docs/extracting-components/).

Pero sobre todo, nos abre su código para [personalizarlo](https://tailwindcss.com/docs/configuration/) ya que esta escrito en [PostCSS](https://postcss.org/" \t "_blank) y configurado en JavaScript.

Tailwind is more than a CSS framework, *it’s an engine for creating design systems*.

# ¿Qué es Tailwind CSS?

Dejo por aquí unos apuntes que cree para iniciar la configuración desde 0, por si le sirve a la comunidad.

## Dependencias

- `npm init -y`

- `npm install tailwindcss autoprefixer postcss-cli`

### Inicializamos las herramientas instaladas:

<!-- Genera archivo configuracion vacio de nombre tailwind.config.js -->

- `npx tailwindcss init`

<!-- Genera archivo configuracion completo -->

- `npx tailwindcss init tailwind.config.full.js --full`

<!-- Plugin recomendado para VSCode: Tailwind CSS IntelliSense -->

### Creamos archivo de configuracion postcss.config.js

<!-- Instrucciones archivo postcss.config.js: -->

- `touch postcss.config.js`

- `module.exports = { plugins: [require('tailwindcss'), require('autoprefixer')], };`

### Creacion archivo html y origen CSS

- `mkdir css`

- `touch css/tailwind.css`

### Configuracion archivo css/tailwind.css

- `@tailwind base; @tailwind components; @tailwind utilities;`

### Completamos script en package.json

- `"scripts": { "build": "postcss css/tailwind.css -o public/css/styles.css",`

<!-- Para autoregenerar el tailwind.css cuando creamos paquetes -->

- `"dev": "postcss css/tailwind.css -o public/css/styles.css --watch"}`

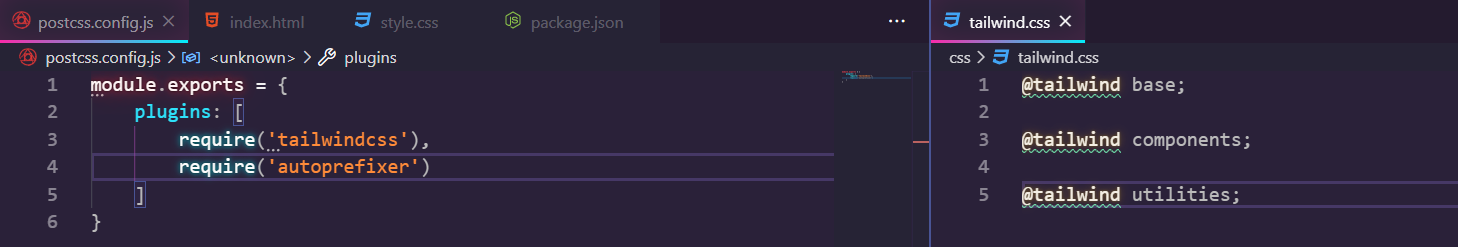
<!-- Ejecutar para compilar -->

- `npm run dev`

### Inicializamos script para crear el css

<!-- genera una directorio css con su styels.css en la carpeta public -->

- `npm run build`



# Directivas de Tailwind

Directiva es una instrucción que utiliza tailwind para insertar código en el archivo final de css que genera.

@tailwind base  
Esto inyecta los estilos base de Tailwind y cualquier estilo base registrado por plugins.

@tailwind components  
Esto inyecta las clases de componentes de Tailwind y cualquier clase de componente registrado por los plugins.

@tailwind utilities  
Esto inyecta las clases de utilidad de Tailwind y cualquier clase de utilidad registrada por los plugins.

@tailwind screens  
Esta directiva sirve para controlar donde Tailwind inyecta las variaciones responsivas de cada utilidad. Si se omite, Tailwind añadirá estas clases al final de tu css, por defecto.

# Personalización y configuración

## Configuración

Ya que Tailwind es un framework para construir UI a la medida, por default, se tiene un archivo opcional llamado tailwind.config.js en la raíz de la carpeta, donde está el package.json.

## Creando un archivo de configuración

Para generar un archivo de configuración para Tailwind, podemos usar el Tailwind CLI:

npx tailwind init

**Nota**. Podemos utilizar esta herramienta cuando instalamos la dependencia via npm.  
Donde como resultado tendremos tailwind.config.js:

module.exports = {

theme: {},

variants: {},

plugins: [],

}

Cabe mencionar que cada sección, del archivo de configuración, es opcional.

## La sección Theme

Esta sección es donde definimos los aspectos relacionados con el diseño visual de nuestro sitio.

...

theme: {

screens: {

sm: '640px',

md: '768px',

lg: '1024px',

xl: '1280px',

},

fontFamily: {

display: ['Gilroy', 'sans-serif'],

body: ['Graphik', 'sans-serif'],

},

borderWidth: {

**default**: '1px',

'0': '0',

'2': '2px',

'4': '4px',

},

extend: {

colors: {

cyan: '#9cdbff',

},

spacing: {

'96': '24rem',

'128': '32rem',

}

}

}

...

## La sección Variants

Esta sección nos permite controlar el comportamiento de las utilidades core, como responsive variants y pseudo-class variants.

...

variants: {

appearance: ['responsive'],

// ...

borderColor: ['responsive', 'hover', 'focus'],

// ...

outline: ['responsive', 'focus'],

// ...

zIndex: ['responsive'],

},

...

## La sección Plugins

Esta sección nos permite registrar plugins de terceros con el objetivo de extender utilidades, componentes, estilos, etc.

...

plugins: [

require('tailwindcss-transforms'),

require('tailwindcss-transitions'),

require('tailwindcss-border-gradients'),

],

...

# Responsive Design, Mobile First y Utility First

**Responsive Web Design**

La web como la conocemos, hoy en día, no es una tecnología pensando en un **UX** o *User Experience* generando, desafortunadamente, que muchos de los sitios web no estén optimizados para los **dispotivos móviles**.

Definido por [Ethan Marcotte](https://alistapart.com/article/responsive-web-design/" \t "_blank), es una filosofía que responde a las necedades de los usuarios y a los dispositivos que estamos usando.

**Mobile First**

Como su nombre sugiere, significa que iniciaremos con el diseño de móviles y expandiendo éstas características para crear una verión en tableta o escritorio/web tradicional.

Cabe mencionar que esta filosofía no es sinónimo de limitación, por lo que tenemos que tener el mismo contenido tanto en escritorio como en móvil. Google describe las *[best practices](https://developers.google.com/search/mobile-sites/mobile-first-indexing" \t "_blank)* en su sitio.

**Utility first**

CSS posee diferentes tipos de paradigmas para abstraer un diseño, como BEM descrito por [Tailwind](https://tailwindcss.com/docs/utility-first/" \t "_blank) en ésta sección:

# Colores

Los elementos que pueden ser afectados por los colores son:

* Fondo.
* Texto.
* Bordes.
* Placeholder.

Taildwind por default tiene valores ya predefinidos, para conocerlos tenemos que generar un archivo de configuración con todos los valores completos:  
npx tailwindcss init tailwind.config.full.js --full

Se puede ver la paleta de colores por defecto aquí:  
<https://tailwindcss.com/docs/customizing-colors/#default-color-palette>

# Dimensiones y Espacios

Tailwind utiliza un sistema de espacios que se puede configurar. Se maneja internamente utilizando rem. Se puede visualizar desde el archivo tailwind.config.full.js.

Se puede utilizar en:

* Height.
* Width.
* Margin.
* Padding.

En las clases de los elementos se añaden cosas como:  
h-32 w-1/2 pt-2 mx-auto

Para manajear el height:  
<https://tailwindcss.com/docs/height/#app>

Para manejar width en porcentajes se puede ver esta documentación:  
<https://tailwindcss.com/docs/width/#app>



# Cambiando las propiedades de la tipografía

## @font-faces

Para aprovechar la regla @font-facepara realizar cualquier personalización fuera de Tailwindcss modificamos el archivo CSS principal de la siguiente manera:

@**tailwind** base;

@**font-face** {

font-family: Proxima Nova;

font-weight: 400;

src: url(/fonts/proxima-nova/400-regular.woff) format("woff");

}

@**font-face** {

font-family: Proxima Nova;

font-weight: 500;

src: url(/fonts/proxima-nova/500-medium.woff) format("woff");

}

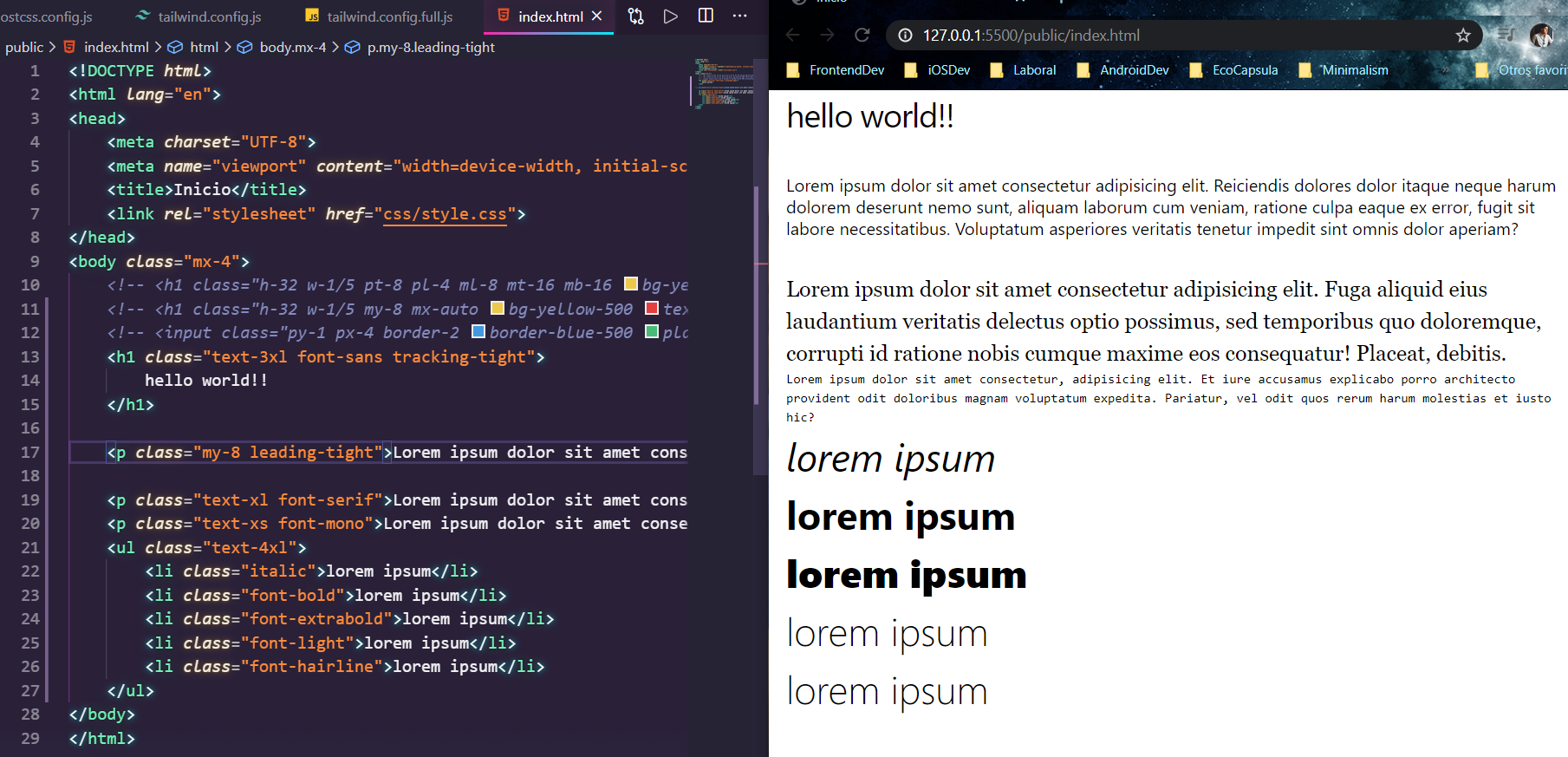
@**tailwind** components;

@**tailwind** utilities;

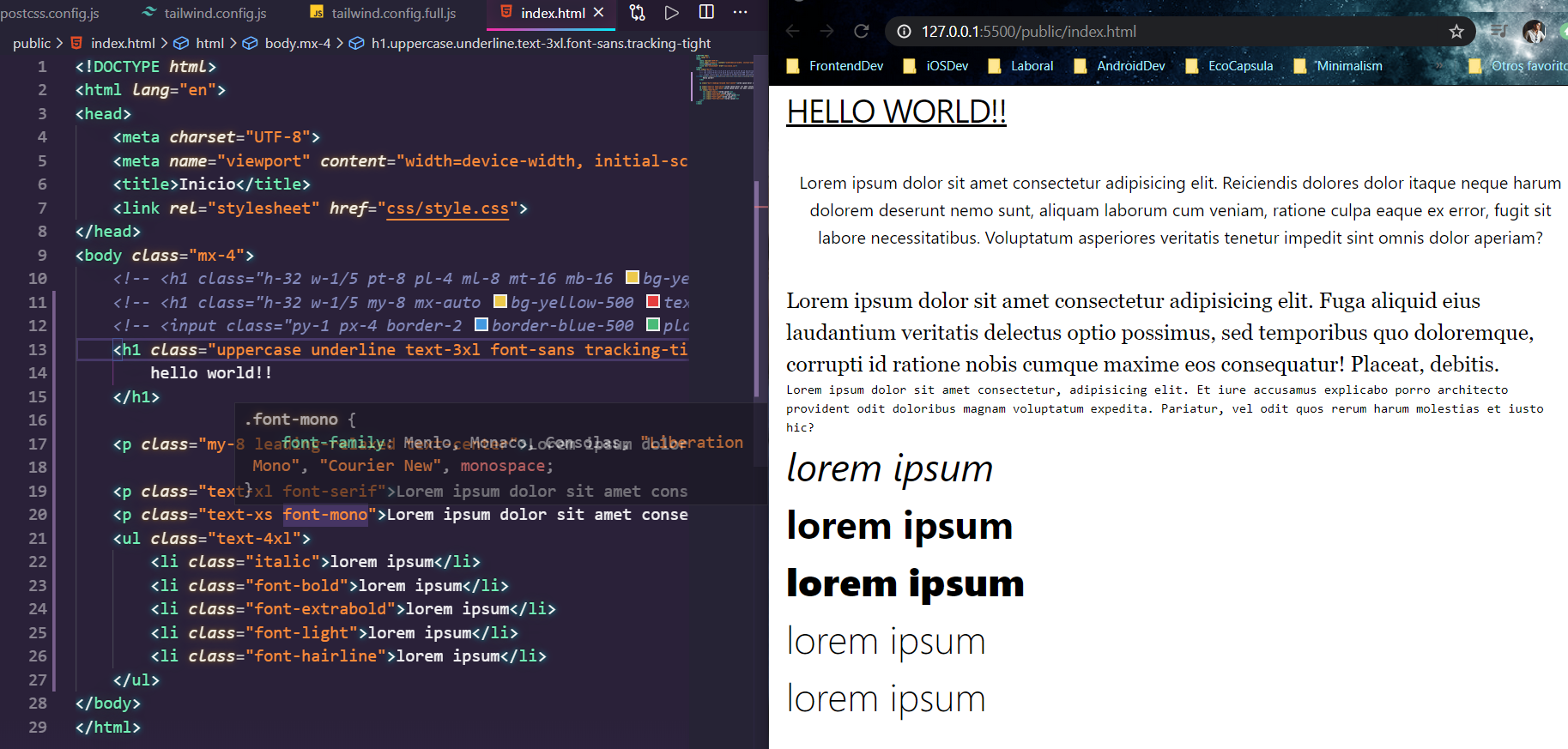
**Nota**. Para definir cualquier personalización hacia los estilos base, agregamos nuestros estilos después de @tailwind base y antes de @tailwind components para evitar problemas de compilación.

# Ajustando el espaciado entre letras y líneas

2 instrucciones que debemos aprender:  
**Tracking** espaciamiento entre letras  
**Leading** espaciamiento entre lineas







# Display

Las propiedades de display son útiles para poder controlar las dimensiones o espacios. Las propiedades básicas de display son:

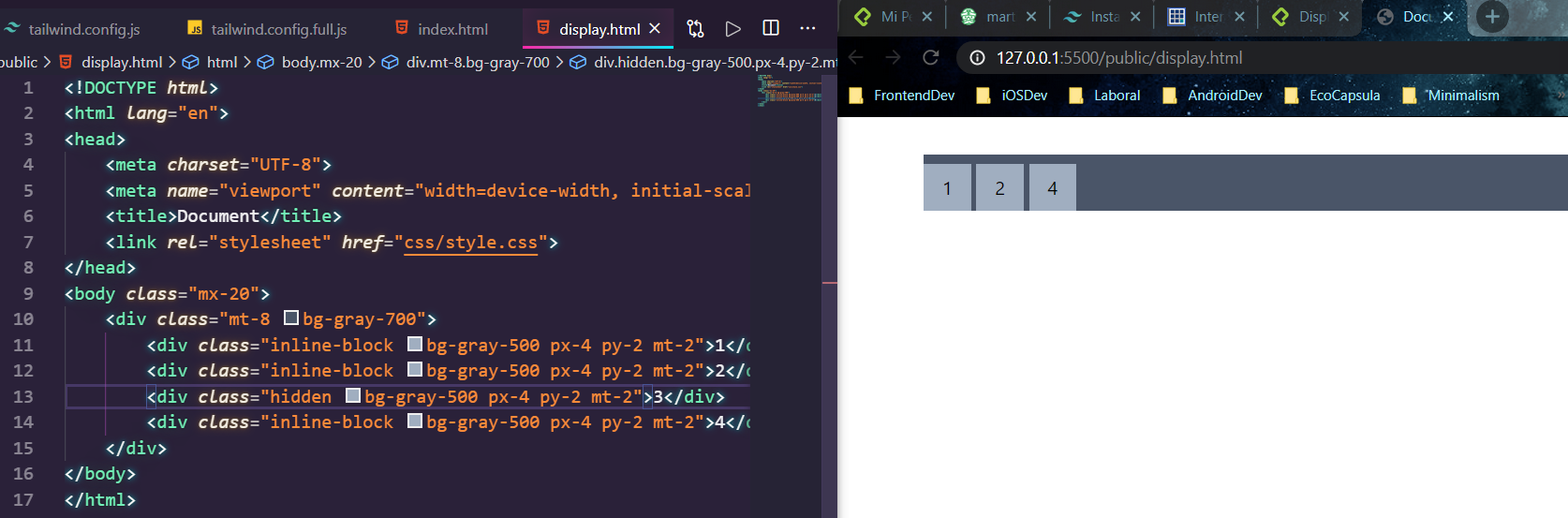
* Block → Con esta configuración los bloques abarcan toda la pantalla.
* <div class="text-center">
* <div class="block bg-blue-800">1</div>
* <div class="block bg-blue-500">2</div>
* <div class="block bg-blue-300">3</div>
* </div>
* Inline-block → Sólo ocupan el espacio necesario para mostrar lo que hay en su interior.
* <div class="text-center">
* <div class="inline-block bg-blue-800">1</div>
* <div class="inline-block bg-blue-500">2</div>
* <div class="inline-block bg-blue-300">3</div>
* </div>
* Inline → Sólo ocupan el espacio necesario para mostrar lo que hay en su interior y la altura del elemento es indiferente.
* <div class="text-center">
* <div class="inline bg-blue-800">1</div>
* <div class="inline bg-blue-500">2</div>
* <div class="inline bg-blue-300">3</div>
* </div>
* Hidden → El elemento en cuestión no se muestra.
* <div>
* <div class="inline-block bg-blue-800">1</div>
* <div class="inline-block bg-blue-500">2</div>
* <div class="hidden bg-blue-300">3</div>
* <div class="inline-block bg-blue-100">4</div>

</div>

Caja de Presentación

“Estas propiedades pueden controlar las dimensiones o espacios.”

-Block - en la caja de presentación no cambia con block.  
-Inlilne-block - se posiciona dentro de la caja.  
-Inline - se posiciona fuera de la caja.  
-Hideden - se oculta el elemento.



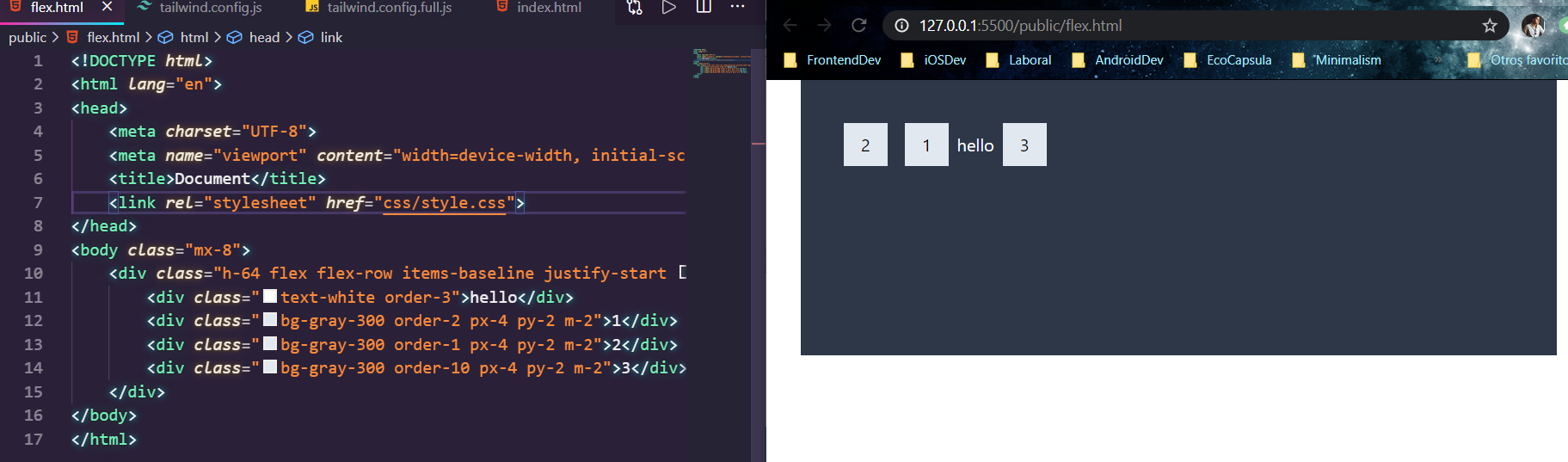
# Flexbox

Flex es otra propiedad de display pero nos habilita otras propiedades para manipular nuestras cajas. Las propiedades propias de flexbox son:

1. Flex direction
2. <div class="flex flex-col">
3. <div class="bg-blue-300">1</div>
4. <div class="bg-blue-300">2</div>
5. <div class="bg-blue-300">3</div>
6. </div>
7. Align items
8. <div class="flex items-center">
9. <div class="bg-blue-300">1</div>
10. <div class="bg-blue-300">2</div>
11. <div class="bg-blue-300">3</div>
12. </div>
13. Justify content
14. <div class="flex justify-center">
15. <div class="bg-blue-300">1</div>
16. <div class="bg-blue-300">2</div>
17. <div class="bg-blue-300">3</div>
18. </div>
19. Order
20. <div class="flex justify-center items-center">
21. <div class="order-3 bg-blue-300">1</div>
22. <div class="order-1 bg-blue-300">2</div>
23. <div class="order-2 bg-blue-300">3</div>
24. </div>

Flexbox es un tema muy amplio, así que los invito a leer más a detalle sobre este tema en este otro artículo.

[Una guía completa de Flexbox!](https://medium.com/@alexcamachogz/una-gu%C3%ADa-completa-de-flexbox-768b038de5e9)



En esta clase vamos a aprender a crear un grid de columnas utilizando flex en Tailwind.

# grid basico

Verás que con lo que hemos aprendido hasta ahora es muy sencillo crear un grid.

1. Crearemos un elemento que contendrá las columnas de nuestro Grid con la clase de flex.

<div **class**=“flex”>;

</div>

1. Teniendo de base el elemento del punto anterior insertaremos una etiqueta por cada columna que queramos tener de la siguiente manera.

Una columna

<**div** class="flex">

<**div**

class="w-full

p-4 bg-blue-800"

></**div**>

</**div**>

Resultado  


Dos columnas

<**div**>

<**div**

class="w-1/2

p-4 bg-blue-800"

></**div**>

<**div**

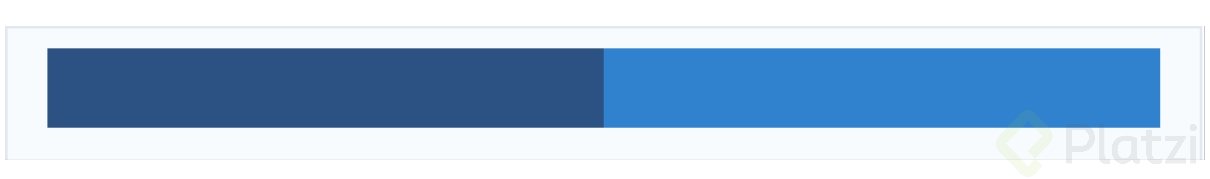
class="w-1/2

p-4 bg-blue-600"

></**div**>

</**div**>

Resultado



Tres Columnas

<**div** class="flex">

<**div**

class="w-1/3

p-4 bg-blue-800"

></**div**>

<**div**

class="w-1/3

p-4 bg-blue-600"

></**div**>

<**div**

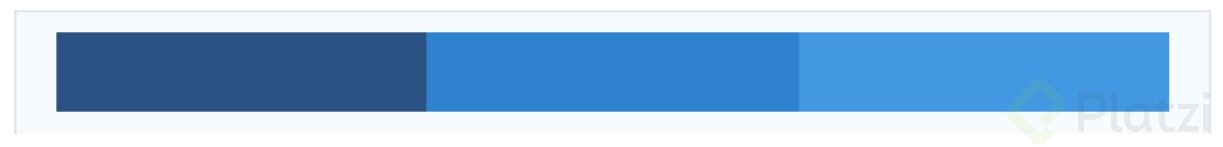
class="w-1/3

p-4 bg-blue-500"

></**div**>

</**div**>

Resultado



Cuatro Columnas

<**div** class="flex">

<**div**

class="w-1/4

p-4 bg-blue-700"

></**div**>

<**div**

class="w-1/4

p-4 bg-blue-400"

></**div**>

<**div**

class="w-1/4

p-4 bg-blue-600"

></**div**>

<**div**

class="w-1/4

p-4 bg-blue-500"

></**div**>

</**div**>

Cinco columnas

<**div** class="flex">

<**div**

class="w-1/5

p-4 bg-blue-700"

></**div**>

<**div**

class="w-1/5

p-4 bg-blue-400"

></**div**>

<**div**

class="w-1/5

p-4 bg-blue-600"

></**div**>

<**div**

class="w-1/5

p-4 bg-blue-500"

></**div**>

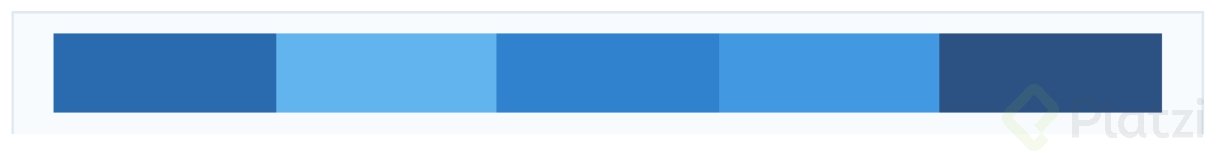
<**div**

class="w-1/5

p-4 bg-blue-800"

></**div**>

</**div**>

Resultado  


Seis columnas

<**div** class="flex">

<**div**

class="w-1/6

p-4 bg-blue-700"

></**div**>

<**div**

class="w-1/6

p-4 bg-blue-400"

></**div**>

<diva

class="w-1/6

p-4 bg-blue-600"

></**div**>

<**div**

class="w-1/6

p-4 bg-blue-500"

></**div**>

<**div**

class="w-1/6

p-4 bg-blue-300"

></**div**>

<**div**

class="w-1/6

p-4 bg-blue-800"

></**div**>

</**div**>

Resultado



## Grid Responsivo con Columnas de Anchura Fija

La diferencia para lograr que nuestro grid sea responsivo es que al elemento base donde tenemos la clase de flex le agregamos flex-wrap con esto logramos que cuando uno de los elemento exceda el tamaño, ese elemento pase al siguiente renglón. Y con el uso de las pseudo-classes responsivas de sm, md, lg y xl logramos el numero de columnas deseada por tamaño de pantalla.

<**div** class="flex flex-wrap m-4">

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/3 lg:w-1/4 xl:w-1/6

p-4 bg-green-700 mb-1"

></**div**>

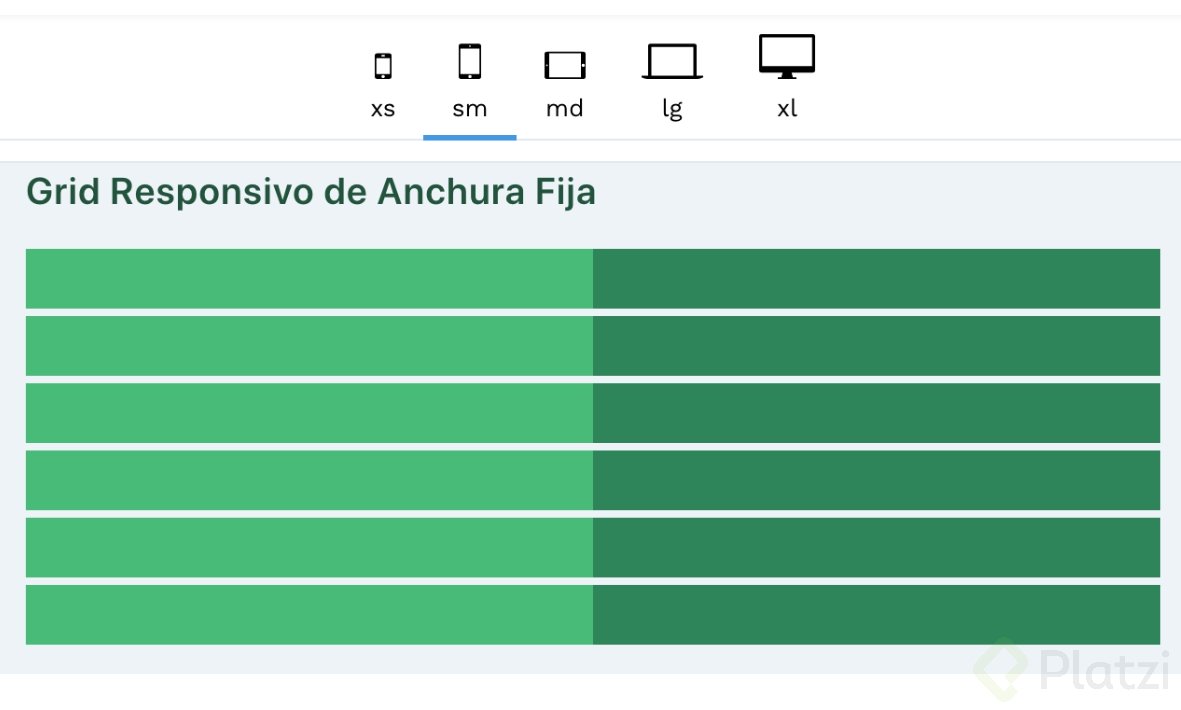
</**div**>

### Resultados

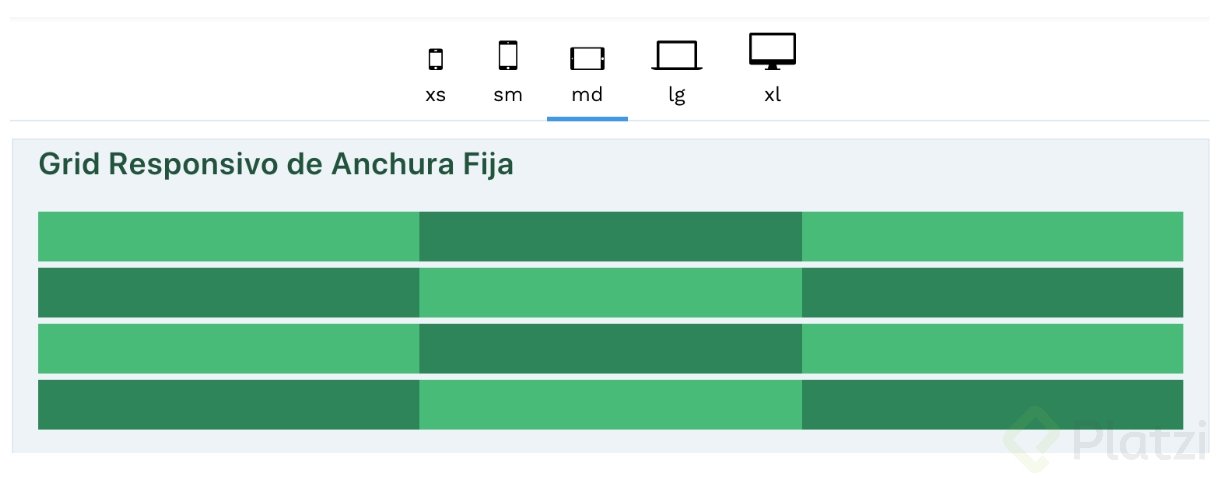
**default o xs**

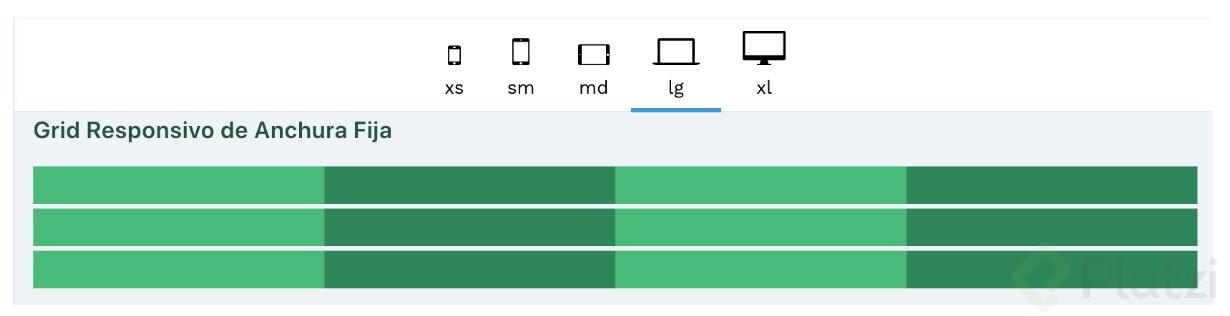


**sm**



**md**

  
&amp;amp;amp;amp;gt; **lg**



**xl**



## Grid Responsivo con Columnas de Anchura Variable

Al igual que con el grid responsivo anterior, agregamos en el elemento base de flex la clase de flex-wrap y de igual manera hacemos uso de las pseudo-classes responsivas de sm, md, lgy xl.

<**div** class=“flex flex-wrap m-4”>

<**div**

class="w-full sm:w-1/2 md:w-2/5 lg:w-1/6 xl:w-7/12

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/3 md:w-1/5 lg:w-3/6 xl:w-1/12

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/6 md:w-2/5 lg:w-2/6 xl:w-4/12

p-4 bg-green-900 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/6 md:w-1/5 lg:w-1/6 xl:w-1/12

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/3 md:w-3/5 lg:w-1/6 xl:w-5/12

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-1/5 lg:w-4/6 xl:w-6/12

p-4 bg-green-900 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/3 md:w-1/5 lg:w-2/6 xl:w-3/12

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/6 md:w-1/5 lg:w-2/6 xl:w-4/12

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-3/5 lg:w-2/6 xl:w-5/12

p-4 bg-green-900 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-3/5 lg:w-3/6 xl:w-8/12

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/6 md:w-1/5 lg:w-2/6 xl:w-2/12

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/3 md:w-1/5 lg:w-1/6 xl:w-2/12

p-4 bg-green-900 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/2 md:w-2/5 lg:w-2/6 xl:w-5/12

p-4 bg-green-700 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/3 md:w-1/5 lg:w-1/6 xl:w-3/12

p-4 bg-green-500 mb-1"

></**div**>

<**div**

class="w-full sm:w-1/6 md:w-2/5 lg:w-3/6 xl:w-4/12

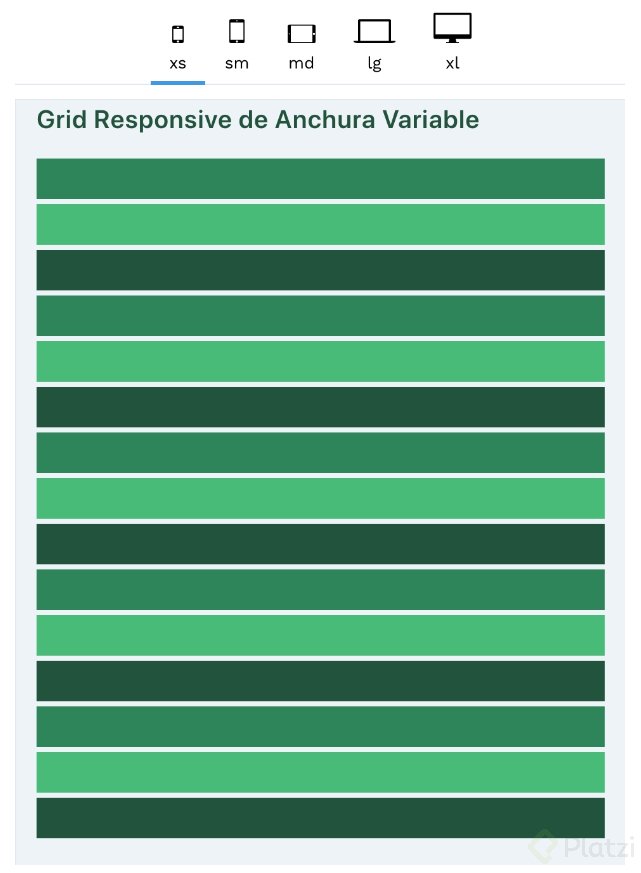
p-4 bg-green-900 mb-1"

></**div**>

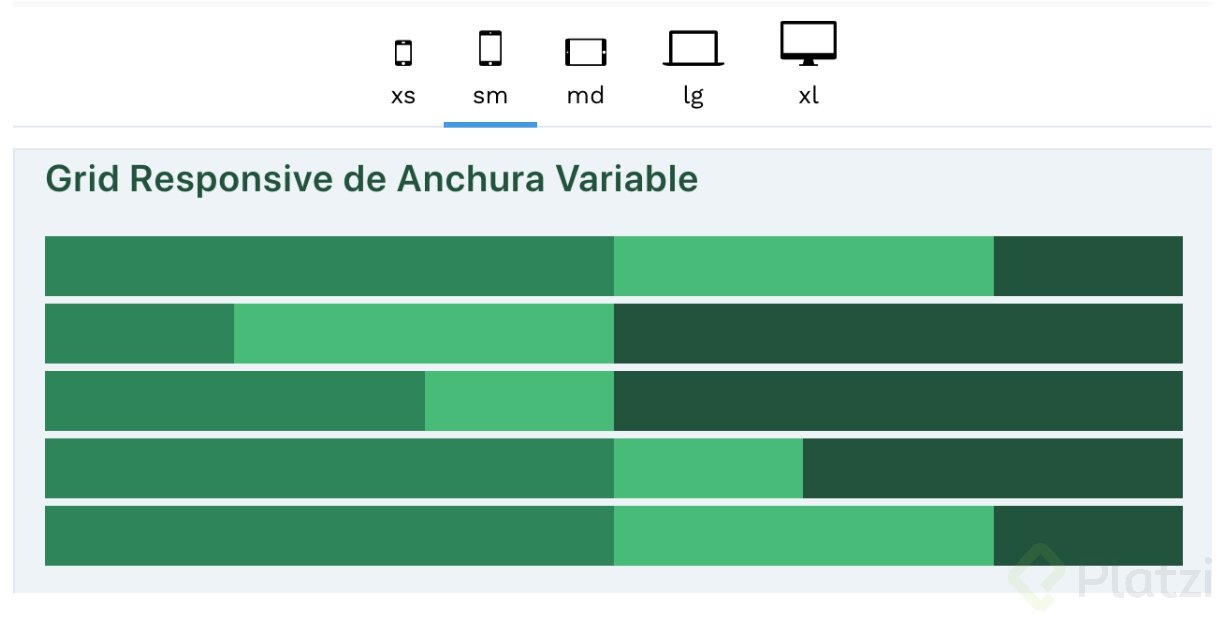
</**div**>

### Resultados

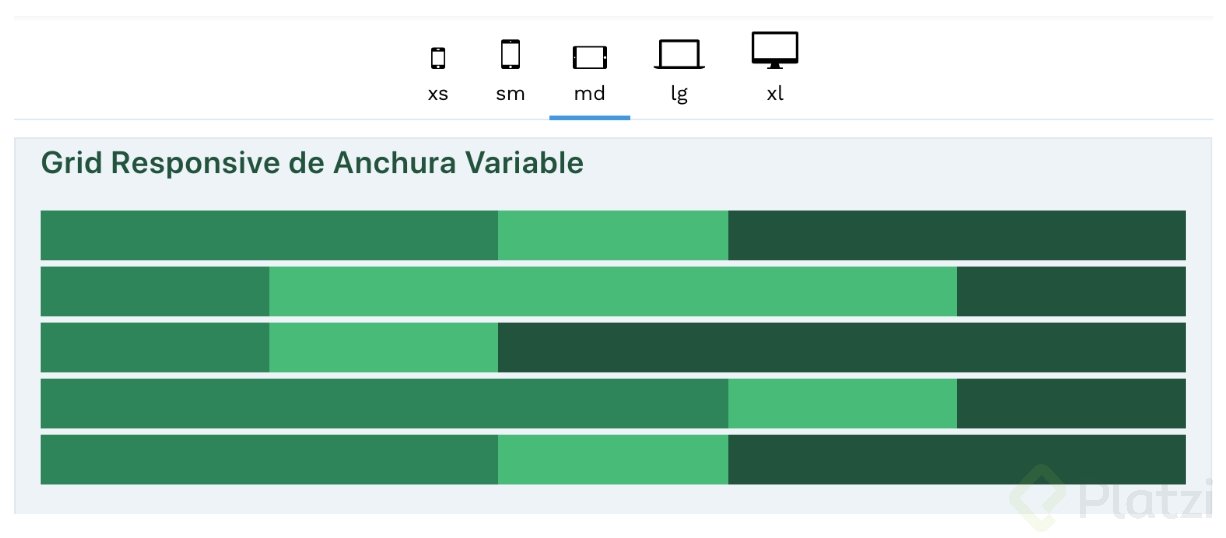
**default o xs**



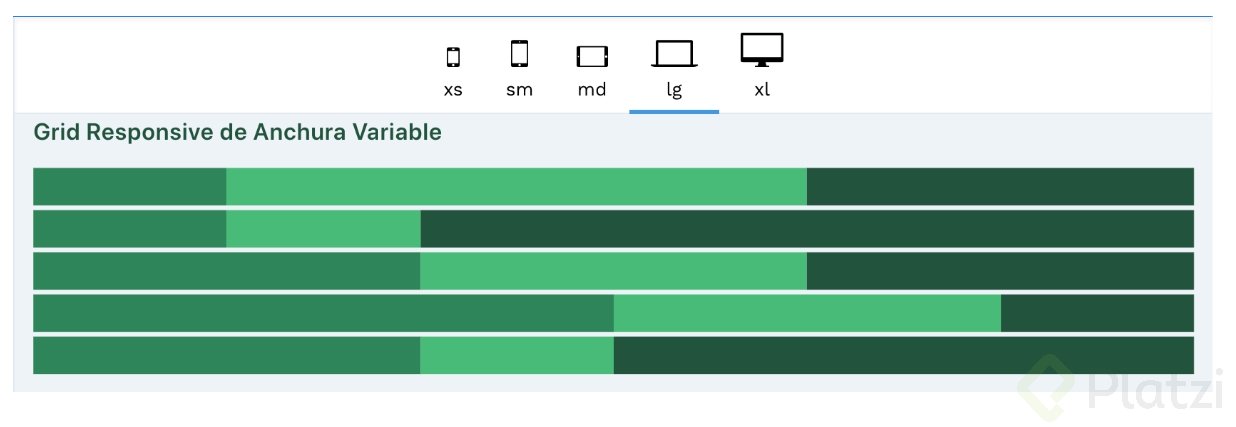
**sm**

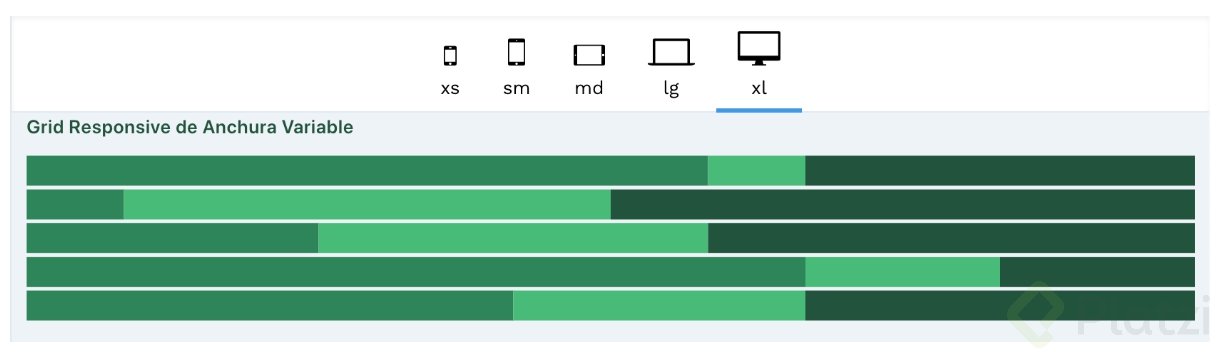


**md**



**lg**



**xl**  


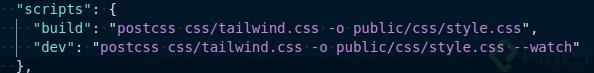
## Conclusión

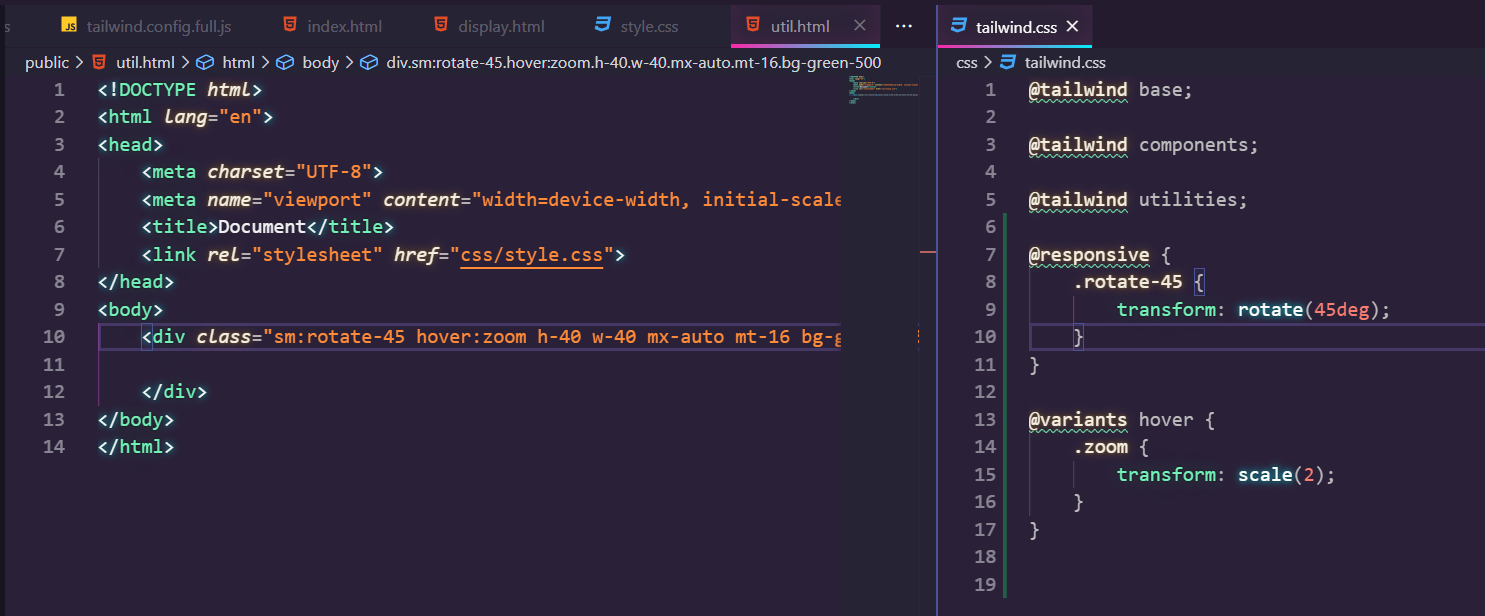
Crear grids en Tailwind usando flex es muy sencillo.  
En flex podemos hacer uso de las clases de dimensiones porcentuales, algunos ejemplos son:  
w-1/2, w-2/3, w-3/4, w-5/6, w-7/12,.  
Estas mediciones nos facilita poder crear columnas de 2, 3, 4, 5, 6 y 12 columnas y sus combinaciones.

Reportar un problema

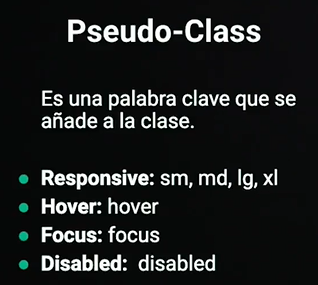
**Crea tus propias utilerías**

# Crea tus propias utilerías

Para que funcione el @variants y @responsive tienes que compilar el archivo css/tailwind.css y esto es corriendo el comando npm run build en la terminal en la carpeta del proyecto ‘platzifood’ cada vez que hagas un cambio supongo ooo otra opción es agregar un script en el archivo package.json que sería el siguiente:  
  
Aegurate de poner style y no styles a mi me pasó xd  
Por ultimo en la terminal en la carpeta del proyecto corres el comando npm run dev y estara watcheando cada cambio que hagas y se mostrará automáticamente.



# Entendiendo las variantes y las pseudo-clases

Aquí esta el código que agrega de la documentación

<**button** class="disabled:opacity-75 bg-blue-500">

**Submit**

</**button>**

El segundo código que agrega

**variants**: {

opacity: ['responsive', 'hover', 'focus', 'disabled'],

},

link de la documentación:  
<https://tailwindcss.com/docs/pseudo-class-variants/#app>

# ¡Es hora de practicar!

Tip: Una buena práctica al combinar colores, como en botones por ejemplo, es no poner la letra color blanco o negro. En su lugar utiliza las tonalidades del mismo color para hacerla resaltar.

● Crea un botón color azul siguiendo estos consejos, recuerda que las tonalidades predefinidas van del 100 al 900.  
● Crea un botón que se adapte según el tamaño de dispositivo.  
● Cuando sea de tamaño pequeño, este botón debe abarcar el ancho de la pantalla y cuando sea un poco más grande debe tener un tamaño predeterminado.

Si no estás seguro sobre cuáles colores escoger, te recomiendo que uses Paletton, ahí podrás escoger un color y el programa te sugerirá colores que puedan combinar. (<https://paletton.com/>)

### Display y Flexbox

Tip: Una buena práctica es utilizar el margen de cada elemento para posicionarlo con respecto a otros. Esto se logra con margin left (ml-x), margin-top (mt-x). Si usamos margin right y margin bottom lo que hacemos es mover otros elementos respecto del elemento con el que estamos trabajando.

● Crea lo que se conoce como group button de 3 buttons o más. Esto lo logras utilizando lo aprendido en display o flexbox, así como margin y padding.

### Pseudo—Class Variants

Utilizando variantes como hover y active, logra crear efectos cambiando el color de los botones creados anteriormente.

● Cuando pases el cursor encima de uno de los botones deberá cambiar el fondo a un color más oscuro, de tal manera que resalte con el texto. Y al estar en estado normal, el fondo deberá ser claro y el texto, color oscuro.

### Extraer componentes usando @apply

Utilizando la directiva @apply, crea un componente de “alert” con sus variaciones para que se comporte distinto según la clase que le acompañe.

Hay que crear:  
● alert (default) - Utiliza un color neutro  
● alert (danger) - Utiliza un color rojo  
● alert (warning) - Utiliza un color amarillo o naranja  
● alert (info) - Utiliza un color azul

Para desarrollar este reto tendrás que hacer uso de:  
● margin  
● padding  
● border  
● background  
● text color

El código de tu componente de poderse utilizar como sigue:  
El código de tu componente debería verse así:

Este mensaje es una alerta default.

</**div**>

<**div** class=“alert alert-danger”>

Este mensaje es una alerta danger.

</**div**>

<**div** class=“alert alert-warning”>

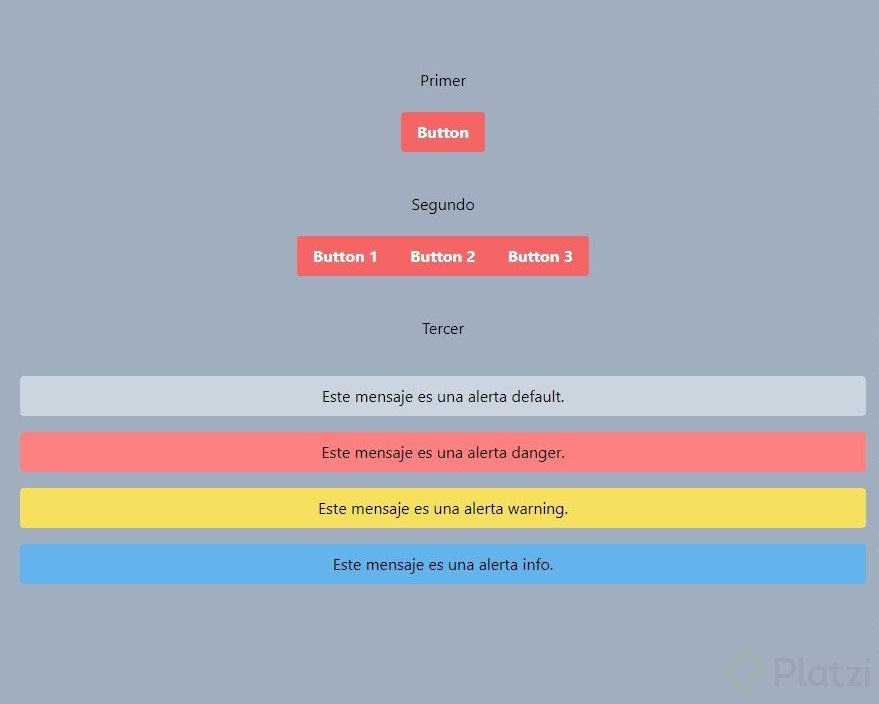
Este mensaje es una alerta warning.

</**div**>

<**div** class=“alert alert-info”>

Este mensaje es una alerta info.

</**div**>

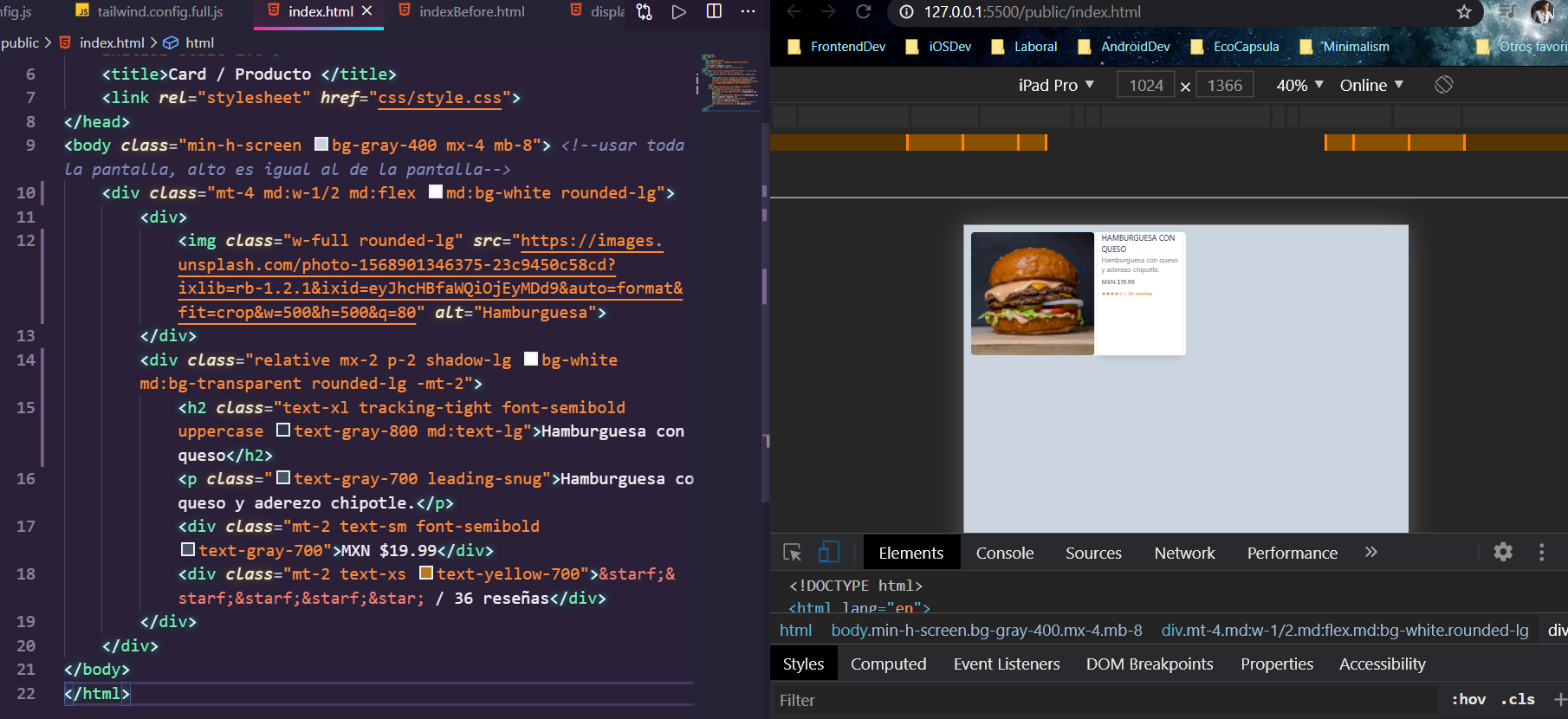


Proyecto : platzifood

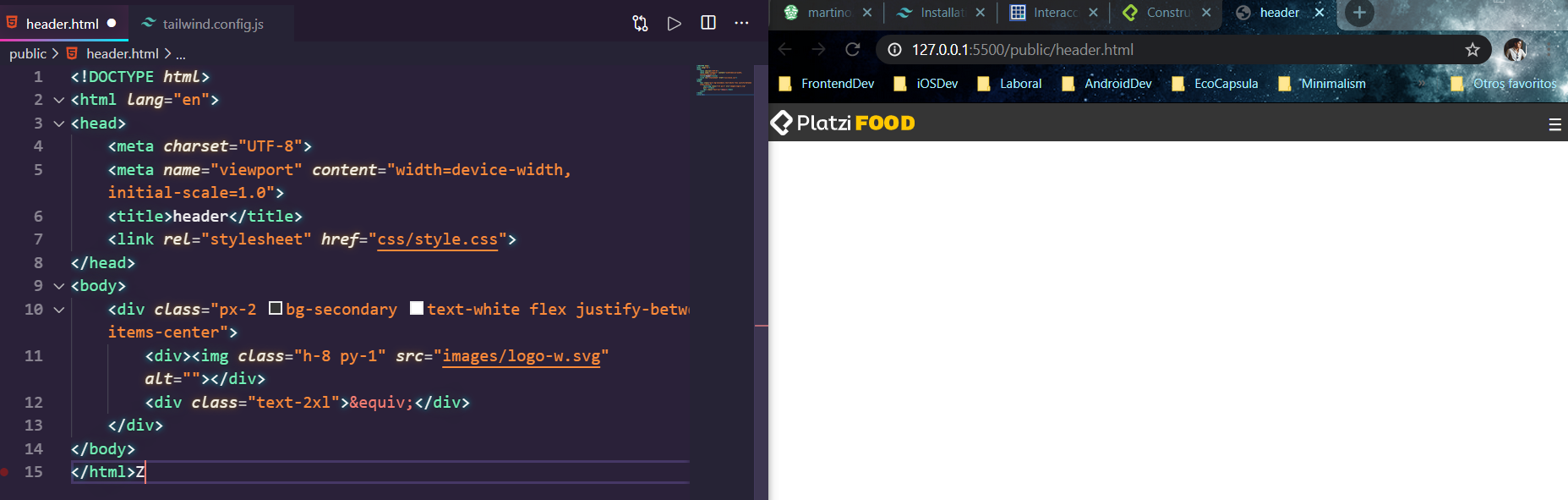
# creando una card, aplicando formato a la card



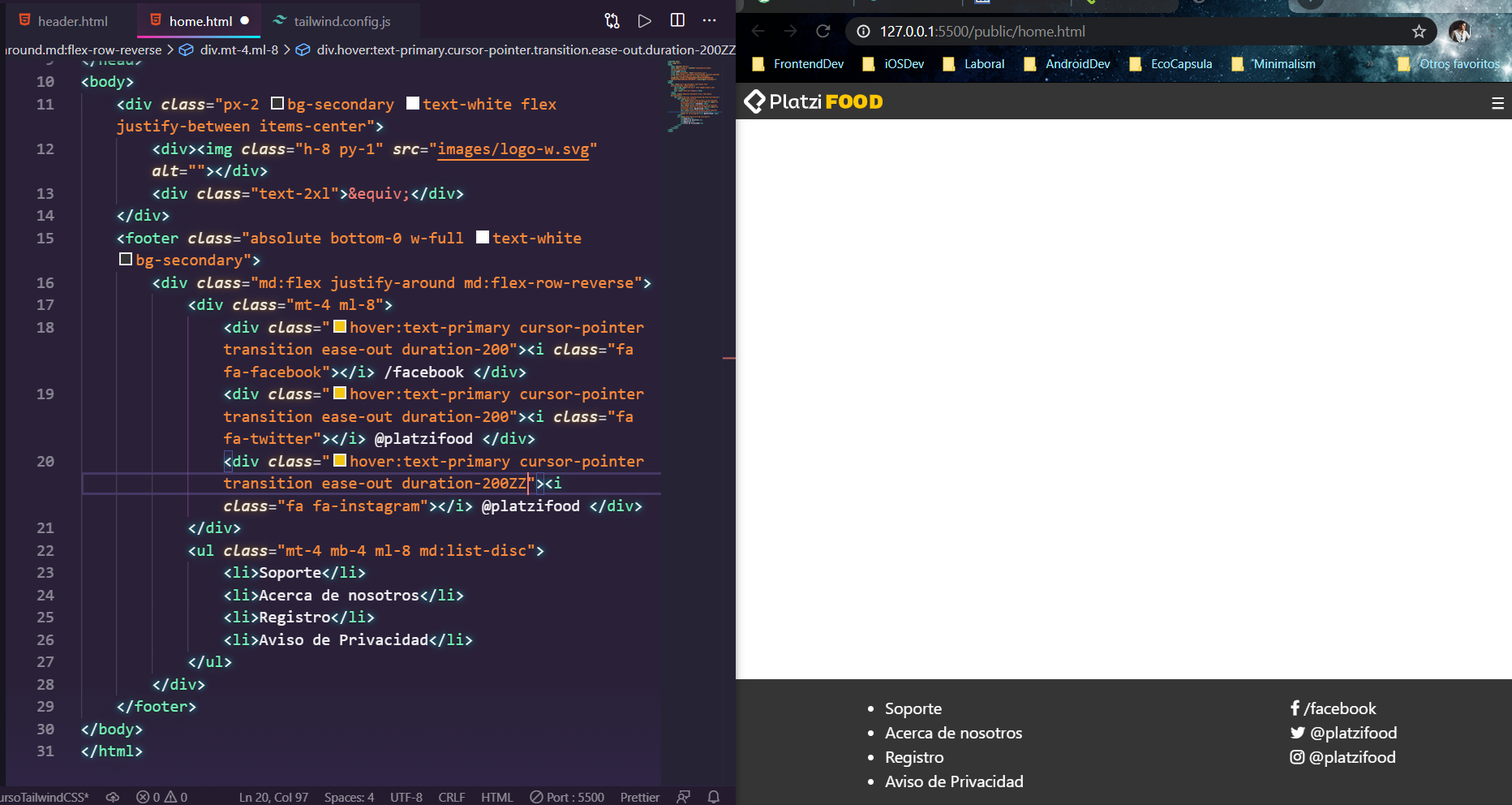
# Responsive design en la card



# construye el header



# crea el footer



Para agregarle un transition al hover pueden usar las siguientes etiquetas

<**div** **class**="hover:text-primary cursor-pointer transition ease-out duration-200" > <i **class**="fa fa-facebook"></i> Facebook </**div**>

<**div** **class**="hover:text-primary cursor-pointer transition ease-out duration-200" > <i **class**="fa fa-twitter"></i> Twitter </**div**>

<**div** **class**="hover:text-primary cursor-pointer transition ease-out duration-200" > <i **class**="fa fa-instagram"></i> Instagram </**div**>

Con la etiqueta **cursor-pointer** van a hacer que aparezca el dedo del click al hacer hover.  
Con la etiqueta **transition** van a hacer que el hover tenga una transición en las propiedades que cambian con el hover.  
Con la etiqueta **ease-out** van a colocar la velocidad de la animación de transición. Pueden ver toda la documentación en [www.tailwindcss.com/docs/transition-timing-function/#/app](https://platzi.com/clases/1822-tailwind-css/26616-crea-el-footer/www.tailwindcss.com/docs/transition-timing-function/#/app)  
Con la etiqueta **duration-200** pueden modificar el tiempo que dura la animación. Esta puede ser modificada dede el 200.