

Features, Tailwind y daisyUI

Organizar un proyecto Next.js y darle estilo

Desarrollo de Entornos Digitales Web — Universidad Icesi

Mapa de la sesión



Features

Carpetas por
funcionalidad y
consumo de API



Tailwind CSS

Utilidades, responsive y
estados



daisyUI

Componentes y temas



Actividad

Componentes con
daisyUI

Organizar por features

Lo que cambia junto vive junto

Por tipo o por feature

Por tipo

components/

LoginForm.tsx

UserCard.tsx

hooks/

useAuth.ts

useUsers.ts

services/

authService.ts

userService.ts

types/

auth.ts

user.ts

Por tipo: cada carpeta mezcla todas las pantallas.

Por tipo o por feature

Por tipo

4 carpetas abiertas

components/

LoginForm.tsx

UserCard.tsx

hooks/

useAuth.ts

useUsers.ts

services/

authService.ts

userService.ts

types/

auth.ts

user.ts

Cambiar el login obliga a abrir 4 carpetas y buscar entre archivos ajenos.

Por tipo o por feature

Por tipo

4 carpetas abiertas

components/

LoginForm.tsx

UserCard.tsx

hooks/

useAuth.ts

useUsers.ts

services/

authService.ts

userService.ts

types/

auth.ts

user.ts

Por feature

1 carpeta

features/auth/

LoginForm.tsx

useAuth.ts

auth.service.ts

types.ts

features/users/

UserCard.tsx

user.service.ts

shared/

api.ts

Button.tsx

Por feature: todo el login vive en una sola carpeta.

Una feature es una funcionalidad completa



auth

Inicio de sesión, registro y recuperación.



users

Lista de perfiles y detalle de usuario.



catalog

Listado, búsqueda y filtros de productos.

Cada una trae su interfaz, su estado, su llamada al backend y sus tipos.

app/ solo enruta

```
src/app/(auth)/login/page.tsx
import { LoginForm } from '@features/auth';

export const metadata = {
  title: 'Iniciar sesión',
};

export default function LoginPage() {
  return (
    <main className="p-4">
      <LoginForm />
    </main>
  );
}
```

/login

La URL vive en app/.

app/ solo enruta

```
src/app/(auth)/login/page.tsx
import { LoginForm } from '@features/auth';

export const metadata = {
  title: 'Iniciar sesión',
};

export default function LoginPage() {
  return (
    <main className="p-4">
      <LoginForm />
    </main>
  );
}
```

<LoginForm />

El formulario vive en features/auth/. La página solo lo conecta.

Anatomía de una feature

src/features/auth/

components/

la interfaz

LoginForm.tsx RegisterForm.tsx

Anatomía de una feature

src/features/auth/

components/

hooks/

la interfaz

LoginForm.tsx RegisterForm.tsx

estado y ciclo de vida

useAuth.ts

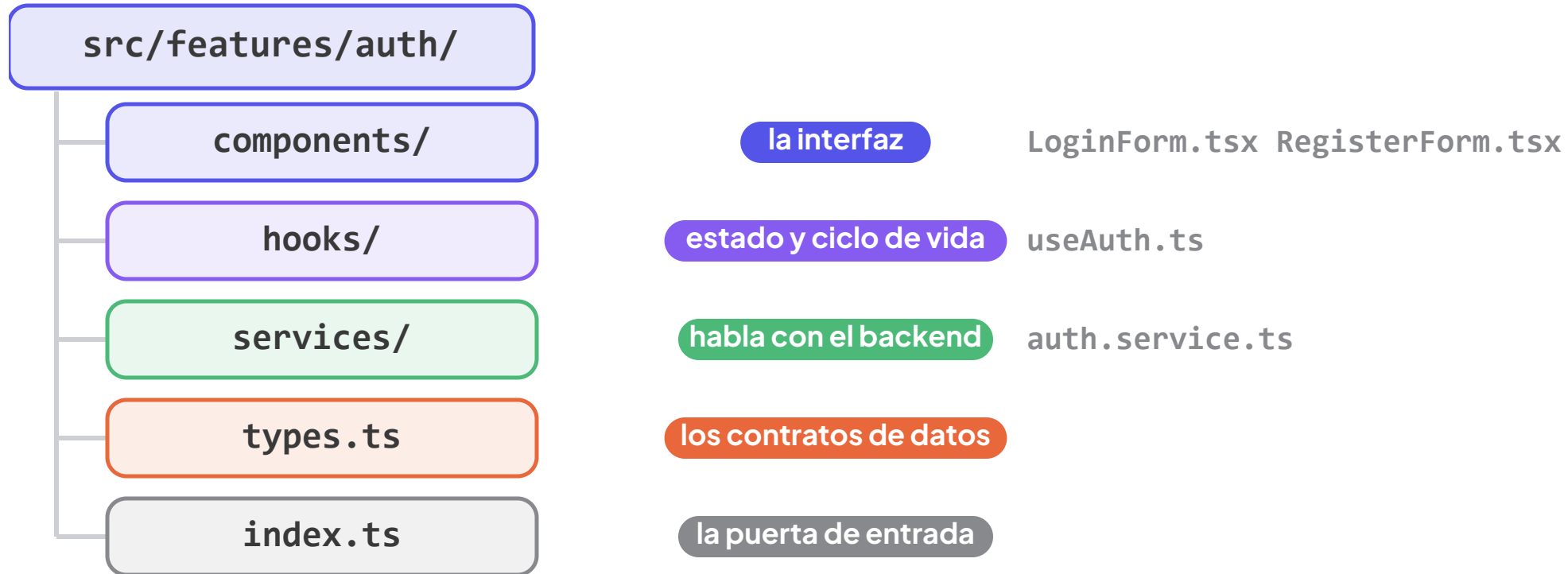
Anatomía de una feature



Anatomía de una feature



Anatomía de una feature



Cada carpeta tiene un único trabajo.

services: habla con el backend

src/features/auth/services/auth.service.ts

```
export const authService = {  
  async login(credentials: LoginInput) {  
    const token = await apiFetch<string>('/auth/login', {  
      method: 'POST',  
      body: JSON.stringify(credentials),  
    });  
    session.setToken(token);  
    return token;  
  },  
};
```

El service conoce la ruta y guarda el token. No sabe nada de la interfaz.

hooks: estado de carga y de error

```
src/features/auth/hooks/useAuth.ts
export function useAuth() {
  const [isLoading, setIsLoading] = useState(false);
  const [error, setError] = useState<string | null>(null);

  const login = async (credentials: LoginInput) => {
    setIsLoading(true);
    try {
      return await authService.login(credentials);
    } catch (err) {
      setError(err instanceof Error ? err.message : 'Error');
    } finally {
      setIsLoading(false);
    }
  };

  return { login, isLoading, error };
}
```

El hook entrega login, isLoading y error a la vista.

Shared o feature

features/auth/

LoginForm.tsx

useAuth.ts

PasswordRequirements.tsx

isValidPassword.ts

Solo le sirven al login:
se quedan en su feature.

Una cosa nace dentro de su feature.

Shared o feature

features/auth/

LoginForm.tsx

useAuth.ts

PasswordRequirements.tsx

isValidPassword.ts

Solo le sirven al login:
se quedan en su feature.

shared/

Button.tsx

Input.tsx

api.ts

useDebounce.ts

Regla de 2 o más

Una feature la usa: vive en la feature.
Dos o más la usan: vive en shared.

Si dos features independientes la necesitan, se promueve a shared.

Shared o feature

features/auth/

LoginForm.tsx

useAuth.ts

PasswordRequirements.tsx

isValidPassword.ts

Solo le sirven al login:
se quedan en su feature.

shared/

Button.tsx

Input.tsx

api.ts

useDebounce.ts

Regla de 2 o más

Una feature la usa: vive en la feature.
Dos o más la usan: vive en shared.

UserAvatar: Header y Perfil

se promueve

El camino de una petición

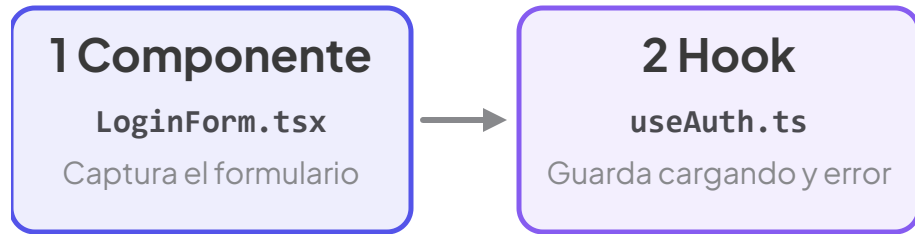
1 Componente

`LoginForm.tsx`

Captura el formulario

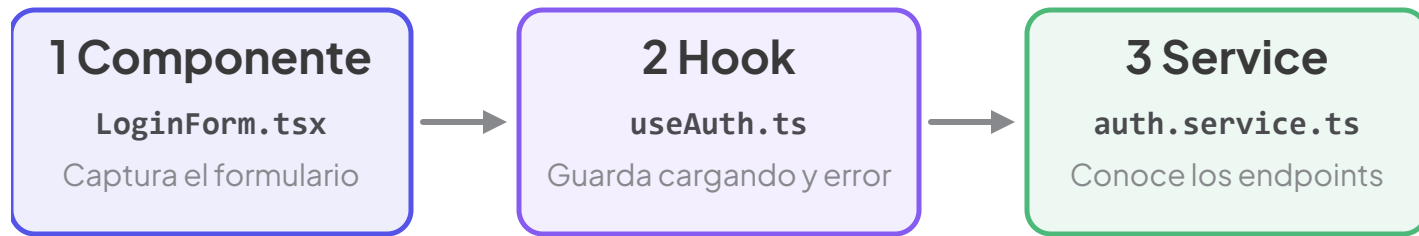
El componente solo sabe pintar y capturar el evento.

El camino de una petición



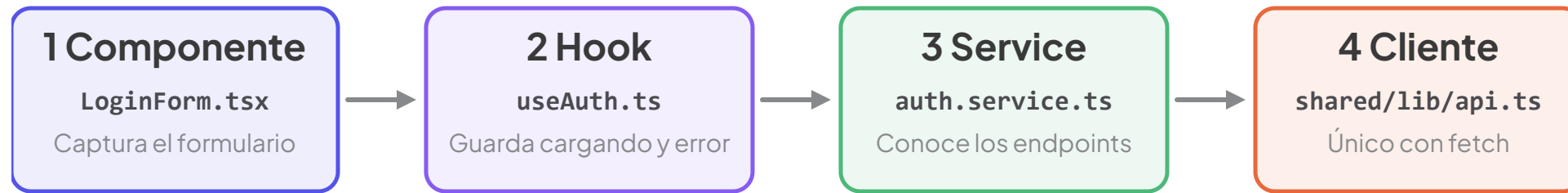
El hook guarda el estado de carga y de error.

El camino de una petición



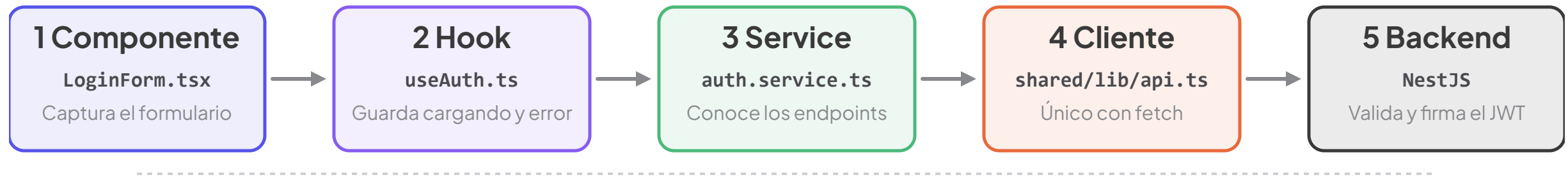
El service conoce las rutas de su feature.

El camino de una petición



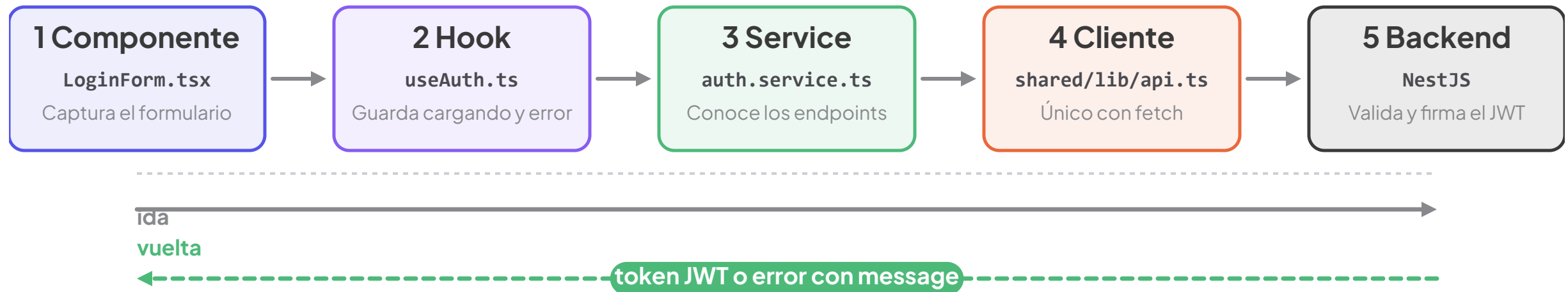
El cliente es el único lugar con fetch.

El camino de una petición



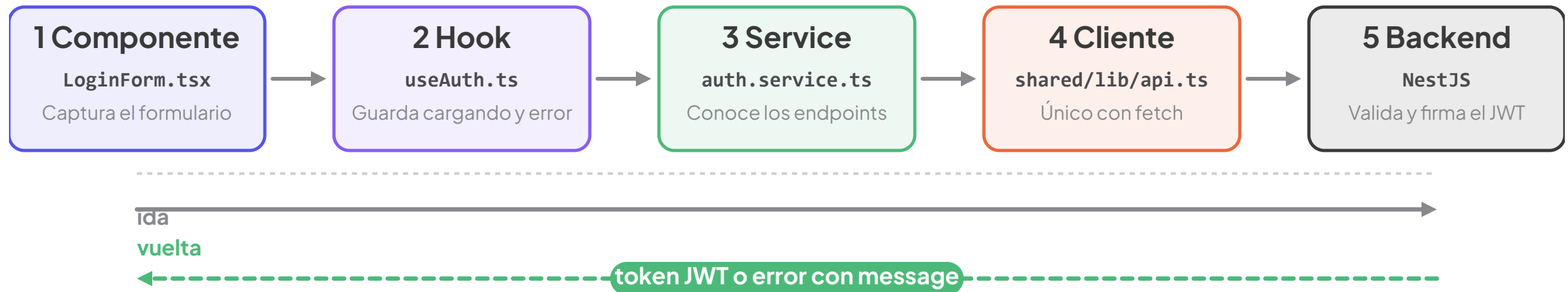
El backend responde con un token o con un error.

El camino de una petición



La respuesta vuelve por el mismo camino.

El camino de una petición



Regla: un componente nunca llama a fetch ni escribe la URL del backend.

shared/lib/api.ts: el único fetch

src/shared/lib/api.ts

```
const API_BASE_URL = process.env.NEXT_PUBLIC_API_URL ?? 'http://localhost:4000';

export async function apiFetch<T>(endpoint: string, options: RequestInit = {}): Promise<T> {
  const token = session.getToken();
  const headers = {
    'Content-Type': 'application/json',
    ...(token ? { Authorization: `Bearer ${token}` } : {}),
  };

  const response = await fetch(`${API_BASE_URL}${endpoint}`, { ...options, headers });

  if (!response.ok) throw new Error(`Error HTTP ${response.status}`);
  return response.json();
}
```

La URL viene de .env.local: NEXT_PUBLIC_API_URL=http://localhost:4000

La estructura completa

src/

app/

(auth)/login/page.tsx

(auth)/register/page.tsx

(app)/dashboard/page.tsx

features/

auth/

users/

shared/

lib/api.ts

components/ui/

solo rutas y layouts delgados

una carpeta por funcionalidad

components hooks services types.ts index.ts

components hooks services types.ts

lo que usan 2 o más features

Tailwind CSS v4

Estilos con clases utilitarias

Utility-first: clases en vez de CSS



Escala consistente

p-4 siempre es 16 px. Nada de valores sueltos.

```
p-4 mt-2 gap-6
```



Estados

Hover, focus y disabled como prefijos.

```
hover:bg-blue-700
```



Responsive

Cada prefijo aplica desde su ancho en adelante.

```
md:grid-cols-2
```

Tailwind v4: configuración en CSS

```
app/globals.css  
  
@import "tailwindcss";  
  
@theme {  
  --color-brand: #5454e9;  
}
```



Sin tailwind.config.js

La configuración vive en tu CSS.



Detecta tus archivos solo

No hay que listar rutas con content.



Nombres nuevos

bg-linear-to-r reemplaza a bg-gradient-to-r.

Espaciado: la escala de 4 px



p-1
4 px

Espaciado: la escala de 4 px



p-1
4 px



p-2
8 px

Espaciado: la escala de 4 px



p-1
4 px



p-2
8 px



p-4
16 px

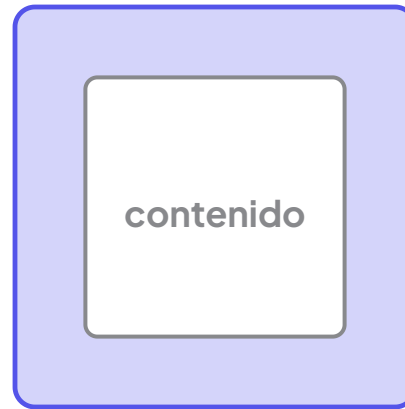
Espaciado: la escala de 4 px



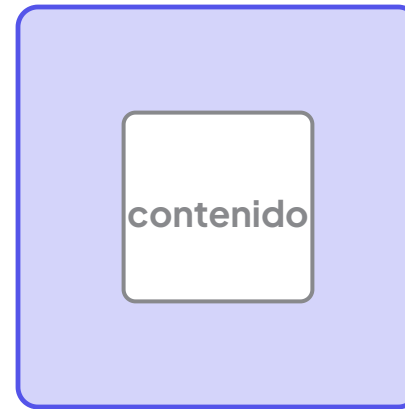
p-1
4 px



p-2
8 px



p-4
16 px



p-6
24 px

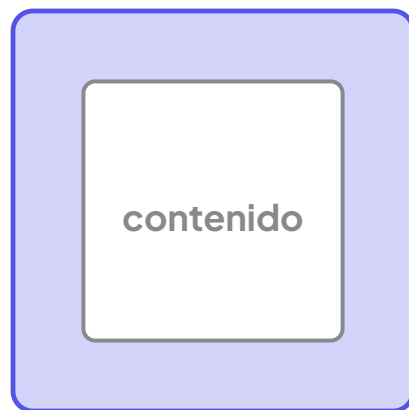
Espaciado: la escala de 4 px



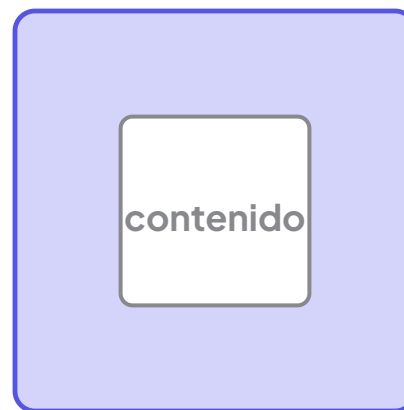
p-1
4 px



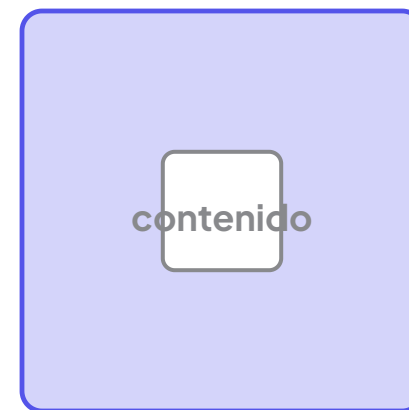
p-2
8 px



p-4
16 px



p-6
24 px



p-8
32 px

La escala va de 4 en 4 px: p-4 siempre es 16 px.

Padding, border y margin

Título y texto

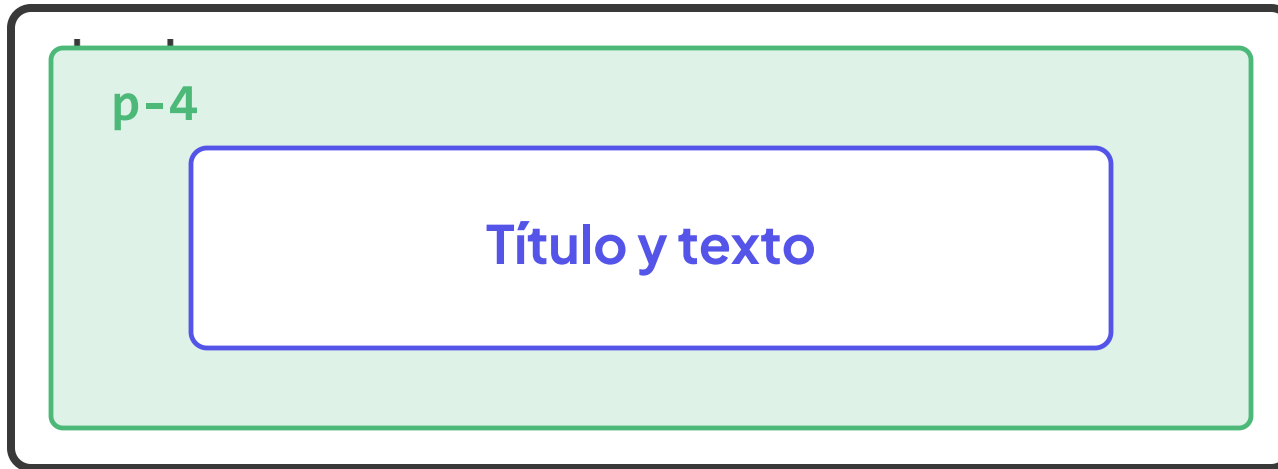
El contenido.

Padding, border y margin



padding (p-4): espacio dentro del borde.

Padding, border y margin



border: el borde del elemento.

Padding, border y margin



margin (m-4): espacio fuera, hacia los vecinos.

Dimensiones



64 px

w-48

192 px

w-1/2

50 % del padre

w-full

100 % del padre

Valores fijos: w-16. Relativos: w-1/2. Un tope: max-w-md. Altura mínima: min-h-screen.

Tipografía

Diseño web	<code>text-sm</code> • 14 px
Diseño web	<code>text-base</code> • 16 px
Diseño web	<code>text-xl</code> • 20 px
Diseño web	<code>text-2xl</code> • 24 px
Diseño web	<code>text-4xl</code> • 36 px

La escala tipográfica va de `text-sm` a `text-9xl`.

Tipografía

Diseño web con Tailwind

`font-normal`

Diseño web con Tailwind

`font-medium`

Diseño web con Tailwind

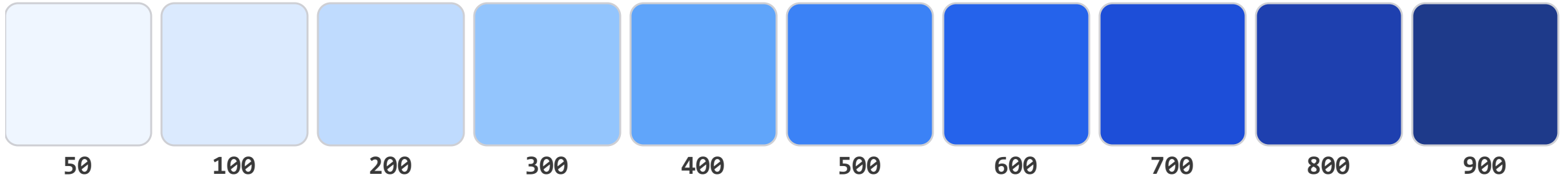
`font-semibold`

Diseño web con Tailwind

`font-bold`

El peso se cambia con `font-normal`, `font-medium`, `font-semibold` y `font-bold`.

Colores



blue-50 ... blue-900: de claro a oscuro

Cada color trae 10 tonos. Se combinan con un prefijo.

Colores

Ejemplo

bg-blue-600
fondo

Ejemplo

text-blue-700
texto

Ejemplo

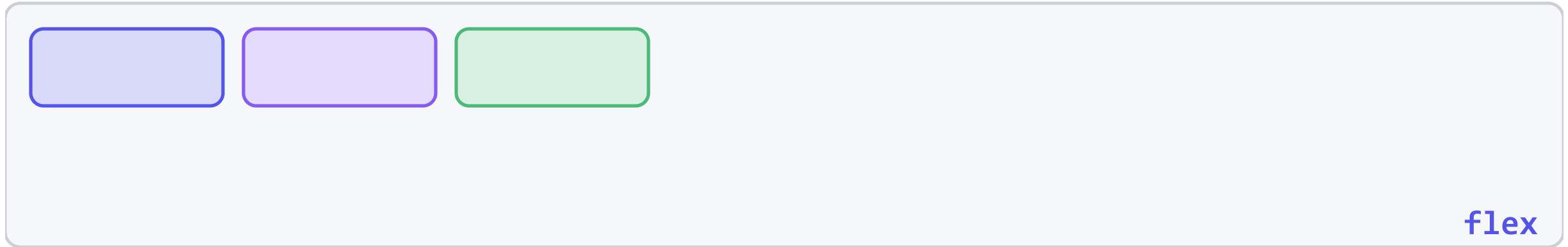
border-blue-300
borde

Ejemplo

bg-blue-600/40
opacidad

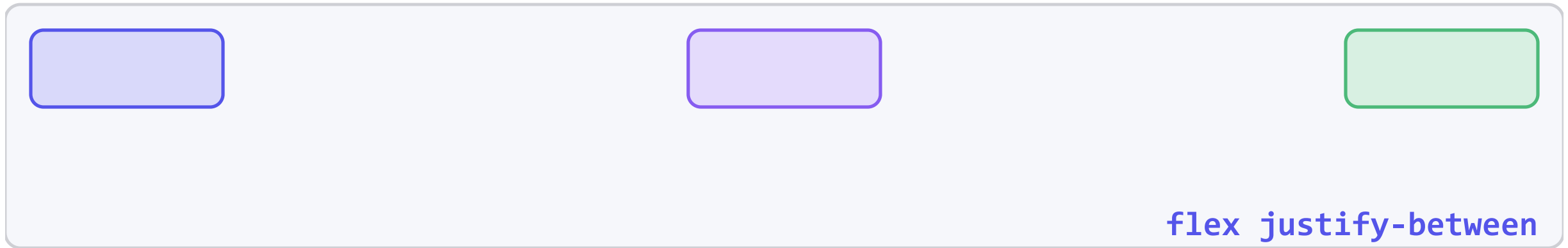
La barra /40 baja la opacidad del color sin tocar el resto.

Flexbox



flex: los hijos se ponen en fila.

Flexbox



justify-between reparte el espacio sobrante en el eje principal.

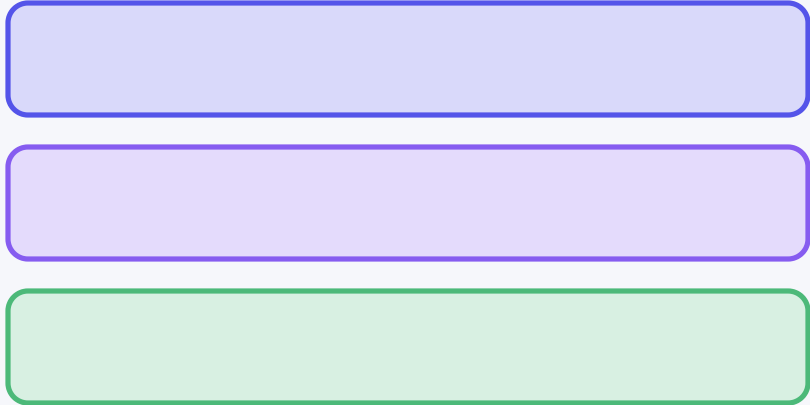
Flexbox



`flex items-center gap-4`

`items-center` alinea en el eje cruzado; `gap-4` deja 16 px entre hijos.

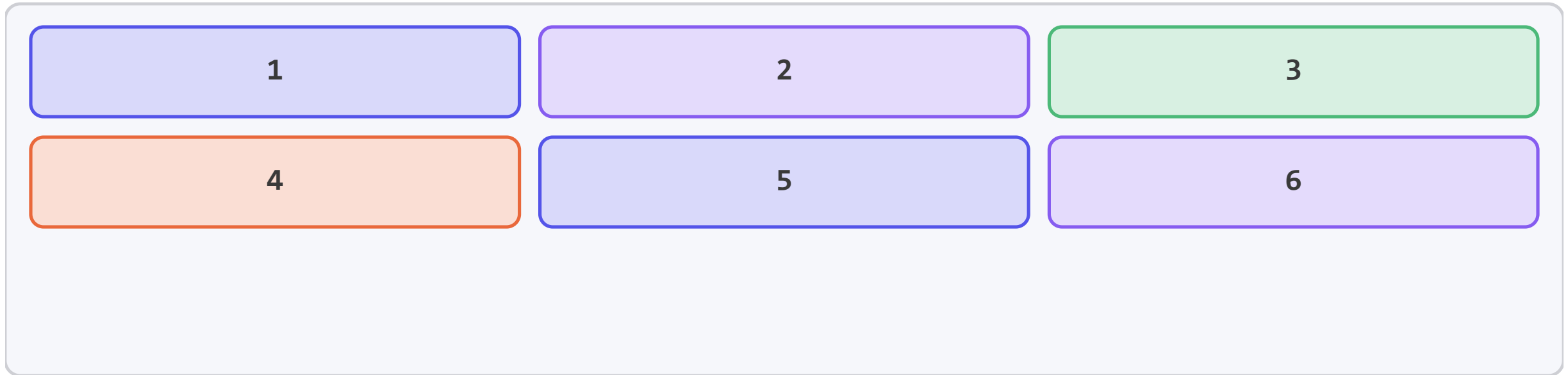
Flexbox



`flex flex-col gap-4`

flex-col cambia el eje principal: los hijos se apilan.

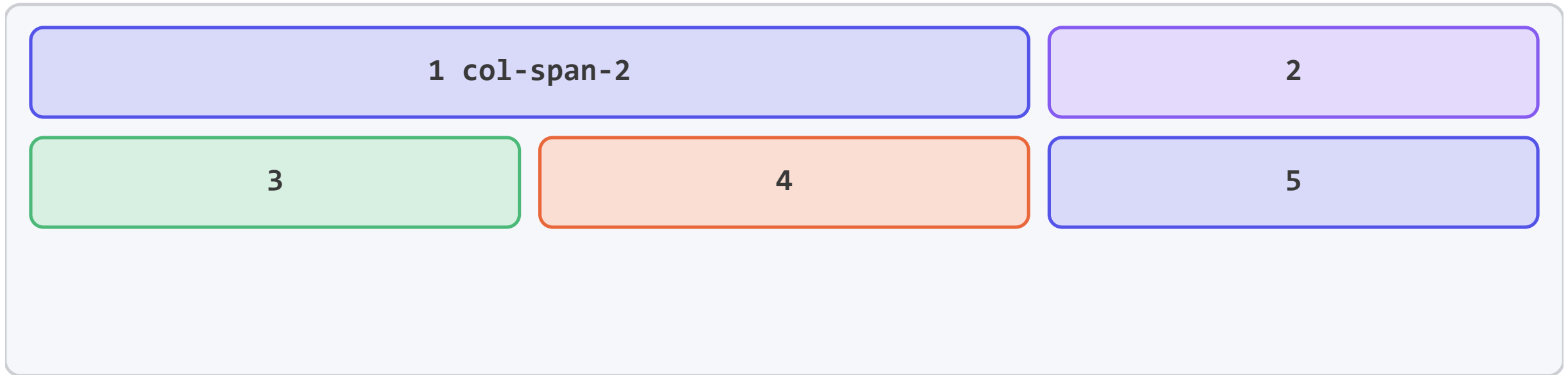
Grid



`grid grid-cols-3 gap-4`

`grid-cols-3` crea 3 columnas iguales.

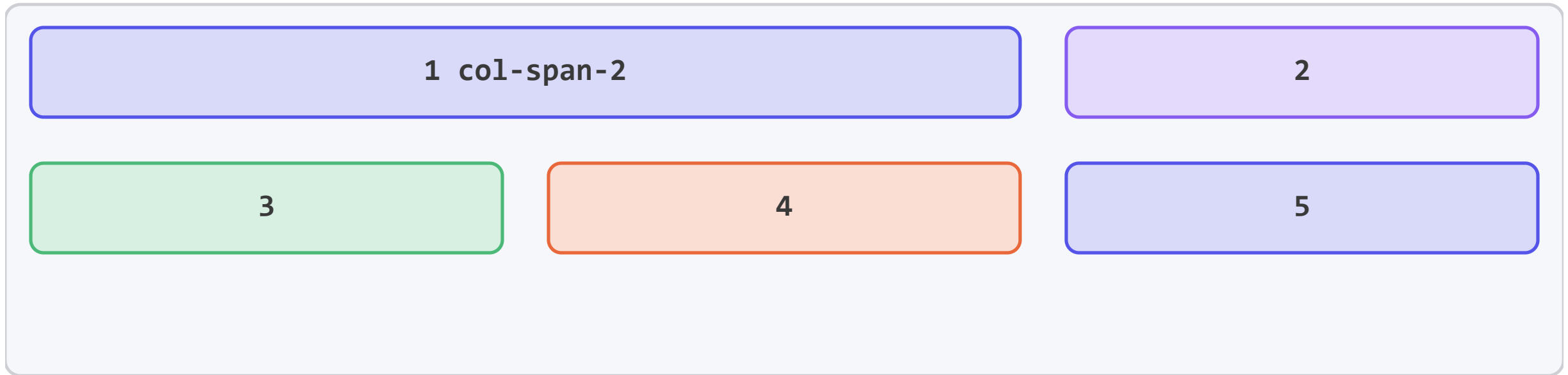
Grid



`grid grid-cols-3 gap-4`

`col-span-2` hace que un hijo ocupe 2 columnas.

Grid



`grid grid-cols-3 gap-9`

gap controla el espacio entre filas y columnas.

Posicionamiento



relative

en la tarjeta (el padre)

relative define el punto de referencia.

Posicionamiento



relative

en la tarjeta (el padre)

absolute top-2 right-2

en la insignia (el hijo)

absolute se mide desde el padre relative más cercano.

Estados con prefijos

Guardar

normal

`class="bg-blue-600"`

Sin prefijo: el estado normal.

Estados con prefijos

Guardar

normal

Guardar

hover

```
class="bg-blue-600 hover:bg-blue-700"
```

hover: solo cuando el cursor pasa por encima.

Estados con prefijos

Guardar

normal

Guardar

hover

Guardar

focus

```
class="bg-blue-600 hover:bg-blue-700 focus:ring-4 focus:ring-blue-300"
```

focus: cuando el teclado llega al botón.

Estados con prefijos

Guardar

normal

Guardar

hover

Guardar

focus

Guardar

disabled

```
class="bg-blue-600 hover:bg-blue-700 focus:ring-4 focus:ring-blue-300 disabled:opacity-50"
```

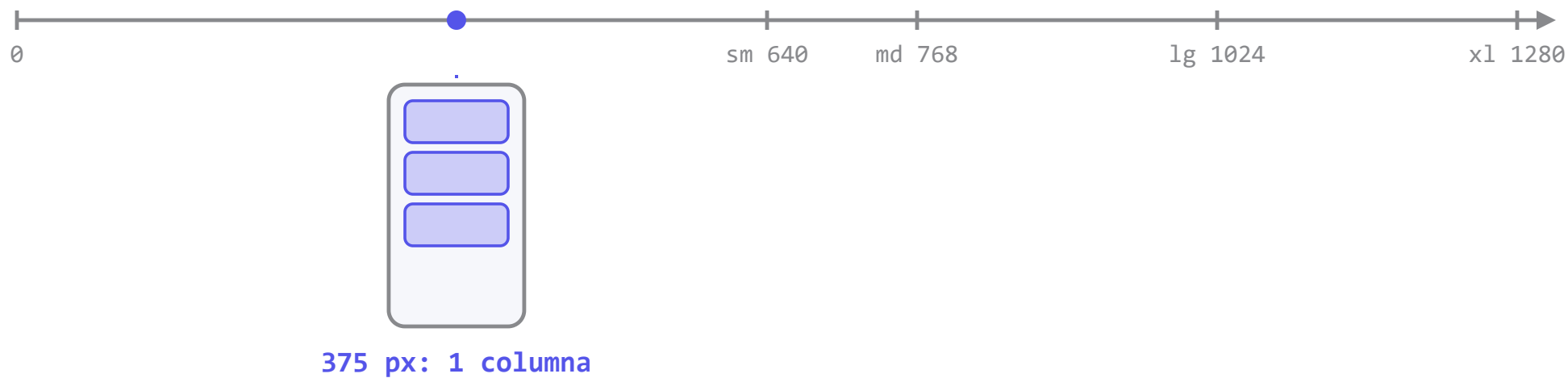
disabled: no se puede usar.

Mobile-first con breakpoints

grid-cols-1

gana

ancho de pantalla



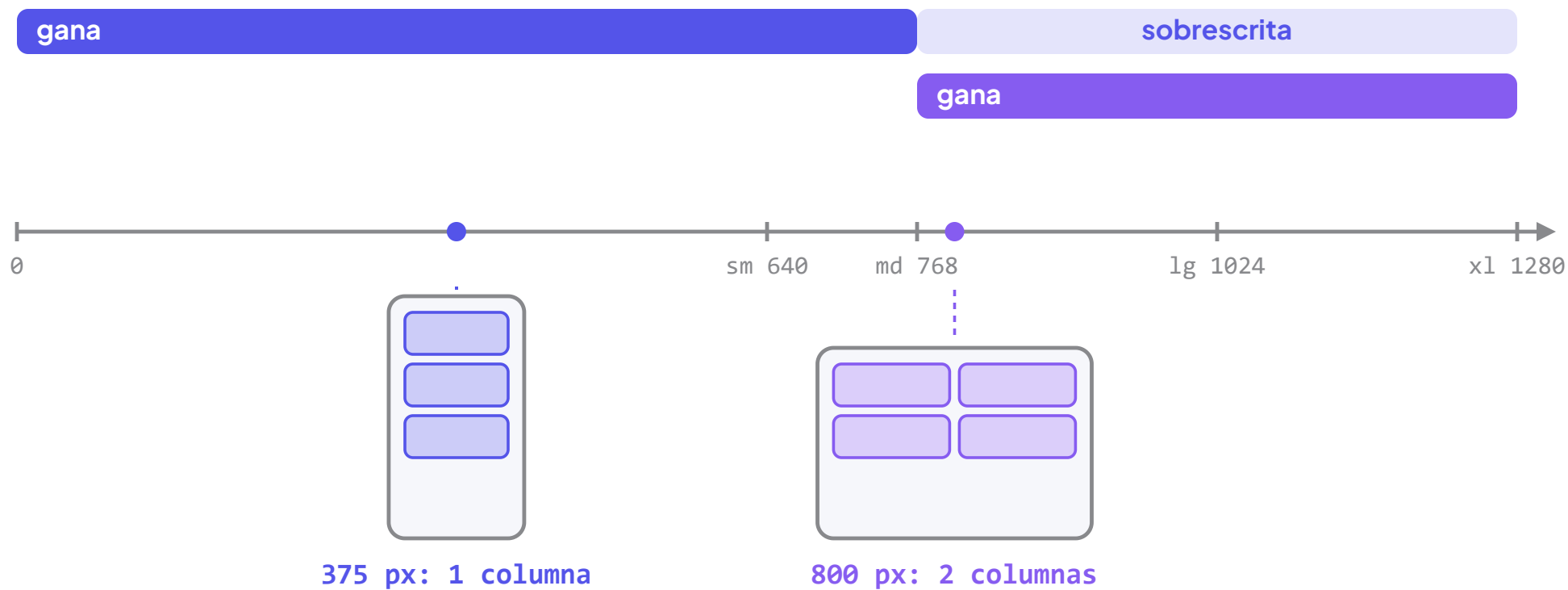
Sin prefijo: aplica desde 0 px. Una columna.

Mobile-first con breakpoints

grid-cols-1

md:grid-cols-2

ancho de pantalla



md: desde 768 px sobrescribe: dos columnas.

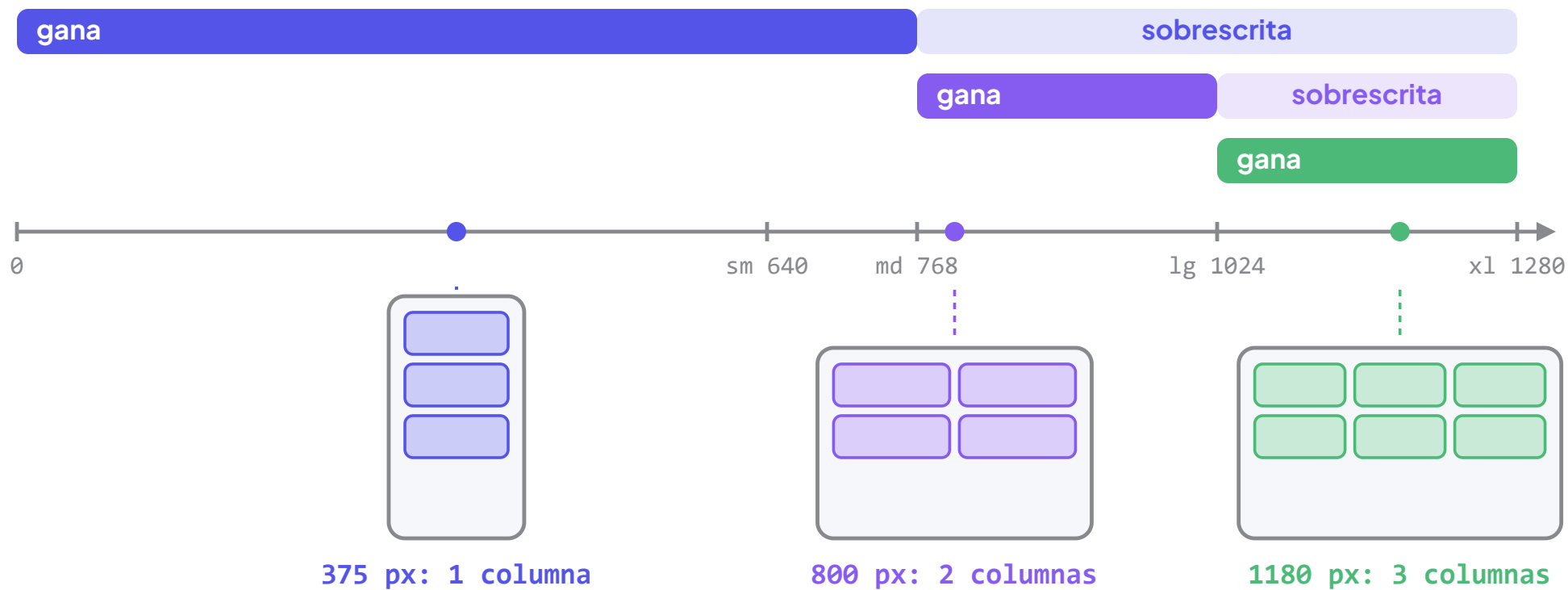
Mobile-first con breakpoints

`grid-cols-1`

`md:grid-cols-2`

`lg:grid-cols-3`

ancho de pantalla



lg: desde 1024 px sobrescribe de nuevo: tres columnas.

Valores a la medida

JSX

```
<div class="w-[320px] bg-[#5454e9] text-white">
```

Valores a la medida entre corchetes.

```
</div>
```

```
<button class="bg-brand hover:bg-brand/80">
```

Color propio desde @theme.

```
</button>
```

Los corchetes sirven para un valor puntual; @theme para uno que se repite.

daisyUI 5

Componentes listos sobre Tailwind

Ingredientes y platos preparados

Solo Tailwind

```
inline-flex items-center justify-center px-4 py-2 text-sm  
font-medium rounded-lg text-white bg-blue-600 hover:bg-blue-  
700 focus:ring-2
```

Guardar

Ingredientes y platos preparados

Solo Tailwind

```
inline-flex items-center justify-center px-4 py-2 text-sm  
font-medium rounded-lg text-white bg-blue-600 hover:bg-blue-  
700 focus:ring-2
```

Guardar

Con daisyUI

```
btn btn-primary
```

Guardar

daisyUI pone el nombre; Tailwind ajusta el layout: `btn btn-primary w-full md:w-auto`.

Instalación

Terminal

```
npm i -D daisyui@latest
```

src/app/globals.css

```
@import "tailwindcss";  
@plugin "daisyui";
```

Dos líneas en el CSS y las clases ya están disponibles.

Capas del sistema de estilos

Tailwind CSS v4

```
flex p-4 rounded-xl bg-blue-600 md:grid-cols-2
```

Base: las utilidades de Tailwind.

Capas del sistema de estilos

daisyUI 5

btn card navbar input badge alert menu

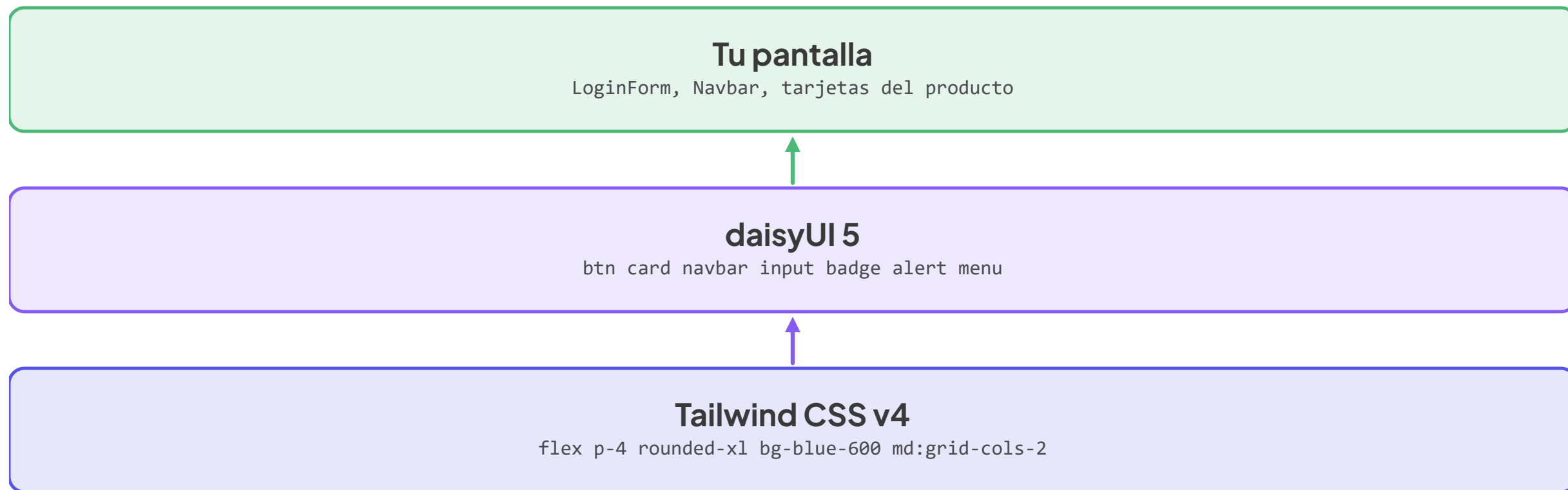


Tailwind CSS v4

flex p-4 rounded-xl bg-blue-600 md:grid-cols-2

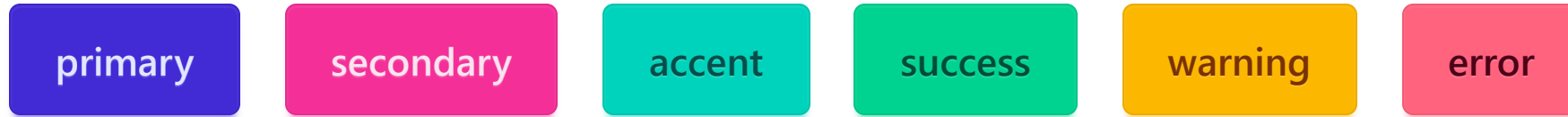
daisyUI agrupa utilidades en componentes con nombre.

Capas del sistema de estilos



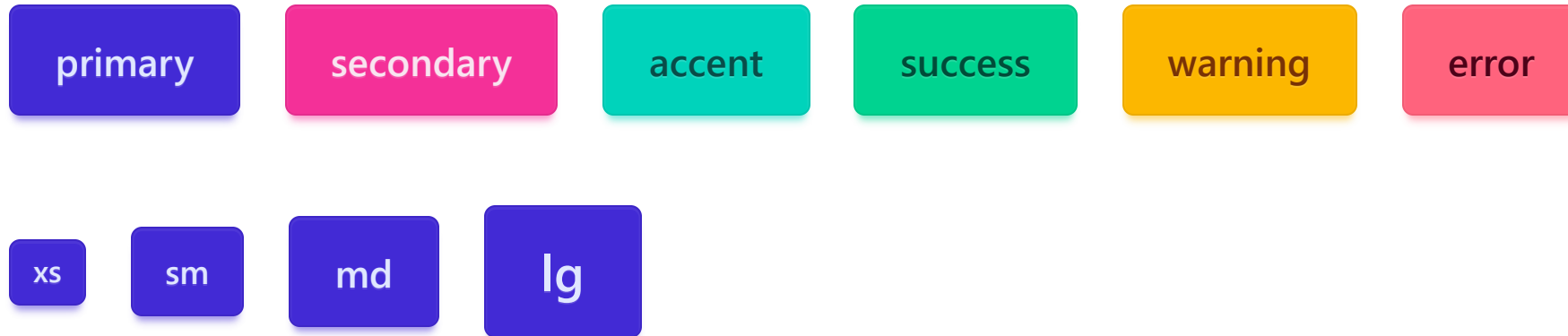
Tu pantalla combina las dos.

Botones



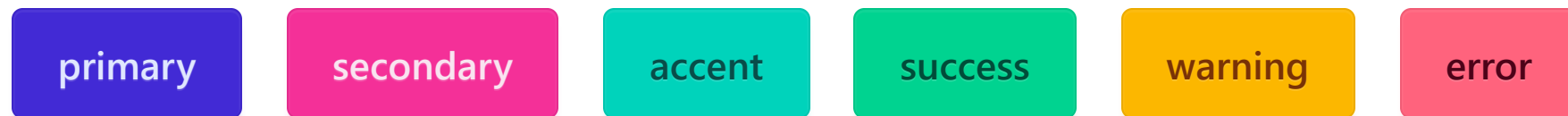
Color: btn-primary, btn-secondary, btn-accent...

Botones



Tamaño: btn-xs, btn-sm, btn-md, btn-lg.

Botones



Estilo: btn-outline, btn-soft, btn-ghost y spinner.

Tarjeta

imagen

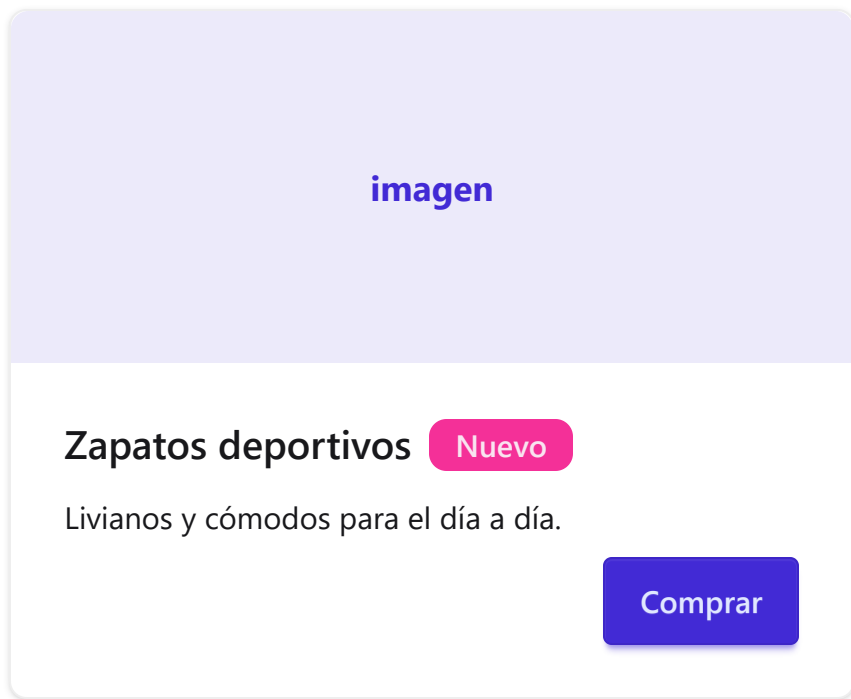
Zapatos deportivos **Nuevo**

Livianos y cómodos para el día a día.

Comprar

Una tarjeta completa con 6 clases de daisyUI.

Tarjeta



card

card-body

card-title

card-actions

card, card-body, card-title y card-actions: cada parte tiene su nombre.

Formulario

Iniciar sesión

Correo

Contraseña

En daisyUI 5 los formularios usan fieldset, label e input.



Formulario

Iniciar sesión

Correo

Contraseña

Mínimo 8 caracteres

Entrar

input-error marca el campo con problema.

Barra de navegación

MiApp

Inicio

Productos

Contacto

Entrar

Registro

navbar, navbar-start/center/end y menu forman la barra.

Barra de navegación

MiApp

Inicio

Productos

Contacto

Entrar

Registro

pantalla pequeña



MiApp

K

En móvil, el menú horizontal pasa a un botón con dropdown.

Mensajes con alert

Tu sesión expira en 5 minutos.

Registro completado.

Revisa los campos marcados.

Credenciales inválidas.

alert con variantes de color para cada tipo de mensaje.

Temas

`data-theme="light"`

Tema

El mismo marcado, otros colores.

primary

secondary

accent

Un atributo en la etiqueta html cambia el tema.

Temas

`data-theme="light"`

Tema

El mismo marcado, otros colores.

primary

secondary

accent

`data-theme="dark"`

Tema

El mismo marcado, otros colores.

primary

secondary

accent

dark: mismos componentes, otra paleta.

Temas

`data-theme="light"`

Tema

El mismo marcado, otros colores.

primary

secondary

accent

`data-theme="dark"`

Tema

El mismo marcado, otros colores.

primary

secondary

accent

`data-theme="cupcake"`

Tema

El mismo marcado, otros colores.

primary

secondary

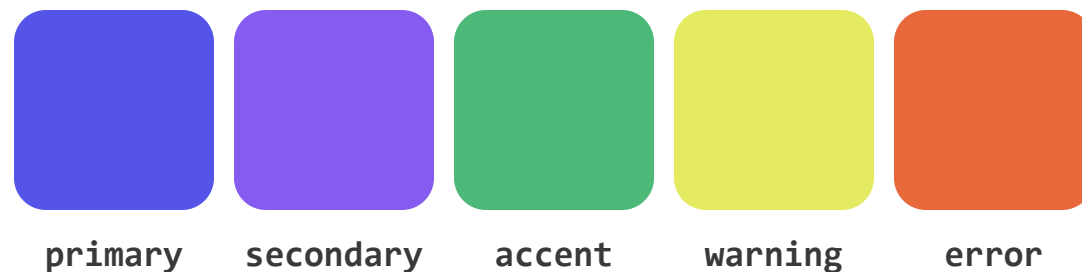
accent

Hay más de 30 temas incluidos.

Tema propio con la marca Icesi

app/globals.css

```
@plugin "daisyui/theme" {  
  name: "icesi";  
  default: true;  
  --color-primary: #5454e9;  
  --color-secondary: #865cf0;  
  --color-accent: #4cb979;  
  --color-warning: #e4eb60;  
  --color-error: #e9683b;  
}
```



Un tema propio con los colores de la marca Icesi.

daisyUI y Tailwind trabajan juntos

JSX

```
<button className="btn btn-primary w-full md:w-auto">  
  Guardar  
</button>  
  
<div className="card bg-base-100 shadow-sm p-6 max-w-md mx-auto">  
  ...  
</div>
```

La clase de daisyUI da el componente; las utilidades de Tailwind ajustan el diseño.

Cuidado: daisyUI 5 cambió



fieldset + label + input

La forma de armar formularios en la versión 5.



Ya no existen

input-bordered, form-control y label-text son de la versión 4.

Actividad

Componentes con daisyUI

Qué se entrega



7 componentes

2 átomos, 3 moléculas y
2 organismos del
catálogo.



Un archivo cada uno

En
shared/components/ui
o dentro de su feature.



Mobile-first

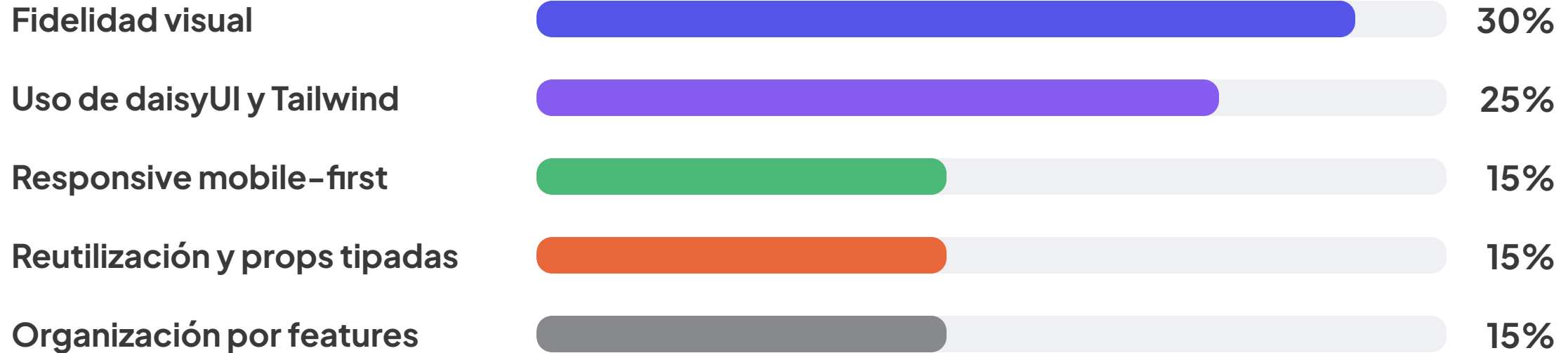
Debe verse bien desde
360 px.



Repositorio privado

Dentro de la
organización del curso.

Cómo se evalúa



Preguntas

Semana 12

Desarrollo de Entornos Digitales Web — Universidad Icesi