TALLER DE DISEÑO MULTIMEDIA
TALLER 2

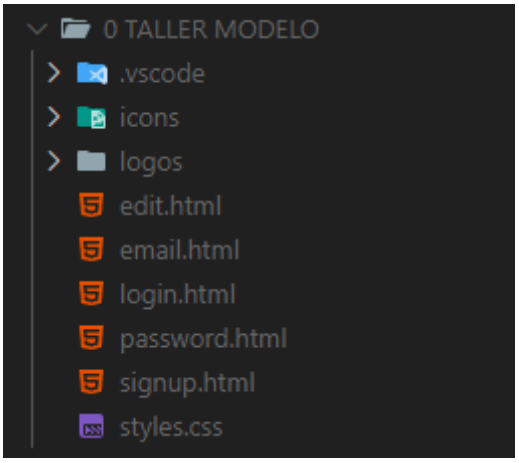
DESARROLLO FRONTEND USANDO HTML Y CSS - PARTE 2

OBJETIVO: Aplicar los conocimientos de HTML y CSS creando una página web sencilla para un negocio ficticio de ventas, enfocándose únicamente en el frontend y basando el diseño sobre uno previamente entregado.

1. Parámetros base

Continuamos trabajando sobre el taller anterior. Debemos tener la siguiente estructura mínima.

Nota: Si no tienes la carpeta .vscode, es normal, esta se genera automáticamente en algunas distribuciones, pero no hace falta para este taller.



Para recordar los estilos proporcionados, a continuación se especifica la paleta de colores junto con los estilos tipográficos. Recordemos que estos los debemos relacionar en el :root y body.

E5252E	FFFFFF	000000	C7C7C7	F7F7F7
Rojo	White	Black	Silver	Seasalt

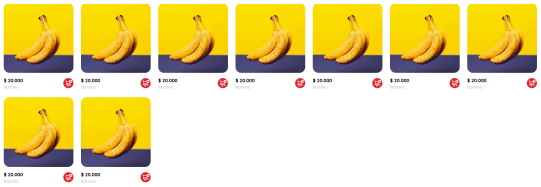
- **Fuente base:** Quicksand, sans-serif
- **Tamaños de fuente:**
  - **SM:** 14px
  - **MD:** 16px
  - **LG:** 18px

2. Componente principal

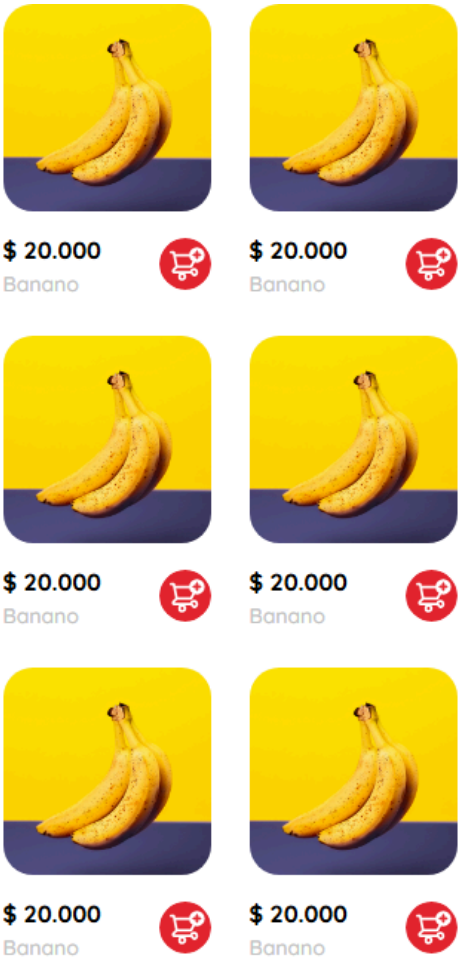
Para efectos de este taller, cada vista a partir de este punto no va a ser una página entera, sino **un componente** que luego podremos reutilizar y juntar a través de librerías como React, Angular y Vue.

Habiendo dicho esto, vamos a hacer el componente de la vista principal (main.html), en el cual vamos a la maquetación usando grid. Debemos tener en cuenta que esta vista debe ser totalmente responsive, acomodando los elementos según el tamaño de la pantalla.

Escritorio



Mobile



Lo primero que debemos hacer es crear el contenedor general de la sección (main-container) y, dentro de este, el contenedor de cada una de las tarjetas (cards-container).

```
1  <section class="main-container">
2    <div class="cards-container">
3
4    </div>
5  </section>
```

Luego, debemos crear las tarjetas de los productos como tal. Aquí debemos de tener en cuenta el contenedor de los elementos de la tarjeta (product-card).

Cada producto, según indicaciones del backend, tiene una imagen y luego una sección de información, donde contiene precio y nombre. Adicionalmente, cada tarjeta tiene un botón de compra, el cual se encuentra en nuestra carpeta de icons.

```html
<div class="product-card">
  <img src="https://images.pexels.com/photos/2875814/pexels-photo-2875814.jpeg" alt="">
  <div class="product-info">
    <div>
      <p>$ 20.000</p>
      <p>Banano</p>
    </div>
    <figure>
      <img src="./icons/bt_add_to_cart.svg" alt="">
    </figure>
  </div>
</div>
```

Asegúrate de buscar una imagen propia para tus productos. Puedes usar cualquier imagen que tenga una URL pública, como por ejemplo las imágenes gratuitas de Pexels.

Una vez tengas la tarjeta, asegúrate de duplicarla unas 10-15 veces para poder verificar el comportamiento del responsive. Esto se hará de manera automática cuando pasemos a ver React, Angular y Vue, extrayendo los datos desde el backend, pero, de momento, lo realizaremos manual

Primero, vamos a estilizar el contenedor, el cual va a ser un grid de tamaño automático.

El tamaño de las tarjetas lo asignamos en una variable, la cual debes crear en el :root como se realizó con las demás.

```css
.cards-container {
  display: grid;
  grid-template-columns: repeat(auto-fill, var(--card-size));
  gap: 26px;
  place-content: center;
}

.product-card {
  width: var(--card-size);
}

.product-card img {
  width: var(--card-size);
  height: var(--card-size);
  border-radius: 20px;
  object-fit: cover;
}
```

Segundo, vamos a estilizar la información de los productos. Para esto vamos a usar flex. Nótese cómo podemos usar flex y grid en conjunto para lograr resultados ordenados.

De igual manera, vemos como no es necesario que añadamos clases e identificadores a todos los elementos de la tarjeta; con usar selectores de descendencia es suficiente.

Finalmente, para colocar el precio y el nombre del producto usamos la pseudoclase :nth-child(n) que nos permite seleccionar el hijo número n. En este caso, el hijo 1 es el precio y el hijo 2 es el nombre del producto.

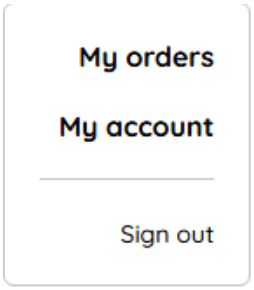
```css
.product-info {
  display: flex;
  justify-content: space-between;
  align-items: center;
  margin-top: 12px;
}

.product-info figure {
  margin: 0;
}

.product-info figure img {
  width: 35px;
  height: 35px;
}

.product-info div p:nth-child(1) {
  font-weight: bold;
  font-size: var(--md);
  margin-top: 0;
  margin-bottom: 4px;
}

.product-info div p:nth-child(2) {
  font-size: var(--sm);
  margin-top: 0;
  margin-bottom: 0;
  color: var(--very-light-pink);
}
```

Lo último que nos resta es realizar el responsive de tal forma que, en tamaños de pantalla inferiores a 640px, nuestra variable --card-size (o el nombre que hayas usado en el :root) cambie a 140px. Podemos ver cómo es posible cambiar directamente variables en ciertos tamaños para no tener que cambiar varios estilos.

```css
@media (max-width: 640px) {
  :root {
    --card-size: 140px;
  }
}
```

3. Componente de menú (versión escritorio)

Para este componente, el diseñador nos entrega la siguiente vista. Se trata de un componente alineado a la parte superior izquierda de la pantalla. No es más.



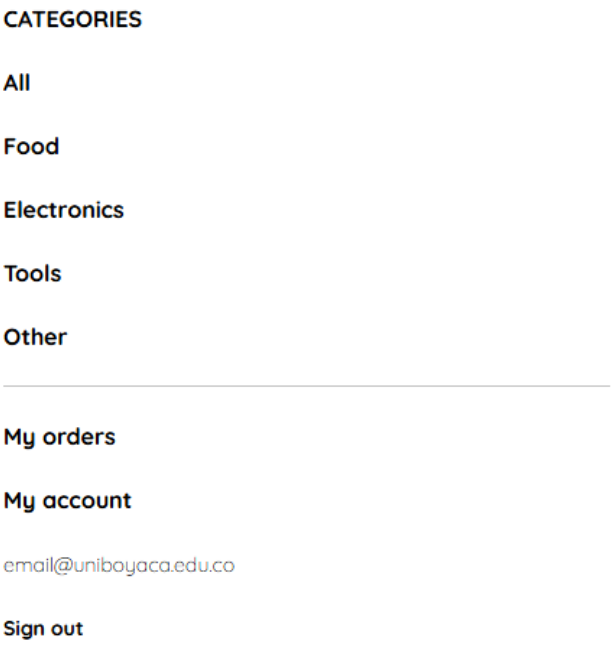
Crea un archivo menu-desktop.html.

Tip: Utiliza una etiqueta <ul> con sus respectivos <li> para hacer cada elemento del menú. Cada <li> va a tener una etiqueta <a> correspondiente al texto de cada elemento. Fíjate que el contenedor completo tiene un borde. Puedes usar la pseudoclase :last-child y :not(:last-child) para añadir los estilos de los elementos.

Puedes realizar este componente con 7 reglas de estilos o menos.

4. Componente de menú (versión mobile)

Para este componente, el diseñador nos entrega la siguiente vista. Se trata de un componente que se ajusta al tamaño horizontal de la pantalla.



Crea un archivo menu-mobile.html. Los realizamos en componentes diferentes porque, para efectos del responsive usando librerías, es más sencillo decir que en ciertos tamaños de pantalla utilizaremos menu-desktop.html y, en otros, menu-mobile.html. A veces el responsive es más eficiente usando componentes separados que tener que estilizar decenas de estilos para cada vista.

Tip: Sigue la misma lógica para crear el html que con menu-desktop.html, pero utiliza varias etiquetas <ul>: una para las categorías, otra para las cosas de la cuenta y quizás una última para el email y el botón de “Sign out”. Puedes usar la pseudoclase :nth-child(1) para añadir el borde inferior a la lista de las categorías.

Puedes realizar este componente con 7 reglas de estilos o menos.

5. Reto: Reorganización

Este reto es sencillo: organizar el proyecto. Los 3 archivos que creamos en este taller son componentes, por ende, para mejor organización, deberíamos separarlos en una carpeta llamada components.

Crea la nueva carpeta y coloca aquí dentro los componentes que hemos creado. La estructura final de nuestro proyecto debe ser: