

“UNIVERSIDAD PRIVADA DEL VALLE”
DEPARTAMENTO SISTEMAS Y TECNOLOGIA INFORMATICA
INGENIERÍA DE SISTEMAS INFORMÁTICOS



PROGRAMACIÓN WEB I
INFORME TECNICO

ESTUDIANTE: Luis Fernando Flores Martinez

DOCENTE: Ing. Henry Miranda Ordonez

La Paz - Bolivia

2025

Índice

1. INICIALIZADOR PÁGINAS WEB	1
2. CUERPO PRINCIPAL	2
3. ÁREA DEL LOGO	2
4. BOTÓN HAMBURGUESA	3
5. ÁREA DE BÚSQUEDA.....	3
6. REGISTRO	4
7. NAV-PRINCIPAL.....	4
8. TEXTO INFORMATIVO	5
9. BANNER PRINCIPAL	5
10. SECCIÓN DE TIENDA Y TÍTULOS	6
11. GRID DE CATEGORÍAS / TARJETAS	7
12. FOOTER	7
13. ENLACE AL SCRIPT Y CIERRE	9
14. BODY.....	10
15. ENCABEZADO GENERAL.....	10
16. CONTENEDOR SUPERIOR DEL HEADER.....	11
17. ÁREA DEL LOGO	12
18. LOGO	12
19. TEXTO DEL LOGO	13
20. AREA DE BÚSQUEDA.....	13
21. INPUT DE BÚSQUEDA.....	13
21. BOTÓN BÚSQUEDA.....	14
22. ENLACE "REGISTRO" DEL USUARIO	14
23. HOVER.....	15
24. NAV PRINCIPAL.....	15
25. HAMBURGUESA	16
26. MEDIA QUERY MÓVIL.....	16
26. Burguea bloque.....	16
27. NAV PRINCIPAL.....	16
28. NAV OPEN	18
29. NAV PRINCIPAL.....	18
30. TITULOS	19

31. TÍTULOS	20
31. BANNER	21
32. TARJETA	24
33. BLOQUE: DEBOUNCE	27
34. MENÚ MÓVIL	27
35. BUSCADOR DE PRODUCTOS	29
36. BLOQUE: SCROLL REVEAL	31

INFORME TÉCNICO

1. INICIALIZADOR PÁGINAS WEB

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width,initial-scale=1" />
  <title>Equipos - Ensayos de Materiales</title>
  <link rel="stylesheet" href="styles.css" />
</head>
```

Las presentes líneas nos sirven de manera para:

1. <!DOCTYPE html>
 - Indica al navegador que el documento usa HTML.
2. <html lang="es">
 - Elemento raíz del documento HTML. El atributo lang="es" informa a navegadores y tecnologías de asistencia lectores de pantalla que el contenido está en español
3. <head>
 - Contenedor de metadatos.
4. <meta charset="utf-8" />
 - Define la codificación de caracteres como UTF-8 soporta signos acentuados, caracteres especiales.
5. <meta name="viewport" content="width=device-width,initial-scale=1" />
 - Hace la página responsive: le indica al navegador móvil que el ancho de la ventana de visualización debe ser el ancho del dispositivo.
6. <title>Equipos - Ensayos de Materiales</title>
 - Título que aparece en la pestaña del navegador y resultados de búsqueda.
7. <link rel="stylesheet" href="styles.css" />
 - Enlaza tu archivo CSS (styles.css). El navegador carga este CSS para aplicar estilos.

8. </head>
 - Cierra el head.

2. CUERPO PRINCIPAL

```
<body>
<header class="encabezado">
  <div class="encabezado-superior">
```

Las presentes líneas nos sirven de manera para:

1. <body>
 - Contenedor del contenido visible de la página.
2. <header class="encabezado">
 - Elemento semántico que agrupa la cabecera principal del sitio logo, búsqueda, menú, avisos.
3. <div class="encabezado-superior">
 - División interna para organizar la fila superior logo, botón burger, búsqueda, usuario.

3. ÁREA DEL LOGO

```
<div class="area-logo">
  <a href="index.html"></a>
  <div class="texto-logo">Construction Materials Testing Equipment</div>
</div>
```

Las presentes líneas nos sirven de manera para:

1. <div class="area-logo">
 - Agrupa logo y texto.
2.
 - Enlace al inicio suele ser index.html. Acceso desde el logo.
3.

- Imagen del logo. alt="Logo": texto alternativo que describe la imagen
 - class="logo" para estilos.
4. <div class="texto-logo">Construction Materials Testing Equipment</div>
- Texto descriptivo al lado del logo; en tu CSS lo colocas con borde izquierdo rojo.

4. BOTÓN HAMBURGUESA

```
<button class="btn-burger">☰</button>
```

Las presentes líneas nos sirven de manera para:

1. <button class="btn-burger">☰</button>
 - Botón que muestra el icono de menú (☰).
 - En tu JS añades un listener a .btn-burger para abrir y cerrar .nav-principal.

5. ÁREA DE BÚSQUEDA

```
<div class="area-busqueda">
  <input class="input-busqueda" type="search" placeholder="Buscar nombre,
  ASTM o SKU">
  <button class="btn-buscar">🔍</button>
</div>
```

Las presentes líneas nos sirven de manera para:

1. <div class="area-busqueda">
 - Contenedor para el input y el botón de búsqueda; centras/ajustas con CSS.
2. <input class="input-busqueda" type="search" placeholder="Buscar nombre,
 ASTM o SKU">

- Campo de búsqueda. type="search" puede ofrecer pequeñas optimizaciones en móviles botón de borrar, teclado adecuado.
 - placeholder muestra texto guía. Si quieres accesibilidad completa, añade aria-label o un <label> oculto.
3. <button class="btn-buscar">  </button>
 - Botón que activa la búsqueda. En JS actualmente escuchas input para filtrar; este botón puede disparar búsqueda explícita.

6. REGISTRO

```
<div class="area-usuario">
  <a href="registro.html">Registro</a>
</div>
```

Las presentes líneas nos sirven de manera para:

1. <div class="area-usuario">
 - Contenedor para enlaces de usuario Registro.
2. Registro
 - Enlace a la página de registro.

7. NAV-PRINCIPAL

```
</div>
<nav class="nav-principal">
  <a href="suelos.html">Suelos</a>
  <a href="agregados.html">Agregados</a>
  <a href="laboratorio_suelos.html">Laboratorio Suelos</a>
  <a href="mecanicasuelos.html">Mecánica Suelos</a>
  <a href="concreto.html">Concreto</a>
  <a href="cemento.html">Cemento</a>
  <a href="asfalto.html">Asfalto</a>
  <a href="laboratorio.html">Laboratorio</a>
```

```
<a href="productos.html">Productos</a>
</nav>
```

Las presentes líneas nos sirven de manera para:

1. <nav class="nav-principal">
 - Elemento semántico para navegación principal. Contiene los enlaces de categoría. nav es importante para motores de búsqueda y lectores de pantalla.
2. Cada Texto
 - Enlaces de navegación a secciones/subpáginas.
3. </nav>
 - Cierra la navegación.

8. TEXTO INFORMATIVO

```
<div class="aviso">¿No encuentra lo que busca? <strong>Llámenos al  
1.800.544.7220</strong></div>
```

```
</header>
```

Las presentes líneas nos sirven de manera para:

1. <div class="aviso">...</div>
 - Mensaje corto con llamada a la acción. resalta el número telefónico en negrilla.

9. BANNER PRINCIPAL

```
<section class="banner">
  <h1>Materiales de Construcción</h1>
  <h2>Equipos de Ensayo</h2>
</section>
```

Las presentes líneas nos sirven de manera para:

1. <section class="banner">
 - o Sección hero con imagen de fondo se encuentra definida en CSS.
.banner en tu CSS tiene background: url("Imagenes/Medio.png").
2. <h1>Materiales de Construcción</h1>
 - o Título principal de la página.
3. <h2>Equipos de Ensayo</h2>
 - o Subtítulo del hero.
4. </section>
 - o Cierre del banner.

10. SECCIÓN DE TIENDA Y TÍTULOS

```
<main class="principal">
  <section class="seccion-titulos">
    <div class="linea-roja"></div>
    <h3>Tienda</h3>
    <p class="subtitulo">Equipos de Ensayos de Materiales</p>
  </section>
```

Las presentes líneas nos sirven de manera para:

1. <main class="principal">
 - o Contenedor semántico para el contenido principal de la página.
2. <section class="seccion-titulos">
 - o Sub-sección para títulos.
3. <div class="linea-roja"></div>
 - o Elemento decorativo barra roja), estilado en CSS.
4. <h3>Tienda</h3>
 - o Título de la sección. Está bien un h3 si h1 ya está usado.
5. <p class="subtitulo">...</p>
 - o Texto descriptivo pequeño bajo el título.

11. GRID DE CATEGORÍAS / TARJETAS

```
<section class="grid-categorias">
  <a href="suelos.html" class="enlace-tarjeta" data-cat="suelos">
    <article class="tarjeta" data-cat="suelos">
      
      <p class="titulo-tarjeta">Suelos</p>
    </article>
  </a>
  <!-- ... otros items similares ... -->
</section>
```

Las presentes líneas nos sirven de manera para:

1. <section class="grid-categorias">
 - Contenedor con grid CSS para tarjetas.
2.
 - Enlace que envuelve la tarjeta completa, así la tarjeta es clickeable.
 - data-cat="suelos" es un atributo data personalizado que puede usarse en JS para filtros o categorización.
3. <article class="tarjeta" data-cat="suelos">
 - artículo es semántico para un contenido independiente la tarjeta.
 - Clase tarjeta para estilos.
4.
 - Imagen de la tarjeta. alt describe la imagen.
5. <p class="titulo-tarjeta">Suelos</p>
 - Texto del título dentro de la tarjeta. Muchos diseños usan h4 o h3 para títulos — usar etiqueta de título semántica mejora SEO; sin embargo p está bien si quieras simple estilo.
6. </article>
 - Cierre de tarjetalink.

12. FOOTER

```
<footer class="pie-final">
  <div class="pie-barra-superior">
    <span>  </span>
    <span>Recibe ofertas y noticias</span>
  </div>

  <div class="pie-contenedor">
    <h2 class="pie-ayuda">¿Necesita ayuda? Llame al
    <strong>1.800.544.7220</strong> o use o use Ask <span class="curva-roja">Humboldt</span></h2>

    <div class="pie-grid">
      <div class="pie-col">
        <h4>Información</h4>
        <a href="registro.html">Contáctenos</a>
      </div>

      <div class="pie-col">
        <h4>Soporte</h4>
        <a href="registro.html">Crear Ticket</a>
      </div>

      <div class="pie-col">
        <h4>Cuenta</h4>
        <a href="registro.html">Iniciar Sesión</a>
        <a href="#">Blog</a>
      </div>

      <div class="pie-col">
        <h4>Síguenos</h4>
        <span>Facebook</span><br>
      </div>
    </div>
  </div>
</footer>
```

```

<span>TikTok</span>
</div>

</div>
</div>

<div class="pie-bajo">
    © 2025 Empresa SRL. Todos los derechos reservados.
</div>
</footer>
```

Las presentes líneas nos sirven de manera para:

1. <footer class="pie-final">
 - Pie de página semántico.
2. .pie-barra-superior
 - Barra negra con ícono y texto de newsletter.
3. .pie-contenedor y .pie-grid
 - Contenedores estructurados del footer en cuadrícula (grid) con columnas para enlaces.
4. <h2 class="pie-ayuda">¿Necesita ayuda? ... Humboldt</h2>
 - Mensaje destacado: span.curva-roja lo estilizas en rojo y itálica con CSS.
5. Las columnas .pie-col contienen enlaces.
 - Para mejorar: usa listas <a>... dentro del footer por semántica y accesibilidad.
 6. <div class="pie-bajo">
 - Última barra con copyright y fondo gris.

13. ENLACE AL SCRIPT Y CIERRE

```
<script src="script.js"></script>
```

```
</body>
</html>
```

Las presentes líneas nos sirven de manera para:

1. <script src="script.js"></script>
 - o Cargo el archivo JavaScript al final del body para que el DOM esté cargado antes de que JS intente acceder a elementos.
2. </body> y </html>
 - o Se cierran los contenedores.

ARCHIVO CSS

14. BODY.

```
body {
  font-family: Arial, Helvetica, sans-serif;
  background: #f5f7f8;
  color: #111;
  margin: 0;
  line-height: 1.3;
}
```

1. body {
 Inicio del selector global para toda la página.
 }
2. font-family: Arial, Helvetica, sans-serif;
 Define la familia tipográfica.
3. background: #f5f7f8;
 Color de fondo general (#f5f7f8 es gris muy claro).
4. color: #111;
 Color de texto por defecto (gris casi negro).
5. margin: 0;
 Elimina márgenes que los navegadores ponen por defecto.
6. line-height: 1.3;
 Altura de línea

15. ENCABEZADO GENERAL

```
.encabezado {
    background: #fff;
    border-bottom: 1px solid #ddd;
}
```

Las presentes líneas nos sirven de manera para:

1. .encabezado {
 Selecciona el contenedor <header>.
2. background: #fff;
 Fondo blanco.
3. border-bottom: 1px solid #ddd;
 Línea inferior gris clara.
 Estructura:
4. grosor | estilo | color, 1px | solid | #ddd

16. CONTENEDOR SUPERIOR DEL HEADER

```
.encabezado-superior {
    max-width: 1200px;
    margin: auto;
    display: flex;
    align-items: center;
    justify-content: space-between;
    padding: 12px 16px;
    gap: 16px;
}
```

Las presentes líneas nos sirven de manera para:

1. max-width: 1200px;
 El contenido no crece más de 1200px de ancho.
2. margin: auto;
 Centra horizontalmente.

3. display: flex;

Activa flexbox para organizar elementos en fila.

4. align-items: center;

Alinea verticalmente al centro.

5. justify-content: space-between;

Deja el primer elemento a la izquierda y el último a la derecha.

6. padding: 12px 16px;

Espaciado interno.

7. arriba/abajo | izquierda/derecha, 12px | 16px

8. gap: 16px;

Espacio entre elementos flex.

17. ÁREA DEL LOGO

.area-logo {

display: flex;

align-items: center;

gap: 12px;

}

1. display: flex;

Pone el logo y el texto en una fila.

2. align-items: center;

Centrado vertical.

3. gap: 12px;

Espacio entre imagen y texto.

18. LOGO

.logo {

width: 120px;

height: auto;

}

- **width: 120px;**

Ancho fijo del logo.

- **height: auto;**
Ajusta la altura proporcionalmente.

19. TEXTO DEL LOGO

```
.texto-logo {
    font-size: 20px;
    font-weight: bold;
    border-left: 3px solid #b80000;
    padding-left: 10px;
}

• font-size: 20px;  
Tamaño del texto.  

• font-weight: bold;  
Negrita.  

• border-left: 3px solid #b80000;  
Línea roja a la izquierda.  

• padding-left: 10px;  
Espacio entre línea y texto.
```

20. AREA DE BÚSQUEDA

```
.area-busqueda {
    flex: 1;
    display: flex;
    justify-content: center;
}

1. flex: 1;  
Ocupa el espacio sobrante disponible.  

2. display: flex;  
Para centrar contenido.  

3. justify-content: center;  
Centra horizontalmente el input + button.
```

21. INPUT DE BÚSQUEDA

```
.input-busqueda {
```

```

width: 75%;
max-width: 700px;
padding: 10px 14px;
border: 1px solid #bbb;
border-radius: 20px 0 0 20px;
font-size: 16px;
}

1. border-radius: 20px 0 0 20px;

Estructura completa:

2. superior-izq | sup-der | inf-der | inf-izq ; 20px   |  0   |  0   | 20px

```

1. Define forma semicircular izquierda.

21. BOTÓN BÚSQUEDA

```

.btn-buscar {

padding: 10px 14px;
background: #e6e6e6;
border: 1.8px solid #bbb;
color: white;
border-radius: 0 20px 20px 0;
cursor: pointer;
}

1. border-radius: 0 20px 20px 0;

Hace el lado derecho redondeado.

Estructura:

2. izq-sup | der-sup | der-inf | izq-inf
3. 0    20px    20px    0

```

22. ENLACE "REGISTRO" DEL USUARIO

```

.area-usuario a {

font-family: "Segoe UI", Arial, Helvetica, sans-serif;
font-size: 17px;
font-weight: 600;

```

```

color: #000;
text-decoration: none;
transition: 0.3s ease;
}

1. text-decoration: none; quita subrayado
2. color: #000; negro elegante
3. transition: 0.3s ease; suaviza hover

```

23. HOVER

```

.area-usuario a:hover {
color: #b80000;
transform: translateY(-1px);
}

1. transform: translateY(-1px);
Mueve 1px hacia arriba.

```

24. NAV PRINCIPAL

```

.nav-principal {
background: #000;
color: white;
display: flex;
justify-content: center;
gap: 26px;
padding: 10px;
}

```

Explicación de estructura:

1. gap: 26px; espaciado entre cada enlace
 2. padding: 10px;
- Equivale a:
3. 10px arriba
 4. 10px derecha
 5. 10px abajo
 6. 10px izquierda

25. HAMBURGUESA

```
.btn-burger {  
    display: none;  
    font-size: 28px;  
    background: none;  
    border: none;  
    cursor: pointer;  
}
```

1. Se oculta, solo para móvil

26. MEDIA QUERY MÓVIL

```
@media (max-width: 760px) {
```

2. Todo lo dentro se aplica solo en pantallas móviles.

```
@media (max-width: 760px) {
```

- o Es un comentario.
- o Sirve para identificar que lo siguiente corresponde al estilo móvil.

```
@media (max-width: 760px) {
```

- o Esta es una **media query**, una condición de CSS.
- o Significa:

26. Burguea bloque

```
.btn-burger {  
    display: block;  
}
```

1. .btn-burger, selecciona el ícono del menú hamburguesa.
2. display: block;
 - o En pantallas grandes el botón está oculto (display: none;).
 - o Aquí se muestra el menú hamburguesa solo debe existir en móvil.

27. NAV PRINCIPAL

```
.nav-principal {
    display: none;
    flex-direction: column;
    position: fixed;
    top: 70px;
    left: 0;
    right: 0;
    background: #000;
    padding: 20px;
    z-index: 999;
}
```

Explicación

display: none;

- Oculta el menú por defecto en móvil.
- Solo aparece cuando se activa .nav-open.

flex-direction: column;

- El menú se acomoda vertical uno debajo del otro.
- Antes, en PC, estaba horizontal.

position: fixed;

- Fija el menú en la pantalla.
- Aunque el usuario haga scroll, el menú se mantiene visible.

top: 70px;

- Mueve el menú 70px hacia abajo.
- Esto evita que el menú tape el logo o el encabezado.

left: 0;

- Alinea el menú al borde izquierdo de la pantalla.

right: 0;

- Alinea el menú al borde derecho.
- Combinando left 0 + right 0 = ocupa todo el ancho.

background: #000;

- Fondo negro del menú móvil.

padding: 20px;

- Esto es importante:

👉 padding tiene 1 valor → se aplica a *los 4 lados*.

- top: 20px
- right: 20px
- bottom: 20px
- left: 20px

z-index: 999;

- Asegura que el menú quede por encima de todo, incluyendo banner y tarjetas.

28. NAV OPEN

```
.nav-principal.nav-open {
    display: flex;
}
```

Explicación:

1. .nav-open es la clase que agrega JavaScript cuando haces click en la hamburguesa.
2. display: flex;
 - Hace visible el menú (antes estaba display: none).
 - Usa flexbox para organizar los enlaces verticalmente.

29. NAV PRINCIPAL

```
.nav-principal a {
    font-size: 20px;
    padding: 12px;
    text-align: center;
}
```

Explicación:

- .nav-principal a los enlaces dentro del menú.

font-size: 20px;

- Letras más grandes para que sean fáciles de tocar en móvil.

padding: 12px;

- Significa:

◦ top: 12px ,right: 12px ,bottom: 12px, left: 12px

- Hace cada enlace más grande.

text-align: center;

- Centra el texto dentro del enlace.

30. TITULOS

.seccion-titulos h3 {

font-size: 40px ;

font-weight: 800 ;

color: #111;

margin-bottom: 6px;

}

- Es solo una descripción.

- Indica que el siguiente bloque aplica al título principal dentro de .seccion-titulos.

.seccion-titulos h3 {

- .seccion-titulos es un contenedor.

- h3 significa: *todos los títulos <h3> que estén dentro de .seccion-titulos*.

font-size: 40px;

- Tamaño del texto.

- 40px significa que la letra medirá 40 píxeles de alto.

- Cuanto más grande el valor, más grande se muestra el texto.

- No acepta múltiples como 0 20 0 0 porque es una propiedad simple, no compuesta.

font-weight: 800;

- Grosor de la letra.
- Valores:
 - 100 = muy delgado
 - 400 = normal
 - 700 = negrita

color: #111;

- Color del texto.
- #111 es un color muy oscuro, casi negro.
- Mientras más repetido el número (111), más oscuro.
- #111 es equivalente a rgb(17,17,17).

margin-bottom: 6px;

- Agrega espacio hacia abajo del título.

31. TITULOS

```
.aviso {
  text-align: center;
  background: #fff;
  padding: 10px;
  border-bottom: 3px solid #eee;
}
```

- Es un comentario.
- Sirve solo para organización humana.
- Indica que este estilo corresponde al aviso que aparece debajo del menú.

```
.aviso {
```

- Abre el bloque de estilos para la clase .aviso.

- En HTML, sería el <div class="aviso">...</div> que muestra el texto "*¿No encuentra lo que busca?*".

text-align: center;

- Alinea horizontalmente el contenido del div.
- El texto se centra dentro del espacio disponible.

background: #fff;

- Asigna un fondo blanco puro.
- #fff

padding: 10px;

- Añade espacio interno entre el borde del contenedor y el contenido.
- Como solo se usa un valor, se interpreta como:

border-bottom: 3px solid #eee;

- Crea una línea inferior.
- Se interpreta como:
 - 3px → grosor de la línea.
 - solid → tipo de línea.
 - #eee → color gris muy claro.

31. BANNER

```
.banner {
  height: 550px;
  display: flex;
  justify-content: center;
  align-items: center;
  flex-direction: column;
  text-align: center;
  color: white;
  background: url("Imagenes/Medio.png") center;
}
.banner h1 {
  font-size: 40px;
```

```
text-shadow: 2px 2px 6px black;  
margin: 0;  
}
```

```
.banner h2 {  
font-size: 70px;  
font-weight: 800;  
text-shadow: 4px 4px 20px black;  
margin-top: 10px;  
}
```

Exlicacion:

```
.banner {
```

Selecciona el contenedor <section class="banner">.

```
height: 550px;
```

Define una altura fija de 550 píxeles para el banner.

Esto asegura que el banner sea grande, alto y visible.

```
display: flex;
```

Activa Flexbox, un sistema moderno de alineado.

Permite controlar alineaciones horizontales y verticales.

```
justify-content: center;
```

Alinea los elementos en el eje horizontal.

```
align-items: center;
```

Alinea los elementos en el eje vertical

Como resultado: el texto del banner queda perfectamente centrado.

flex-direction: column;

Indica que el contenido del banner (h1 y h2) se ordena en columna
Por defecto Flexbox ordena en fila esto lo invierte.

text-align: center;

Centra el texto internamente dentro del banner.

color: white;

Define el color de texto para todo el contenido dentro del banner blanco.

background: url("Imagenes/Medio.png") center;

Define la imagen de fondo del banner.

.banner h1 {

Aplica estilos exclusivamente al título <h1> dentro del banner.

font-size: 40px;

El tamaño del texto se vuelve bastante grande.

text-shadow: 2px 2px 6px black;

- Primer valor 2px: sombra hacia la derecha.
- Segundo 2px: sombra hacia abajo.
- Tercer 6px: difuminado.
- black: color de sombra.

margin: 0;

Elimina todos los márgenes del <h1>.

```
.banner h2 {
```

Estilos del segundo título grande en el banner.

```
font-size: 70px;
```

Mucho más grande que el h1.

```
font-weight: 800;
```

Texto muy grueso negrita fuerte.

```
text-shadow: 4px 4px 20px black;
```

Sombra más fuerte que en el h1:

- 4px derecha
- 4px abajo
- 20px difuminado
- black color

32. TARJETA

```
.titulo-tarjeta {
```

```
margin-top: 14px;
```

```
color: #000 ;
```

```
text-decoration: none ;
```

```
font-size: 20px;
```

```
font-weight: 700;
```

```
letter-spacing: 0.3px;
```

```
text-align: center;
```

```
font-family: 'Segoe UI', Arial, Helvetica, sans-serif;
```

```
}
```

```
.enlace-tarjeta,
```

```
.enlace-tarjeta p {
```

```
text-decoration: none ;  
color: inherit ;  
}
```

```
.titulo-tarjeta {
```

- .titulo-tarjeta, afecta a cualquier elemento HTML con esta clase, usualmente un <p> dentro de article.tarjeta.

```
margin-top: 14px;
```

- Separa el texto de la imagen.
- “Empuja” el texto hacia abajo.
- 14px es la distancia desde la parte superior del elemento.

```
color: #000 ;
```

- Color del texto.
- #000 = negro.

```
text-decoration: none ;
```

- Elimina subrayado.
- Útil cuando interactúa con <a>.

```
font-size: 20px;
```

- Tamaño de letra en píxeles.
- 20px es un tamaño visible y elegante.

```
font-weight: 700;
```

- Grosor de la letra.
- 700 = bold negrita fuerte.

letter-spacing: 0.3px;

- Espacio entre letras.
- 0.3px es un pequeño ajuste para hacer el texto más elegante y legible.

text-align: center;

- Centra el texto horizontalmente dentro de su contenedor.

font-family: 'Segoe UI', Arial, Helvetica, sans-serif;

- Lista de fuentes:
 1. Segoe UI → si existe en el sistema, se usa.
 2. Arial → si no está Segoe.
 3. Helvetica → si no está Arial.
 4. sans-serif → fuente genérica en última opción.

.enlace-tarjeta,

.enlace-tarjeta p {

- a cualquier elemento con la clase .enlace-tarjeta
- y también a cualquier <p> dentro de .enlace-tarjeta

text-decoration: none ;

- Elimina el subrayado en:
 - el enlace principal
 - el texto interno

color: inherit ;

- inherit significa "hereda el color de su contenedor".
- el texto cambie de color al pasar el mouse.
- Mantiene el diseño uniforme.

33. BLOQUE: DEBOUNCE

```
function debounce(fn, wait = 200) {
  let t;
  return (...args) => {
    clearTimeout(t);
    t = setTimeout(() => fn.apply(this, args), wait);
  };
}
```

Explicación:

1. function debounce(fn, wait = 200) {
 - o Declara la función debounce que recibe dos parámetros:
 - fn: la función que queremos "debouncear" evitar ejecuciones muy frecuentes.
 - wait = 200: tiempo en milisegundos
2. let t;
 - o Declara la variable t que almacenará el identificador del setTimeout. Se usa para poder cancelar el timeout anterior con clearTimeout.
3. return (...args) => {
 - o Devuelve una función que acepta cualquier número de argumentos (...args) y que será la versión "debounceada" de fn.
4. clearTimeout(t);
 - o Cancela cualquier timeout pendiente evita que la invocación previa de fn se ejecute.
5. t = setTimeout(() => fn.apply(this, args), wait);
 - o Programa una nueva ejecución de fn después de wait ms.
 - o Usa fn.apply(this, args) para llamar fn con los mismos argumentos recibidos.

34. MENÚ MÓVIL

```
document.addEventListener("DOMContentLoaded", () => {
  const nav = document.querySelector(".nav-principal");
  const burger = document.querySelector(".btn-burger");
```

```

if (!nav || !burger) return;

burger.addEventListener("click", () => {
  const isOpen = nav.classList.toggle("nav-open");

  if (isOpen) {
    document.body.style.overflow = "hidden";
    nav.style.maxHeight = "100vh";
    nav.style.overflowY = "auto";
  } else {
    document.body.style.overflow = "";
    nav.style.overflowY = "";
    nav.style.maxHeight = "";
  }
});

});

```

Explicación:

1. document.addEventListener("DOMContentLoaded", () => {
 - o Espera a que el DOM esté cargado antes de ejecutar el bloque. Así evitamos null si los elementos aún no existen.
}
2. const nav = document.querySelector(".nav-principal");
 - o Selecciona el elemento del menú principal (el <nav>). nav será null si no existe.
3. const burger = document.querySelector(".btn-burger");
 - o Selecciona el botón hamburguesa. burger será null si no existe.
4. if (!nav || !burger) return;
 - o Guard clause: si alguno de los elementos no existe, termina la función (evita errores).
5. burger.addEventListener("click", () => {
 - o Añade un listener para el clic en el botón hamburguesa.
}
6. const isOpen = nav.classList.toggle("nav-open");

- Alterna la clase nav-open en el nav. toggle devuelve true si la clase quedó añadida (menú abierto), false si se removió (cerrado).
- Guardamos ese boolean en isOpen.

7. if (isOpen) {
 - Si el menú quedó abierto...
8. document.body.style.overflow = "hidden";
 - Bloquea el scroll del body (fondo) estableciendo overflow: hidden.

Esto evita que la página detrás del menú se mueva mientras el menú esté abierto.
9. nav.style.maxHeight = "100vh";
 - Establece una altura máxima al nav igual a la altura de la ventana (100vh). Eso evita que el menú intente crecer más que la pantalla.

Es útil para limitar su tamaño cuando el contenido es largo.
10. nav.style.overflowY = "auto";
 - Permite que el propio nav tenga scroll vertical (overflow-y: auto) si su contenido supera maxHeight. Esto es clave para que puedas desplazar el contenido dentro del menú sin mover el fondo.
11. document.body.style.overflow = "";
 - Restaura el comportamiento de scroll del body (por defecto). Dejar "" quita el estilo en línea.
12. nav.style.overflowY = "";
 - Quita el overflow-y del nav vuelve a lo declarado en CSS
13. nav.style.maxHeight = "";
 - Quita maxHeight del nav, recuperando su estilo original de CSS.

35. BUSCADOR DE PRODUCTOS

```
document.addEventListener("DOMContentLoaded", () => {
  const input = document.querySelector(".input-busqueda");
  const grid = document.querySelector(".grid-categorias, .grid-suelos");

  if (!input || !grid) return;
```

```

const tarjetas = [...grid.querySelectorAll(".tarjeta, .item-suelo")];

const items = tarjetas.map(el => {
  const tituloEl = el.querySelector(".titulo-tarjeta") || el.querySelector("h3");
  const titulo = tituloEl ? tituloEl.textContent.toLowerCase() : "";
  return { el, titulo };
});

input.addEventListener("input", debounce(() => {
  const texto = input.value.toLowerCase();
  items.forEach(it => {
    it.el.style.display = it.titulo.includes(texto) ? "" : "none";
  });
}, 150));
});

```

Explicación>

1. document.addEventListener("DOMContentLoaded", () => {
 - Bloque dentro de DOMContentLoaded asegura que los elementos existen.
}
2. const input = document.querySelector(".input-busqueda");
 - Selecciona el campo de búsqueda.
3. const grid = document.querySelector(".grid-categorias, .grid-suelos");
 - Selecciona el contenedor de tarjetas.
 - document.querySelector con la coma devuelve el primer selector que coincide. Si tienes ambos elementos en distintas páginas, este patrón busca el primero que exista.
4. if (!input || !grid) return;
 - Si no existen, no sigue.
5. const tarjetas = [...grid.querySelectorAll(".tarjeta, .item-suelo")];
 - Crea un array con todas las tarjetas dentro del grid soporta dos clases diferentes .tarjeta o .item-suelo.

6. const items = tarjetas.map(el => {
 - Mapea cada tarjeta a un objeto con referencias útiles.
}
7. const tituloEl = el.querySelector(".titulo-tarjeta") || el.querySelector("h3");
 - Intenta localizar el elemento que contiene el título (primero .titulo-tarjeta, si no un h3).
}
8. const titulo = tituloEl ? tituloEl.textContent.toLowerCase() : "";
 - Lee el texto del título y lo normaliza a minúsculas para búsquedas insensibles a mayúsculas.
}
9. return { el, titulo };
 - Devuelve un objeto con la tarjeta (el) y su texto (titulo).
}
- 10.});
 - Cierre del map.
}
11. input.addEventListener("input", debounce(() => {
 - Escucha el evento input en el campo y llama la función debounceada (evita búsquedas en cada pulsación).
})
12. const texto = input.value.toLowerCase();
 - Toma el texto actual del input y lo pasa a minúsculas.
}
13. items.forEach(it => {
 - Itera todas las tarjetas preprocesadas.
})
14. it.el.style.display = it.titulo.includes(texto) ? "" : "none";
 - Si el título contiene la cadena buscada, muestra la tarjeta (""
restablece estilo); si no, la oculta (display: none).
})

36. BLOQUE: SCROLL REVEAL

```
document.addEventListener("DOMContentLoaded", () => {
  const elementos = document.querySelectorAll(".tarjeta, .item-suelo");
  if (!elementos.length) return;

  const obs = new IntersectionObserver(entries => {
    entries.forEach(e => {
      if (e.isIntersecting) e.target.classList.add("visible");
    });
  });
})
```

```

}, { threshold: 0.15 });

elementos.forEach(el => obs.observe(el));
});

```

Explicación:

1. document.addEventListener("DOMContentLoaded", () => {
 - o Ejecuta cuando DOM listo.
}
2. const elementos = document.querySelectorAll(".tarjeta, .item-suelo");
 - o Selecciona todos los elementos que queremos animar al aparecer.
}
3. if (!elementos.length) return;
 - o Si no hay elementos, sale.
}
4. const obs = new IntersectionObserver(entries => {
 - o Crea un IntersectionObserver que vigila la visibilidad de los elementos en la viewport.
}
5. entries.forEach(e => {
 - o Itera las entradas (cada elemento observado).
}
6. if (e.isIntersecting) e.target.classList.add("visible");
 - o Si el elemento está entrando en el viewport (isIntersecting), añade la clase .visible (la CSS maneja la animación/transición).
}
7. });
 - o Cierre forEach.
8. }, { threshold: 0.15 });
 - o Configuración del observer:
 - threshold: 0.15 → se considera visible cuando el 15% del elemento entra en vista. Puedes ajustar entre 0–1.
9. elementos.forEach(el => obs.observe(el));
 - o Comienza a observar cada elemento.

37. BOTÓN VOLVER

```

document.addEventListener("DOMContentLoaded", () => {
  const btn = document.createElement("button");
  btn.className = "btn-arriba";

```

```
btn.innerHTML = "↑";
document.body.appendChild(btn);

window.addEventListener("scroll", () => {
  btn.style.display = window.scrollY > 500 ? "block" : "none";
});

btn.addEventListener("click", () => {
  window.scrollTo({ top: 0, behavior: "smooth" });
});

1. document.addEventListener("DOMContentLoaded", () => {
  ○ Espera DOM.
2. const btn = document.createElement("button");
  ○ Crea dinámicamente un botón.
3. btn.className = "btn-arriba";
  ○ Le asigna la clase para aplicar estilos desde CSS.
4. btn.innerHTML = "↑";
  ○ Contenido (flecha). Podrías usar un ícono SVG si lo prefieres.
5. document.body.appendChild(btn);
  ○ Inserta el botón al final del body.
6. window.addEventListener("scroll", () => {
  ○ Escucha el evento scroll.
7. btn.style.display = window.scrollY > 500 ? "block" : "none";
  ○ Muestra el botón si se scrolllearon más de 500px (ajustable).
8. });
  ○ Cierre del listener.
9. btn.addEventListener("click", () => {
  ○ Cuando el usuario hace click en el botón...
10. window.scrollTo({ top: 0, behavior: "smooth" }));
```

- Hace scroll suave hacia arriba (0, top) usando la API scrollTo con behavior: "smooth".