

GRADO FORMATIVO	DAW – DESARROLLO DE APLICACIONES WEB
CURSO (1º o 2º)	2º
ASIGNATURA	Desarrollo interfaces web
NOMBRE Y APELLIDOS DEL ALUMNO	CARLOS PARRO PÉREZ
DNI	48159548D
FECHA	12/11/2021

HITO INDIVIDUAL 1 - PRIMER TRIMESTRE 2021

Objetivos:

- Asimilar los contenidos de la asignatura Desarrollo interfaces web desarrollados en el primer trimestre 2021
- Aprender los fundamentos más relevantes de Planificación de interfaces gráficas y uso de estilos CSS
- Investigación de los recursos web en relación con la analítica y accesibilidad de sitios web

Materiales:

- Editores / IDE para editar código en Javascript / Typescript. Por ejemplo **Visual Studio Code**, Atom, bracktes, Sublime...
- Git / Github

Enunciado: LEE BIEN EL ENUNCIADO ANTES DE PASAR AL DESARROLLO

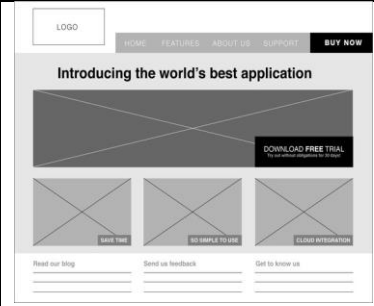

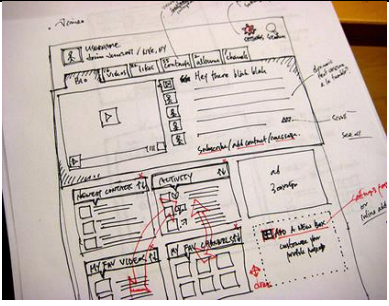
El hito en realizar tres actividades sobre interfaces web en la aplicación de desarrollo web.

Desarrollo: LEE BIEN LAS CUESTIONES PLANTEADAS ANTES DE RESOLVERLAS. LEE BIEN LO QUE SE PIDE COMO RESPUESTA

CUESTIÓN 1. *Planificación de una interfaz gráfica*

- *Explica con un ejemplo sencillo la diferencia entre prototipo, wireframe y mockup*

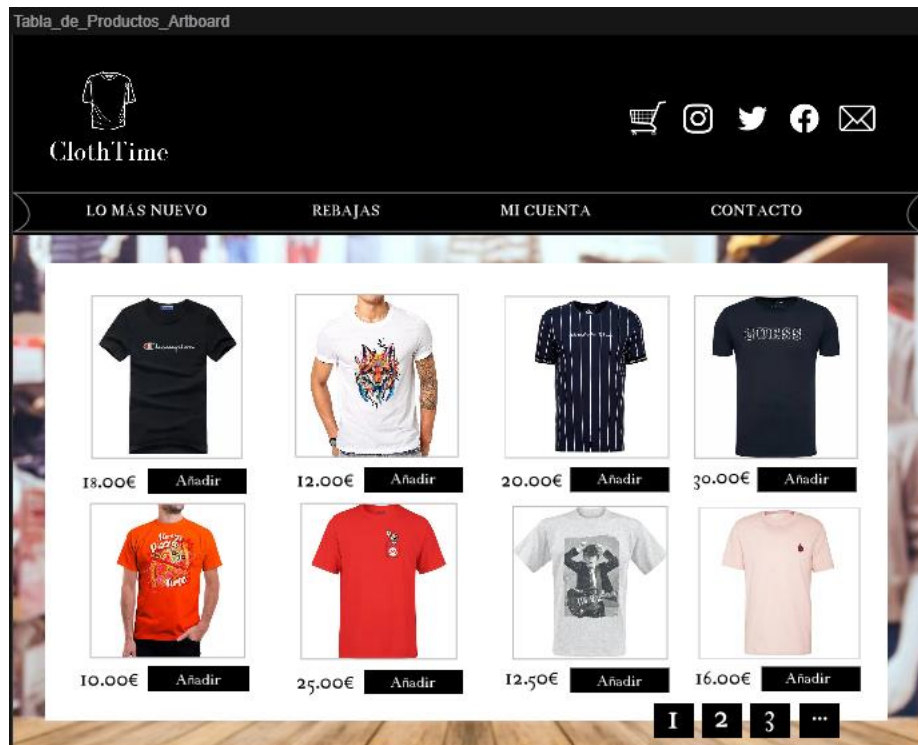
Al hablar de prototipado podemos entablar tres elementos y sus diferencias, por ejemplo, el **Wireframe** el cual utiliza un diseño gráfico basado en conceptos, componentes y estructura básicos de la web sin entrar en detalles utilizando una escala de grises, objeto de ideas y propuestas dentro de un equipo de diseño; el **Mockup** define la apariencia de la web en cuestión entrando más en detalles visuales tales como colores, tipografías sombras etc para evaluar el aspecto visual y la comunicación con el usuario; y, por último, los **Prototipos** que definen la interacción con la interfaz y la funcionalidad para tener conocimiento del comportamiento de la web.

Wireframe	Mockup	Prototipo
		

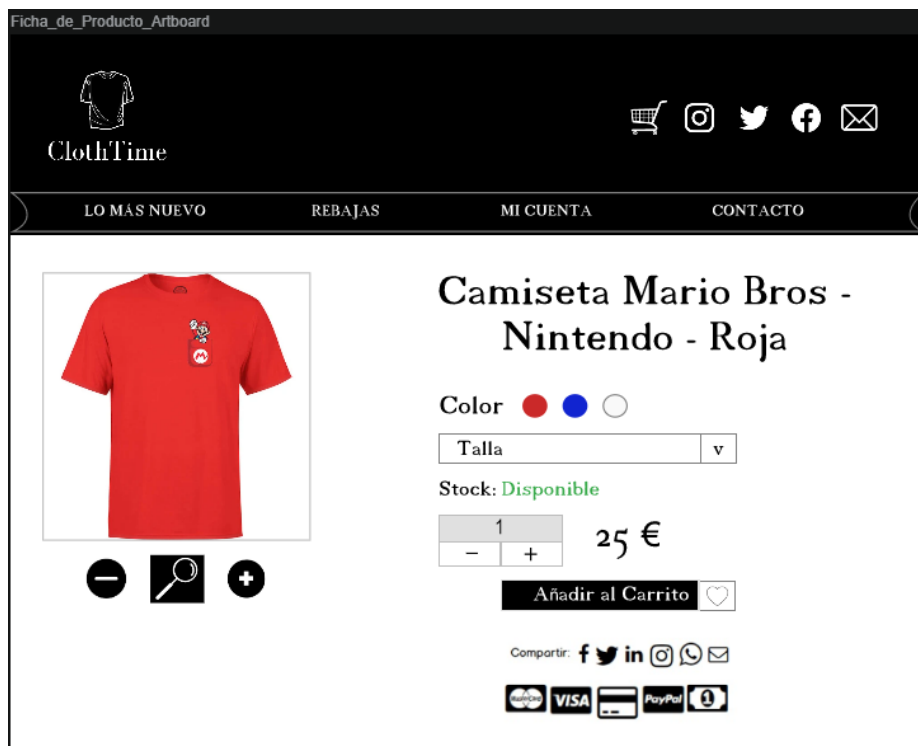
- Diseña alguno de los elementos anteriores para un sitio web basado en E-Commerce

MOCKUP

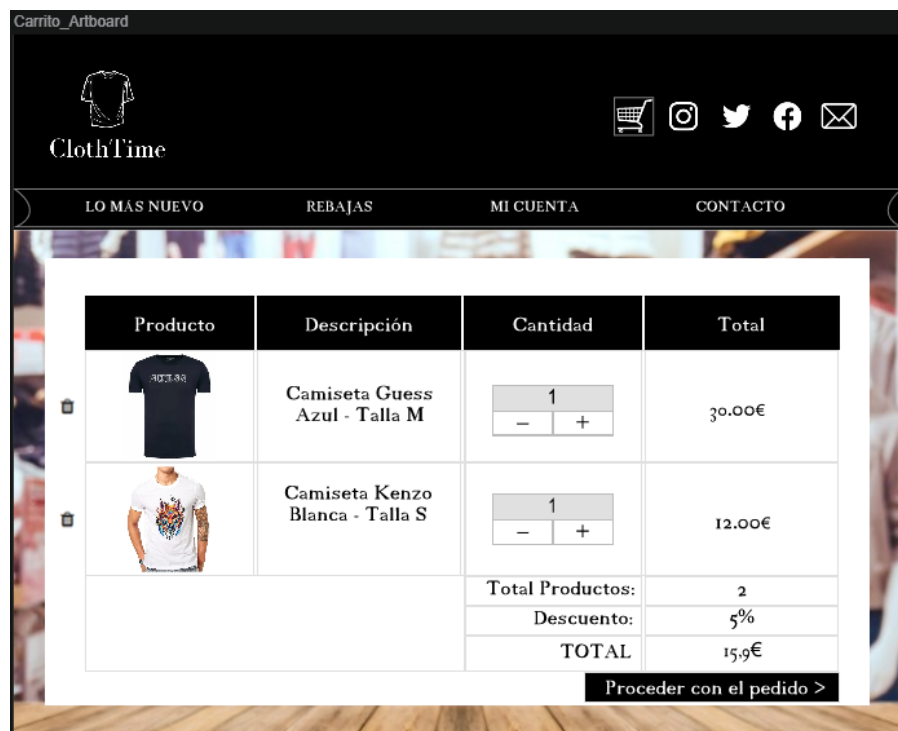
1.1..1. Tabla de productos



1.1..2. Ficha de producto



1.1.3. Carrito



1.1.4. Finalizar pago



RESPUESTA CUESTION 1: Documento con explicación y enlace con el “diseño”. Por ejemplo, si lo haces en InVision, el enlace de invision en donde veamos las pantallas y su interacción

<https://chaaarly9942488.invisionapp.com/prototype/ckvy8fpte002ipa01t5loeryt/play>

CUESTIÓN 2. *Estilos CSS*

- Diseña una web en donde realizas un ejemplo aplicando los diferentes selectores de estilos

*2.1..1. Etiqueta – id – class***Etiqueta**

Accedemos a los elementos p (texto) mediante el nombrado de la etiqueta.

```
p{  
  font-weight:bold;  
  text-align: left;  
}
```

Id

Accedemos al elemento cuya Id sea #fs1.

```
#fs1{  
  background-color: ■ #f5f5dc;  
}
```

Class

Accedemos al elemento cuya Id sea #fs1.

```
.responsive {  
  padding: 8px 8px 8px 8px; /*Izquierda, derecha,  
  arriba, abajo*/  
  float: left;  
  width: 24.99999%;  
}
```

2.1..2. Pseudoselectores para enlaces de navegación

Los pseudoselectores se usan para definir un estado especial en un elemento y se pueden utilizar para aplicar estilos cuando el usuario pasa el ratón por un elemento, para aplicar estilo cuando el usuario ha visitado o no visitado un enlace o aplicar estilo al enfocar un elemento.

div.galeria: hover -> Establece un borde negro sólido de un píxel de ancho al elemento div cuyo atributo class es “galería” cuando el usuario posiciona el mouse en el elemento.

div.galeria:visited -> Establece un borde azulado sólido de un píxel de ancho al elemento div cuyo atributo class es “galería” cuando el usuario ha visitado el link mediante el click del elemento.

```
div.galeria: hover {
  border: 1px solid black;
}
div.galeria:visited {
  border: 1px solid lightseagreen;
}
```

2.1..3. Selectores secundarios (*div a*, *div >a*, *div :first-child*)

Accedemos a todos los elementos hijo del elemento div cuyo atributo class sea “galería”.

```
div.galeria img {
  width: 100%;
  height: 100px;
}
```

Accedemos a todos los elementos legend que estén dentro de un elemento Fieldset

```
fieldset > legend{
  color: black;
  text-align: center;
  font-weight: bold;
  background-color: #f5f5dc;
  border: 1px solid black;
}
```

Accedemos al primer elemento Fieldset de la web

```
fieldset: first-child{
  border: 1px solid black;
}
```

- *Explica con un ejemplo la precedencia de estilos*

Al hablar de precedencia de estilos nos referimos a cómo evalúa el navegador la preferencia de un selector sobre otro a la hora de aplicar un estilo a un elemento en concreto. Está basada en las reglas de coincidencia entre los diferentes tipos de selectores de css y cada tipo contiene un peso (importancia o valor) correspondiente. Cuando aparecen varias declaraciones con el mismo peso o importancia, se aplicará la última declaración especificada en el css.

Podemos distinguir los siguientes tipos de selectores los cuales coinciden en orden con el valor o importancia dentro de la precedencia de estilos que son los **Selectores de Tipo** como `div` y **pseudoelementos** (permiten añadir estilos a una parte concreta del elemento, por ejemplo `::first-line`, que apunta a la primera línea de elemento texto, o `::backdrop` que aplica un estilo al fondo de un elemento), los **Selectores de Clase** como `.clase`, Selectores de atributos como `[type:"text"]` y **pseudoclases** (que especifican un estado especial de un elemento, por ejemplo `:read-only` que representa un elemento no editable por el usuario, o `:required` que representa un elemento que es obligatorio de rellenar con datos válidos para poder enviar a través de un formulario), y, por último, los **Selectores de ID** por ejemplo `#id`.

El empleo de **!important** al declarar un estilo a un elemento css es una mala práctica que no debe usarse ya que podría hacer más difícil la posible depuración del código lo que provocaría un conflicto de declaraciones. Esta declaración sobrescribe a cualquier otra, por lo tanto, hay que realizar un mejor uso de las propiedades en cascada.

Hay que tener en cuenta que al utilizar la pseudoclase de negación `:not` como tal no influye en la precedencia de estilos pero sí lo harán los selectores dentro de ella, por ejemplo:

```
div.peliculas p{
    background-color: blue;
}
div:not(.peliculas) p{
    background-color: red;
}
```

Con el html:

```
<div class="peliculas">
    <p>Este es un texto.</p>
    <div class="container">
        <p>Este es otro texto.</p>
    </div>
</div>
```

El color de la fuente del elemento `<p>` del div cuyo atributo class es "películas" será azul.
`Este es un texto.`

El color de la fuente del elemento `<p>` del div cuyo atributo class es "container" será rojo.
`Este es otro texto.`

Cuando hablamos de la precedencia de estilos basados en la forma hablamos de la especificación de un atributo para determinar a qué elemento queremos apuntar, por ejemplo, quiero apuntar a los elementos cuyo id sea campusfp:

```
#campusfp {  
    text-align:center;  
}  
  
[id="campusfp"] {  
    text-align:left;  
}
```

Con el html:

```
<p id="campusfp">Este es un texto.</p>
```

El elemento p cuyo atributo id es "campusfp" tendrá una alineación al centro debido a que al ser el mismo elemento el selector ID tiene un mayor nivel dentro de la precedencia de estilos.

Por último, cabe destacar la diferencia en cuanto a precedencia de estilos de la declaración directa y los estilos heredados. Los estilos de declaración directa siempre tendrán preferencia ante los estilos heredados.

Por ejemplo:

```
#padre {  
    border: 1px solid purple;  
}  
  
div {  
    border: 1px solid red;  
}
```

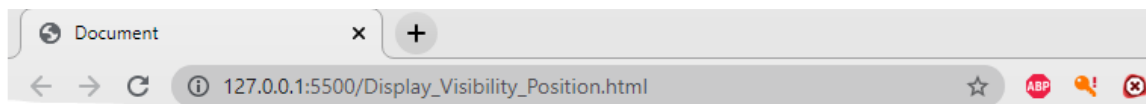
Con el html:

```
<html>  
<body id="padre">  
    <div>  
        <p>Hola.</p>  
    </div>  
</body>  
</html>
```

El div tendrá un borde rojo ya que en el css se selecciona específicamente el elemento a estilizar, el borde de color morado simplemente es heredado del padre <body> pero no se aplica.

- Muestra un ejemplo con la diferencia entre *display*, *position* y *visibility*

En cuanto a las propiedades css **Display** y **Visibility** estamos hablando de establecer el comportamiento en cuanto a una caja (box), en el cual la diferencia está en que **Display:none** no reserva el espacio del elemento a eliminar y **Visibility:hidden** si lo reserva, los cuales son propiedades similares pero no iguales en comportamiento. La propiedad **Position** especifica el tipo de posicionamiento para un elemento dentro de una web. Position, a diferencia de *Display* y *Visibility*, no tiene un atributo none o hidden que especifique que un elemento no se visualice en la web.



Diferencias entre Display, Visibility y Position

Display: none:

Display none no mantiene la posición del elemento en la web al desaparecer. El elemento ha desaparecido y ha perdi

Visibility: hidden:

Visibility hidden mantiene la posición del elemento en la web al desaparecer.

El elemento ha desaparecido pero su posición la sigue manteniendo.

Position:

Position establece el posicionamiento de un elemento en la web. Pudiendo tomar valores como **STATIC**(que se posiciona de acuerdo con el flujo de la página), **RELATIVE**(que puede tomar valores en torno a las propiedades left, right, up y down para ajustarse con respecto a su posición normal), **FIXED**(que se posiciona en relación a la ventana gráfica, por lo que siempre se queda en una posición fija aunque el usuario desplace la página), **ABSOLUTE**(que se posiciona en relación con su elemento padre, si no tiene, se posiciona con respecto al body), y **STICKY**(que se posiciona según el desplazamiento del usuario alternando entre relative y fixed).

RESPUESTA CUESTION 2.1.: Archivos css y html si así lo consideras publicados en GitHub.

CUESTIÓN 3. *Preprocesadores de estilos CSS*

- *Explica qué es un preprocesador de CSS y su utilidad*

Programas que te permiten la generación de código css a partir de la sintaxis del preprocesador en concreto. Se utilizan para hacer la estructura y el código de estilo más fácil de mantener y entender. Para utilizar un preprocesador necesitas instalar el compilador del preprocesador a utilizar para poder ser procesado por el navegador.

- *Realiza una web con texto, formulario, tabla e imagen y aplica estilo con uno de tus preprocesadores favorito*

Ver en la web.

- *Navega la web y comprueba si se aplica CSS o el estilo con el preprocesador*

RESPUESTA CUESTION 3.1.: *Archivos css,less,sass,scss y html si así lo consideras publicados en GitHub.*

Firma del alumno:

A handwritten signature in black ink, consisting of several loops and a long horizontal stroke.

Rúbrica:**PUNTUACIÓN MÍNIMA PARA CONSIDERAR LA PRUEBA SUPERADA: 5 PUNTOS**

CUESTION 1 Prototipado	La explicación no es correcta y el prototipo no funciona	La explicación es correcta y el prototipo funciona con elementos básicos	El prototipo muestra una interacción y diseños atractivos cuidando UX/UI
	0 puntos	2 puntos	3 puntos
CUESTION 2 CSS	Los estilos NO se aplican correctamente	Los estilos se aplican correctamente	Utiliza selectores avanzados y propiedades menos habituales. Por ejemplo, no todo es Font-size, color, background-color... Utilizas last-child, >,
	0 puntos	2 puntos	3 puntos
CUESTION 3 Preprocesadores CSS	Los procesadores no funcionan	Los procesadores funcionan correctamente	Has utilizado scss o sass y aplicas un diseño de display, grid, layout... más avanzado
	0 puntos	2 puntos	3 puntos
DOCUMENTACION APORTADA	Documentación correctamente maquetada y publicada en GitHub ordenado		
	1 punto		