	ESCUELA SUPERIOR DE ECONOMÍA Y NEGOCIOS **INGENIERÍA DE SOFTWARE Y NEGOCIOS DIGITALES**
CICLO 03-2025	**GUIA DE LABORATORIO Nº 4** **Nombre de la práctica:** Introducción al Responsive Design **Tiempo estimado:** 2 horas **Materia:** Desarrollo Web I

I. OBJETIVOS

Que al finalizar la practica el estudiante aprenda a:

- Crear y estilizar formularios utilizando la diversidad de controles de HTML5.
- Utilizar correctamente la meta-etiqueta viewport.
- Defina correctamente las media queries para aplicar reglas de estilo en función del ancho del viewport del navegador.
- Utilice los conocimientos adquiridos durante el transcurso de la asignatura para la construcción de páginas web responsivas.

II. INTRODUCCIÓN TEORICA

En la actualidad es más frecuente que el acceder a Internet se realice con diferentes dispositivos, que a su vez poseen diferentes tamaños o resoluciones de pantallas, inclusive podemos notar que en el caso de manipular una Tablet o Smartphone, tenemos la posibilidad de utilizar la función de rotación de pantalla, esto quiere decir que los actuales sitios web deben tener la capacidad de adaptarse a este tipo de comportamiento.

El diseño web adaptivo (Responsive Web Design o RWD) es una técnica de diseño y desarrollo web que permite generar sitios web que se adapten a cualquier tipo de resolución de pantalla, esto es gracias al uso de estructuras fluidas y media queries basadas en HTML5 y CSS3.

Hay que tener en cuenta que el concepto de diseño responsivo y diseño adaptivo difieran, ya que por una parte un diseño responsivo responde en todo momento a las dimensiones del dispositivo, mientras que un diseño adaptable es aquel que se adapta, pero no necesariamente responde en todo momento.

Por otro lado, para trabajar correctamente en diseños responsive hay que tener en cuenta que debemos trabajar con unidades relativas e intentar evitar las unidades fijas o estáticas, las cuales no responden a la adaptación de nuestros diseños flexibles:

Unidades relativas

Unidades estáticas

La utilización de las propiedades como utilizar propiedades como **min-width** o **max-width**, donde definimos tamaños mínimos o máximos, para que los elementos de nuestra página puedan ampliar o reducirse según sea necesario dependiendo de la pantalla del dispositivo utilizado.

Con estas propiedades podemos crear diseños que aprovechen al máximo toda la pantalla de dispositivos pequeños (como móviles o tablets), mientras que establecemos unos máximos en pantallas de dispositivos grandes, para crear unos espacios visuales que hacen que el diseño sea más agradable.

Otra característica de los diseños responsivos es la de mantener el flujo de los elementos cuando cambia el tamaño y evitar que estos se traslapen unos con otros. En este escenario aparecen los puntos de control o también conocidos como breakpoints. Por ejemplo: si una resolución de una computadora de escritorio queremos mostrar la información dentro de una cuadricula (Grid) de 4, 6 o 8 celdas de ancho, mientras que en una versión de Tablet serían 3 celdas de ancho y en el resto de dispositivos móviles se mostraría una celda de ancho que estaría acompañada de la función scroll. Con este ejemplo mostramos el funcionamiento de los puntos de control y el flujo de los elementos.

Estrategias de diseño

Generalmente en el diseño responsivo existen dos principales estrategias de diseño las cuales son:

- Mobile first: primero nos enfocamos en dispositivos móviles y luego diseñamos para pantallas de escritorio.
- Destktop first: primero nos enfocamos en dispositivos de escritorio y luego diseñamos para dispositivos móviles.

Elementos del diseño web responsive

Los principales elementos que se deben considerar en la construcción de sitios web responsivos son las siguientes:

1. Meta tag viewport
2. Estructura con diseño fluido
3. Utilización de media queries

1. El meta tag viewport

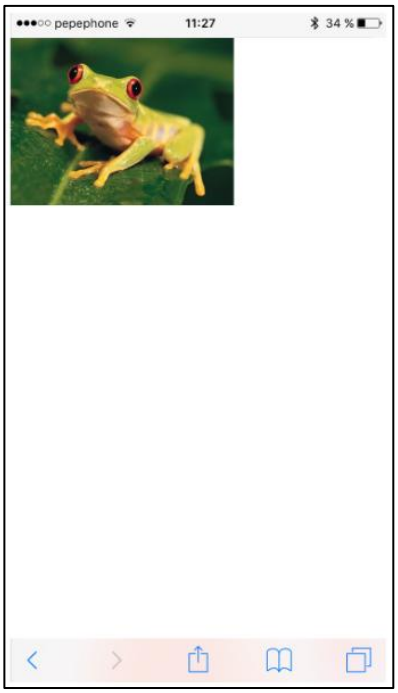
El viewport en un navegador de una computadora es igual al área disponible para renderizar el documento web (o sea, le restamos toda la interfaz del navegador, como botones, barra de direcciones, barra de menús, barras de desplazamiento, etc.) Dicho de otro modo, es el área útil donde se mostrará la página web.

Para el caso de los dispositivos móviles el viewport no corresponde al tamaño real de la pantalla en píxeles, sino al espacio que la pantalla está emulando que tiene. Por ejemplo, en un iPhone, aunque la pantalla en vertical tiene unas dimensiones de 320 píxeles, en realidad el dispositivo está emulando tener 980 píxeles. Esto hace que ciertas páginas web (optimizadas para navegadores de escritorio) quepan en una pantalla de 320 píxeles, porque en realidad el Safari para iOS está emulando tener un espacio de 980 píxeles.

Los desarrolladores somos capaces de alterar el viewport que viene configurado en el navegador, algo que resulta totalmente necesario si queremos que nuestra página se vea correctamente en dispositivos móviles.

Sin manipulación del zoom

Manipulación del zoom de un dispositivo



Si queremos editar ciertos comportamientos del viewport del navegador, podemos editar el documento HTML para especificar el siguiente campo meta, antes de la parte del </head>:

```
<meta name="viewport" content="initial-scale=1, width=device-width">
```

Con esta etiqueta <meta>, estamos estableciendo unos parámetros de comportamiento para el viewport del navegador. En la siguiente tabla se muestra los atributos y valores que podemos modificar.

Propiedades	Valor	Significado
width	device-width	Indica un ancho para el viewport.
height	device-height	Indica un alto para el viewport.
initial-scale	1	Escala inicial con la que se visualiza la página web.
minimum-scale	0.1	Escala mínima a la que se puede reducir al hacer zoom.
maximum-scale	10	Escala máxima a la que se puede aumentar al hacer zoom.
user-scalable	no/fixed | yes/zoom	Posibilidad de hacer zoom en la página web.

Las propiedades initial-scale, minimum-scale y maximum-scale permiten valores desde el 0.1 al 10, aunque ciertos valores se traducen automáticamente a ciertos números determinados:

- yes = 1
- no = 0.1
- device-width = 10
- device-height = 10

Por otra parte, user-scalable permite definir si es posible que el usuario pueda «pellizcar» la pantalla para ampliar o reducir el zoom.

2. Estructura con diseño fluido

El diseño web responsivo tiene que contar con elementos fluidos que se adapten al contexto de los dispositivos, respecto a ello se debe considerar las siguientes estrategias:

2.1. ***Tamaño de los elementos***, aquí debe considerar las siguientes estrategias:

a. Utilizar porcentajes en vez de pixeles. Por ejemplo: width: 60%

b. Limitar el ancho o alto máximo de los elementos usando pixels con el atributo max-width o max-height, para evitar que quede una página kilométrica en caso de monitores muy grandes.

c. No usar posiciones absolutas ni fijas, es preferible flotar los elementos con float:left | right; o utilizar la propiedad display, ya sea con inline | block | inline-block, o bien con los valores correspondientes a tablas, como table | table-row | table-column | table-cell.

2.2. ***Tamaño de fuentes***, antes de la llegada de CSS3, los tamaños de fuentes más utilizados para redimensionar las fuentes eran los px y em, pero ambos tienen desventajas:

a. **px:** los tamaños de letra especificados en px son tamaños absolutos, por lo que no generan el efecto cascada, pero no escalan.

b. **em:** los tamaños de letra em son relativos al elemento padre, por lo que son escalables pero generan el efecto cascada, de tal manera que en caso de elementos anidados el tamaño de letra se va haciendo progresivamente mayor o menor. Por ejemplo:

Utilizando px y em

```
body {
    font-size: 20px;
}

div {
    font-size: 0.5em;
}
```

Resultado de px sobre em

```
<body> =20px
    <div> =10px
        <div> =5px
            <div> =2.5px
                <div> =1.25px
```

CSS3 introduce la unidad rem, cuyo nombre deriva de "root em", y significa que la unidad rem es relativa al elemento raíz, por lo que es escalable y además evita el efecto cascada. De esta manera, una vez definido el tamaño de fuente en el elemento <html>, podemos establecer todos los tamaños rem para que sean porcentuales respecto al elemento raíz. Por ejemplo:

```
html {
    /* =10px, asumiendo que el tamaño de
    fuente predeterminado del navegador es 16px */
    font-size: 62.5%;
}

body {
    font-size: 1.6rem; /* =16px */
}

h1 {
    font-size: 2.4rem; /* =24px */
}
```

2.3. ***Tamaño de imágenes***, para que las imágenes se comporten de manera elástica y ocupen como máximo el ancho del elemento contenedor.

```
img {
    max-width: 100%;
    height: auto;
}
```

2.4. ***Tamaño de videos,*** para que los videos HTML5 (etiqueta <video>) se comporten de manera elástica y ocupen como máximo el ancho del elemento contenedor. Para videos embebidos, que utilizan etiquetas <iframe>, <object> o <embed> en vez de <video>, el video embebido se inserta en un , en este caso deberá aplicar un posicionamiento absolute para la etiquetas <iframe>, <object> o <embed> y para el div que contiene dichas etiquetas deberá de aplicar un posicionamiento relative y ocultar con la propiedad css overflow.

```
video {
    max-width: 100%;
    height: auto;
}
```

2.5. ***Palabras muy largas,*** Para evitar que las palabras o URLs muy largas se salgan de las cajas, usaremos la propiedad word-wrap.

```
.site-content {
    word-wrap: break-word;
}
```

2.6. ***Ajuste del tamaño del texto,*** Para controlar cómo ajustan el tamaño del texto los navegadores basados en Webkit (Chrome, Safari, iPhone) e Internet Explorer y evitar inconsistencias, usaremos la propiedad text-size-adjust. Dado que esta propiedad no está estandarizada, se emplea con prefijos.

```
html {
    -webkit-text-size-adjust: 100%;
    -ms-text-size-adjust: 100%;
}
```

2.7. ***Tablas responsive,*** los datos que son presentados en tablas, se convierten un verdadero reto para el diseño responsive. Por lo que requiere realizar cualquiera de las siguientes técnicas:

a. Mostrar/ocultar columnas: se muestran más o menos columnas según el ancho de la pantalla.
b. Tabla con scroll: se trasponen filas y columnas y se aplica un scroll a los datos.
c. Tabla de 2 columnas: la tabla se convierte en una tabla de 2 columnas donde se muestra cada registro individualmente.
d. Tabla de 1 columna: la tabla se convierte en una tabla de 1 columna donde se muestra cada registro individualmente.
e. Tabla con filas inline: la tabla abandona la estructura de rejilla y las celdas de datos se concatenan entre sí, sin dejar de ser una fila.

3. Utilización de media queries

Las Media Queries son una parte muy importante dentro del responsive design y nos permiten escribir estilos que respondan a los cambios al tamaño de la pantalla, la orientación o el medio en el que se visualiza el sitio web.

Las Media Queries nos permite seleccionar con toda precisión el medio de difusión, mediante criterios específicos o combinaciones de criterios. El resultado obtenido de esta búsqueda será de tipo booleano: el valor será verdadero o falso. De este modo podremos seleccionar una hoja de estilo en función de las respuestas obtenidas en la búsqueda.

Los criterios de búsqueda más importantes para la construcción de media queries son los siguientes:

- El **ancho de la zona de visualización**: width. Podemos examinar el ancho de la zona de visualización del navegador. Se permiten las variantes **min-width** y **max-width**. Es la principal expresión para el responsive design. Ejemplo: width: 780px.

- La **orientación** de la pantalla: orientation. Ejemplo: orientation: portrait o orientation: landscape. Resulta bastante práctico para examinar si el usuario usa su tableta táctil verticalmente (portrait) u horizontalmente (landscape).

Como ya se indicó, los media queries devuelven un resultado booleano: solo pueden ser verdaderos o falsos. Si la condición es verdadera, se aplica el estilo; si es falsa, será ignorada.

```
@media (max-width:600px) {
    /*reglas a implementar*/
}
```

Este sencillo ejemplo tiene una media query que se interpreta de la siguiente manera: Cuando el ancho de la pantalla tenga un ancho menor a 600px se aplican las reglas de estilo encerradas dentro de las llaves que abren y cierran la media queries.

Operadores lógicos

Para la construcción de media queries útiles es necesario utilizar múltiples condiciones relacionadas mediante operadores lógicos.

- **Operador and**
  El operador and es usado para colocar juntas múltiples funciones multimedia. Un query básico con el tipo de medio especificado como all puede lucir así:

  ```
  @media (min-width: 700px) { ... }
  ```

  Si se desea quiere aplicar ese query solo si la pantalla esta en formato horizontal, se puede utilizar el operador and y colocar la siguiente cadena:

  ```
  @media (min-width: 700px) and (orientation: landscape) { ... }
  ```

- **Operador or**
  Al escribir media queries el operador or no existe como tal sin embargo es posible conseguir su comportamiento usando condiciones separados por coma.
  Cuando utilizan condiciones separadas por comas y alguno de los queries retorna verdadero, el estilo o la hoja de estilos serán aplicados.
  Por ejemplo, si se quiere aplicar una serie de estilos para un equipo con un ancho mínimo de 700px o si el dispositivo está colocado en horizontal, debemos escribir la siguiente media query:

  ```
  @media (min-width: 700px), (orientation: landscape) { ... }
  ```

- **Operador not**
  Es una negación de una condición. Cuando esa condición no se cumpla se aplicarán las media queries.

Aplicando media queries

Las media queries se pueden aplicar básicamente en dos posibles puntos de la web:

- Al llamar al archivo css, indicando en cada uno las condiciones para cargarlo.

  ```
  <link rel="stylesheet" media="(max-width: 800px)" href="ejemplo.css" />
  ```

- En el propio archivo CSS, formando un apartado donde incluir fragmentos de css a aplicar

```
@media (max-width: 800px) {
    .prueba {
        display: none;
    }
}
```

Creando puntos de cortes mediante media queries

Típicamente las medias queries que se construyen para responsive design emplean como criterio de búsqueda el ancho del viewport. En ese sentido encontraremos tres formas de crear media queries:

- Aplicar solo en resoluciones de menos de X píxeles de ancho:
  ```
  @media screen and (max-width:[ANCHO]px) {}
  ```
- Aplicar solo en resoluciones de más de X pixeles de ancho:
  ```
  @media screen and (min-width:[ANCHO]px) {}
  ```
- Aplicar solo en resoluciones entre X e Y pixeles de ancho:
  ```
  @media screen and (min-width:[ANCHO X]px) and (max-width:[ANCHO Y]px) {}
  ```

El proceso de creación de breakpoints (puntos de corte) puede llegar a ser muy completo por la gran diversidad de dispositivos y dimensiones de pantalla que presentan los dispositivos en el mercado de los dispositivos móviles. Por ello, en lugar de trabajar con complejas matrices de dimensiones de dispositivos vamos a escribir nuestras medias queries de la forma más genérica posibles.

A continuación se le muestra dos plantillas básicas de media queries para desarrollar prácticamente cualquier proyecto web. Estos valores de breakpoint coinciden además con los utilizados por el framework Bootstrap.

- **Pantilla 1:** Diseño responsive a partir del diseño más pequeño (el de móvil)

```
/*Pantallas muy pequeñas  (móviles en portrait de menos de 576px)
No hace falta media-query porque será nuestro diseño por defecto*/

/*Pantallas pequeñas (móviles en landscape de más de 576px)*/
@media (min-width: 576px) { ... }

/*Pantallas medianas (tablets de más de 768px)*/
@media (min-width: 768px) { ... }

/*Pantallas grandes (desktops de más de 992px)*/
@media (min-width: 992px) { ... }

/*Pantallas muy grandes (desktops de más de 1200px)*/
@media (min-width: 1200px) { ...  }
```

- **Pantilla 2:** Diseño responsive a partir del diseño más grande (el de escritorio)

```
/*Pantallas muy grandes  (desktops de más de 1200px de ancho)
No hace falta media-query porque será nuestro diseño por defecto*/

/*Pantallas grandes (desktops de menos de 1200px)*/
@media (max-width: 1199.98px) { ... }

/*Pantallas medianas (tablets de menos de 992px)*/
@media (max-width: 991.98px) { ... }

/*Pantallas pequeñas (móviles en landscape de menos de 768px)*/
@media (max-width: 767.98px) { ... }

/*Pantallas muy pequeñas (móviles en portrait de menos de 576px)*/
@media (max-width: 575.98px) { ... }
```

Probando los diseños responsivos

Es muy común que para probar un diseño responsivo simplemente se cambien el ancho del navegador. Esta prueba, aunque no es incorrecta, si es muy limitada puesto que haciendo eso no es posible determinar la experiencia de navegación que obtendría el usuario en cada tipo de dispositivo.

Una mejor opción sería entonces probar nuestros diseños en diversos tipos de dispositivos para ver si realmente la experiencia de navegación es cómoda.

La mayoría de navegadores incluyen desde su "inspeccionador de elementos" la posibilidad de probar un sitio web en diferentes tipos de dispositivos. Adicionalmente existen extensiones para algunos navegadores que permiten verificar como se verían nuestros sitios en diversos dispositivos móviles. También puedes utilizar plataformas que te permitan visualizar tu sitio web en diferentes tamaños de dispositivos, por ejemplo: http://www.responsinator.com/ o http://responsivetesttool.com/

III. DESARROLLO DE LA PRÁCTICA

INDICACIONES GENERALES

1. Crear una carpeta y colocarle el nombre **N°Carnet_Guia4.** Nota: Cambiar N°Carnet por su número de carnet brindado por la Universidad.
2. Descargar los recursos de la guía, descomprimirlos y colocar las carpetas **img** y **css** dentro de la carpeta creada en el paso anterior.

EJERCICIO 1: Utilizando los media queries

1. Cree un archivo HTML llamado “mediaqueries.html”. Guardar el archivo en su carpeta de trabajo.
2. Escribir la siguiente estructura HTML.

```html
<!DOCTYPE html>
<html lang="es">

<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=0">
    <title>Ejemplo de Media Queries</title>
</head>

<body>
    <div>
        <h1>Probando las medias queries</h1>
        <p>El fondo de esta pagina se mostrará de la siguiente manera:
            <ol>
                <li>Celeste en smartphones</li>
                <li>Naranja en tablet</li>
                <li>Gris en versiones de escritorio</li>
            </ol>
        </p>
        <p>Tipo de pantalla que se muestra
            <span class="smartphones">smartphones</span>
            <span class="tablet">tablet</span>
            <span class="escritorio">escritorio</span>
        </p>
    </div>
</body>

</html>
```

3. Dentro de la carpeta css cree un archivo llamado **mediaqueries.css**
4. Ahora importe la hoja de estilo en su documento HTML, utilice la etiqueta <link> para completar dicha acción.

5. Editemos nuestro archivo **mediaqueries.css**, con algunas reglas generales:

```css
@import url('https://fonts.googleapis.com/css2?family=Bakbak+One&display=swap');

* {
    font-family: 'Bakbak One', cursive;
}

h1 {
    font-size: 2rem;
}

p {
    font-size: 1rem;
    text-align: justify;
}

div {
    background-color: white;
    border-radius: 10px;
    margin: 0 auto;
    padding: 2rem;
    box-sizing: border-box;
}

span {
    display: none;
    font-weight: bold;
    text-transform: uppercase;
    font-size: 1.2rem;
    color: red;
}
```

6. Creemos reglas de diseño para nuestras medias queries que serían aplicadas a dispositivos móviles.

```css
/* Estilo para móviles  */
@media screen and (max-width: 480px) {
    html {
        font-size: 14px;
    }

    body {
        background-color: #03A9F4;
    }

    div {
        width: 95%;
    }

    .smartphones {
        display: inline-block;
    }
}
```

7. Trabajemos para las pantallas correspondientes a dispositivos Tablet.

```
/* Estilos para tablet */
@media screen and (min-width: 481px) and (max-width: 768px) {
    html {
        font-size: 15px;
    }

    body {
        background-color: #FF9800;
    }

    div {
        width: 85%;
    }

    .tablet {
        display: inline-block;
    }
}
```

8. Por último trabajaremos con reglas que se aplicarían a pantallas de tipo desktop.

```
/* Estilos para PC */
@media screen and (min-width: 769px) {
    html {
        font-size: 16px;
    }

    body {
        background-color: #BBBBBB;
    }

    div {
        width: 75%;
    }

    .escritorio {
        display: inline-block;
    }
}
```

Probando el diseño responsivo

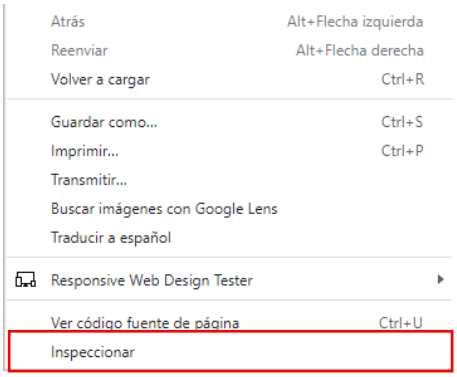
9. Una forma sencilla (pero un poco limitada) de probar los diseños responsivos consiste simplemente en reducir el ancho del viewport del navegador. Pruebe de esta manera la página web construida en el ejercicio anterior y note como cambia el color de fondo una vez que se alcanzan los distintos puntos de quiebre.

Vista en Smartphone

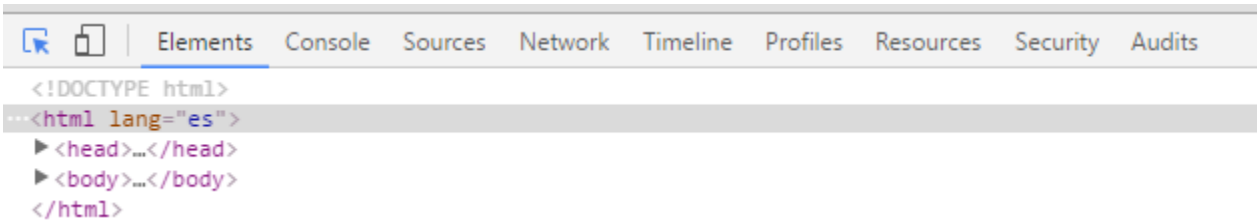
Vista en Tablet

Vista en desktop

10. Los navegadores en su mayoría poseen el “inspeccionador de elementos”, esta herramienta permite probar los diseños responsivos. Para poder disponer de esta herramienta, solo basta en hacer clic derecho sobre el sitio web y luego seleccionar “Inspeccionar”.



En **Google Chrome** para observar estas herramientas basta con abrir el inspeccionador de elementos y seleccionar la opción “Toggle device mode”.



Luego basta con seleccionar alguno de los dispositivos disponibles y observar cómo se visualizaría la página en dicho dispositivo.

EJERCICIO 2: Construyendo un Layout responsivo

1. Para este ejercicio retomaremos el ejemplo de la agencia de viajes de la guía anterior. Los archivos HTML a ocupar son exactamente iguales a los empleados en la guía anterior.
2. Agregue la meta-etiqueta viewport en las tres páginas del sitio (index.html, planes.html y galería.html)

```
<meta name="viewport" content="width=device-width, initial-scale=1.0, user-scalable=no">
```

3. A continuación debe agregar en la cabecera de **sus tres paginas** las llamadas a las hojas de estilo en función del ancho del viewport del navegador.

```
<link type="text/css" rel="stylesheet" href="css/base.css">
<link type="text/css" rel="stylesheet" href="css/movil.css" media="screen and (max-width:480px) ">
<link type="text/css" rel="stylesheet" href="css/tablet.css" media="screen and (min-width:481px) and (max-width:768px) ">
<link type="text/css" rel="stylesheet" href="css/pc.css" media="screen and (min-width:769px) ">
```

Note que la hoja de estilo "base.css" se aplica indistintamente del ancho del viewport del navegador o de cualquier otro criterio. Las hojas de estilo móvil.css, tablet.css y pc.css se aplicarán según se cumpla o no la condición de las media queries (Note que se están usando los mismos puntos de ruptura del ejercicio anterior).

4. Cree dentro de su carpeta css la hoja de estilo llamada "**base.css**". Las reglas de estilo que escribamos en este archivo se van a aplicar a la página independientemente del dispositivo desde el cual se visualice.

5. Incluir las siguientes reglas de estilo dentro del archivo creado en el paso anterior.

```
/* FUENTES */

@import url(http://fonts.googleapis.com/css?family=Didact+Gothic);

*{
	font-family: 'Didact Gothic', sans-serif;
	margin: 0;
	padding: 0;
}

body{
	background-color: #BBB;
	color: #444;
}

h1{
	font-size: 1.3em;
	text-align: center;
	color: #464d52;
}
```

```css
/* PARTE SUPERIOR */
#superior{
	width: 100%;
	padding: 2% 0px;
	background-color: #464d52;

}

/* PIE DE LA PAGINA*/
footer{
	text-align: center;
	margin: 0;
	font-size: .9em;
	line-height: 2;
	clear: both;
	background-color: #464d52;
	color: white;
	width: 100%;
}


/*BOTON DE LA PAGINA INDEX */

.boton{
	background-color:#464d52;
	padding: 7px;
	 margin-top: 10px;
	border-radius: 5px;
	color: white;
	font-size: 0.9em;
	text-align: center;
	vertical-align: middle;
	cursor: pointer;
	transition: background 3s, color 1s;

}

.boton:hover{
	background: #5077c7;

}

.boton a{
	text-decoration: none;
	color: white;
	width: 100%;
	margin: 0px;
	display: block;
}
```

Construyendo el diseño para desktop

Comenzaremos construyendo el diseño para escritorio, este diseño **será prácticamente el mismo diseño fluido que realizamos en la práctica anterio**r.

6. Cree una hoja de estilo llamada **pc.css** y ubique en este archivo las siguientes reglas de estilo:

```
/*    MENU */

nav{
      width: 100%;    /* 960/960*=100%*/
      font-size: 1.4em;

}

nav ul{
      list-style: none;
      margin: 0px 0px 65px 0px;
}

nav ul li{
      float: left;
      margin: 0px;
      padding: 10px 0px 0px 0px;
      border-top: solid 2px white;
}

nav ul li a {
      text-decoration: none;
      padding: .3em .5em;
      margin: 0em .5em;
      display: block;
      color: white;
      transition: color 1s, transform 3s;
}

nav ul li a:hover{
      color: #AAA;
      transform:scale(1.5);
}

nav ul li a#inicio{
      background: url(../img/turismo/home.png) no-repeat 0px 3px;
      padding-left: 40px;
}
nav ul li a#planes{
      background: url(../img/turismo/planes.png) no-repeat 0px 3px;
      padding-left: 40px;
}

nav ul li a#galeria{
      background: url(../img/turismo/galeria.png) no-repeat 0px 3px;
      padding-left: 40px;
}
```

```css
/* PARTE SUPERIOR */
#encabezado{
	margin: 0px auto;
	width: 80%;
	min-width: 650px;/* YA NO REACCIONARA SI LA PANTALLA ES MAS PEQUEÑA QUE 650px*/
	max-width: 960px;/*YA NO REACCIONARA SI LA PANTALLA ES MAS GRANDE QUE 960 px*/
}

#superior #encabezado #logo{
	margin: 2% 0px; /*23/960=2%*/
	width: 21%;	/*200/960= 21%*/
	height: auto;
}

#imgDesc img{
	width: 62.5%; /* 600/960= 62.5% */
	height: auto;
	float: left;
}

#imgDesc{
	overflow: hidden;/*obliga a cubrir a los elementos flotantes*/
}

#descripcion{
	width: 37.5%; /* 360/960=0.375*/
	background-color: #888;
	float: left;
}

#descripcion p{
	margin: 5%; /* 20/360 =5% */
	color: white;
	font-weight: bold;
	line-height: 170%;
	text-align:justify;
	font-size: 0.95em;
}

/*Contenido principal de las paginas */
#contenido{
	width: 80%;
	margin: 0px auto;
	clear: left;
	overflow: auto;
	min-width: 650px;/* YA NO REACCIONARA SI LA CAJA ES MAS PEQUEÑA DE 650px*/
	max-width: 960px;/*YA NO REACCIONARA SI LA CAJA ES MAS GRANDE QUE 960 px*/
}
```

```css
/*   CONTENIDO DE LA PAGINA INDEX */

.plan{
      width: 26.7%; /* 258/960= 27%*/
      padding: 2%; /* 20/960=2%*/
      margin: 30px 1.5%;
      background-color:white;
      border: solid 1px black;
      border-radius: 10px;
      float: left;
      text-align: justify;
}

.plan img{
      width: 100%;
}

/* Primer div, cuarto div,... */
#contenido div:nth-child(3n+1){
      margin-left: 0px;

}

/* Tercer div, sexto div, .... */
#contenido div:nth-child(3n){
      margin-right: 0px;
}

/*   CONTENIDO DE LA PAGINA PLANES */

.planes{
      width: 94%;
      background: white;
      padding: 1% 3%;
}

.planes img{
      width: 100%;
      border-radius: 15px;
}

.planes h1{
      text-align: left;
      font-size: 1.5em;
}

.planes .precio{
      font-size: 1.1em;
      margin: 20px 0px 5px;
      padding-bottom: 15px;
      border-bottom: solid 2px grey;
      font-weight: bold;
}
```

```
/*    CONTENIDO DE LA PAGINA GALERIA*/

#galeria .plan{
      font-size: 1em;
      padding-bottom: 10px;
      margin: 30px 1.4% 10px;
      display: inline-block;
      float: none;
      vertical-align: top;
      text-align: center;
      font-style: italic;

}
```

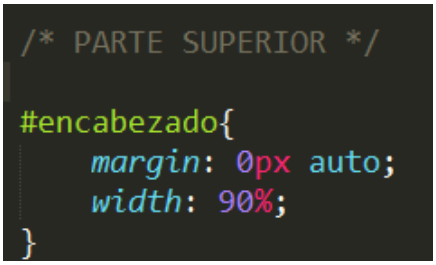
7. Visualice los resultados en su navegador y notara que cuando el ancho del viewport es mayor o igual a 769 pixeles el sitio web funciona correctamente

Construyendo el diseño para tablets

A continuación procederemos a realizar el diseño para tabletas. Salvo un par de cambios en la cantidad de columnas en la que se presenta la información, este diseño será bastante parecido al diseño de escritorio.

8. Cree una copia de su archivo pc.css y nómbrela "**tablet.css**". A continuación se explica los cambios que deben hacerse en las reglas de estilo de ese archivo. Cualquier elemento del que no se presente un cambio, debe asumirse que se mantiene exactamente igual.

9. Dado que el espacio es un poco más limitado en las tablets que en las computadoras de escritorio, debemos replantear el aprovechamiento que se hace de dicho espacio. El primer cambio que realizaremos es que la caja con id "encabezado" se presentará con un ancho del 90% (en la versión desktop era 80%). Además eliminaremos el min-width y el max-width. Esa regla de estilo debería quedar de la siguiente manera:



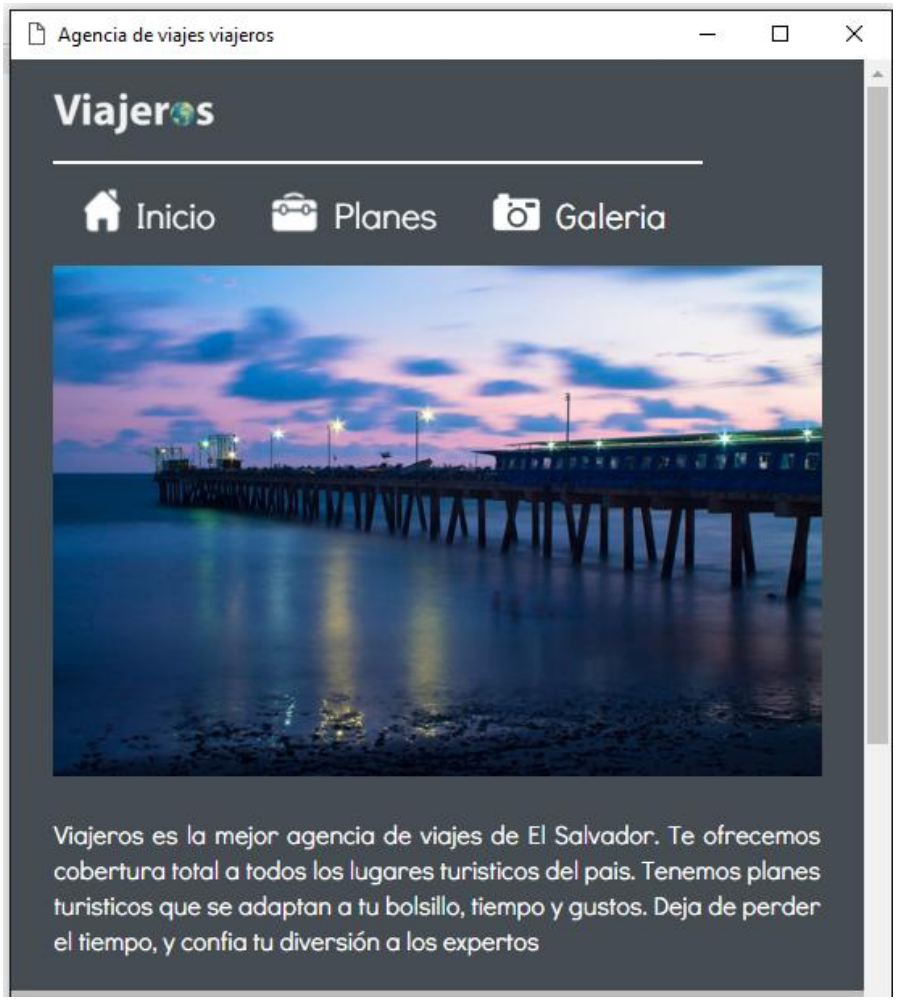
10. A continuación modificaremos la parte superior de la página. En la versión desktop presentamos la imagen y la descripción flotando a la derecha de la imagen. En la versión para tablets no tenemos suficiente espacio para mostrar los elementos a la par, por lo que mostraremos la imagen a ancho completo y la descripción por debajo de la imagen. Para conseguir esto debe modificar las reglas de estilo de los selectores #imgDesc img, #descripcion y #descripcion p de tal forma que luzcan de la siguiente manera:

```css
#imgDesc img{
    width: 100%;
    height: auto;
    float: none;
}

#descripcion{
    width: 100%;
}

#descripcion p{
    margin: 20px 0px 10px 0px;
    color: white;
    line-height: 140%;
    text-align:justify;
    font-size: 1em;
}
```

Ahora el resultado obtenido debería ser similar al siguiente:



Versión para tablets

11. Así como aumentamos el ancho de la caja con id “encabezado” también debemos aumentar el ancho de la caja con id “contenido”. Además se deben eliminar los anchos mínimos y máximos que definimos para la versión desktop. La regla de estilo debe lucir de la siguiente manera:

```css
#contenido{
    width: 90%;
    margin: 0px auto;
    overflow: auto;
}
```

12. A continuación cambiaremos el diseño de la página index.html. En su versión para desktop esta página mostraba su contenido mediante 3 columnas. En el diseño para tablets el contenido se mostrará a una sola columna con la imagen flotando a la izquierda y con el texto a la par de la imagen. Las reglas de estilo de este punto deberían lucir de la siguiente manera:

```css
/*   CONTENIDO DE LA PAGINA INDEX */

.plan{
    width: 94%;
    padding: 1.5%;
    margin: 20px 1.5%;
    background-color:white;
    border: solid 1px black;
    border-radius: 10px;
    float: left;
    text-align: justify;
}

.plan img{
    width: 50%;
    height: auto;
    float: left;
    margin-right: 20px;
}

.boton{
    width: 40%;
    float: right;
}
```

Nótese que se han eliminado las reglas de estilo que tienen los selectores **#contenido div:nth-child(3n+1)** y **#contenido div:nth-child(3n)** puesto que como el diseño es a una sola columna no debemos controlar esos márgenes.

El resultado obtenido debería ser similar al siguiente:

13. La página planes.html presenta su contenido en una sola columna, por tanto no tendremos que realizar ninguna modificación en esta sección.
14. La página **galería.html** presenta su contenido en tres columnas (par la versión desktop). En la versión para tablets conseguiremos que el contenido se muestre en dos columnas. Las reglas de estilo para esta página quedarían de la siguiente manera:

```
/*   CONTENIDO DE LA PAGINA GALERIA*/
#galeria .plan{
    font-size: 0.8em;
    padding-bottom: 10px;
    margin: 10px 1%;
    display: inline-block;
    float: none;
    vertical-align: top;
    width: 44%;
    text-align: center;
    font-style: italic;
}
```

```css
#galeria .plan p{
    clear: both;
}

#galeria .plan img{
    width: 90%;
    height: auto;
    margin: 5px 5%;
}
```

Hasta este punto su diseño para tablets debería de funcionar correctamente. Pruebe todas las páginas del sitio y corrija cualquier error detectado.

Construyendo el diseño para smartphones

A continuación procederemos a realizar el diseño para smartphones. En este diseño se debe ser mucho más cauto en cuanto al aprovechamiento del espacio puesto que este es mucho más limitado en este tipo de dispositivos.

15. Cree una copia de su archivo tablet.css y nómbrela "**movil.css**". A continuación se explica los cambios que deben hacerse en las reglas de estilo de ese archivo. Cualquier elemento del que no se presente un cambio, debe asumirse que se mantiene exactamente igual.

16. Lo primero que haremos es modificar el menú para aprovechar mejor los espacios. **En la versión para móviles preferimos no mostrar los iconos de los elementos del menú.** Los estilos para el menú deberían lucir de la siguiente manera:

```css
/*   MENU */

nav{
    width: 100%;
    font-size: 1.2em;
}

nav ul{
    list-style: none;
    margin: 0px 0px 5px 0px;
}

nav ul li{
    float: left;
    margin: 0px -15px;
    padding: 10px 0px 0px 0px;
    border-top: solid 2px white;
    border-bottom: solid 2px white;
}
```

```css
nav ul li a {
    text-decoration: none;
    padding: .3em .5em;
    margin: 0em .5em;
    display: block;
    color: white;
    transition: color 1s, transform 3s;
}


nav ul li a:hover{
    color: #AAA;
    transform:scale(1.3);
}
```

Nota: En el diseño para móviles no se mostrarán los iconos en los elementos del menú.

17. En la parte superior de la página se ha decidido no mostrar la imagen para dejar más espacio para el contenido de las páginas. Las reglas de estilo para la parte superior deberían lucir de la siguiente manera:

```css
/* PARTE SUPERIOR */


#encabezado{
    margin: 0px auto;
    width: 90%;
}

#superior #encabezado #logo{
    margin: 10px 0px;
    width: 120px;
    height: auto;
}


#imgDesc{
    overflow: auto;
    width: 100%;
}
```

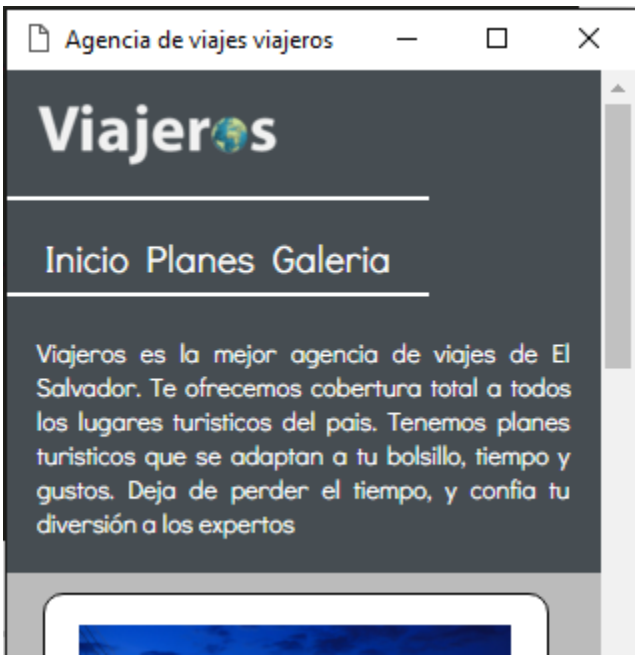
```css
#imgDesc img{
    width: 100%;
    height: auto;
    display: none;
}

#descripcion{
    width: 100%;

}

#descripcion p{
    margin: 20px 0px 10px 0px;
    color: white;
    line-height: 140%;
    text-align:justify;
    font-size: 0.8em;
}
```

El resultado obtenido sería el siguiente:



18. Para el caso de la página index.html el diseño se presentará a una columna como en el diseño para tablets pero con la imagen a ancho completo y el texto debajo de la imagen. Las reglas de estilo de esta página deberían lucir de la siguiente manera:

```css
/*   CONTENIDO DE LA PAGINA INDEX */

.plan{
    width: 92%;
    padding: 10px 2% ;
    margin: 20px 1%;
    background-color:white;
    border: solid 1px black;
    border-radius: 10px;
    text-align: justify;
}

.plan img{
    width: 100%;
    height: auto;
}

.boton{
    width: 92%;
    margin: 5px auto;
}

.plan p{
    font-size: 0.8em;
}

.plan h1{
    font-size: 1em;
}
```

19. Finalmente el contenido de la página galería.html se presentará a una sola columna. Las reglas de estilo para esta página serían las siguientes:

```css
/*   CONTENIDO DE LA PAGINA GALERIA*/

#galeria .plan{
    font-size: 0.8em;
    padding-bottom: 10px;
    margin: 10px  auto;
    float: none;
    width: 90%;
    text-align: center;
    font-style: italic;
}


#galeria .plan p{
    clear: both;
}

#galeria .plan img{
    width: 90%;
    height: auto;
    margin: 5px 5%;
}
```

Pruebe el diseño para móviles y corrija cualquier error detectado.

IV. EJERCICIOS COMPLEMENTARIOS

Usando flexbox y mediaqueries, desarrolle la página de inicio para la presentación de sus guías. Esta página debe ser responsiva y va a asumir un rol de "portafolio estudiantil" que contenga al menos los siguientes elementos:

- Datos personales (nombre, carnet y una breve biografía)
- Datos generales sobre la materia (descripción del curso, contenidos, etc).
- Guías de ejercicios. Por cada guía debe hacerse una breve descripción de su contenido y colocar los enlaces a cada uno de los ejercicios.
- **FODA sobre el desempeño**:
    - **Fortalezas**: ¿Qué habilidades han dominado o mejorado significativamente?
    - **Oportunidades**: ¿Qué han aprendido que les puede servir en otros proyectos o ámbitos profesionales?
    - **Debilidades**: ¿Qué áreas del desarrollo web sienten que necesitan seguir trabajando?
    - **Amenazas**: ¿Qué obstáculos encontraron (personales o técnicos) que podrían haber impactado su progreso?