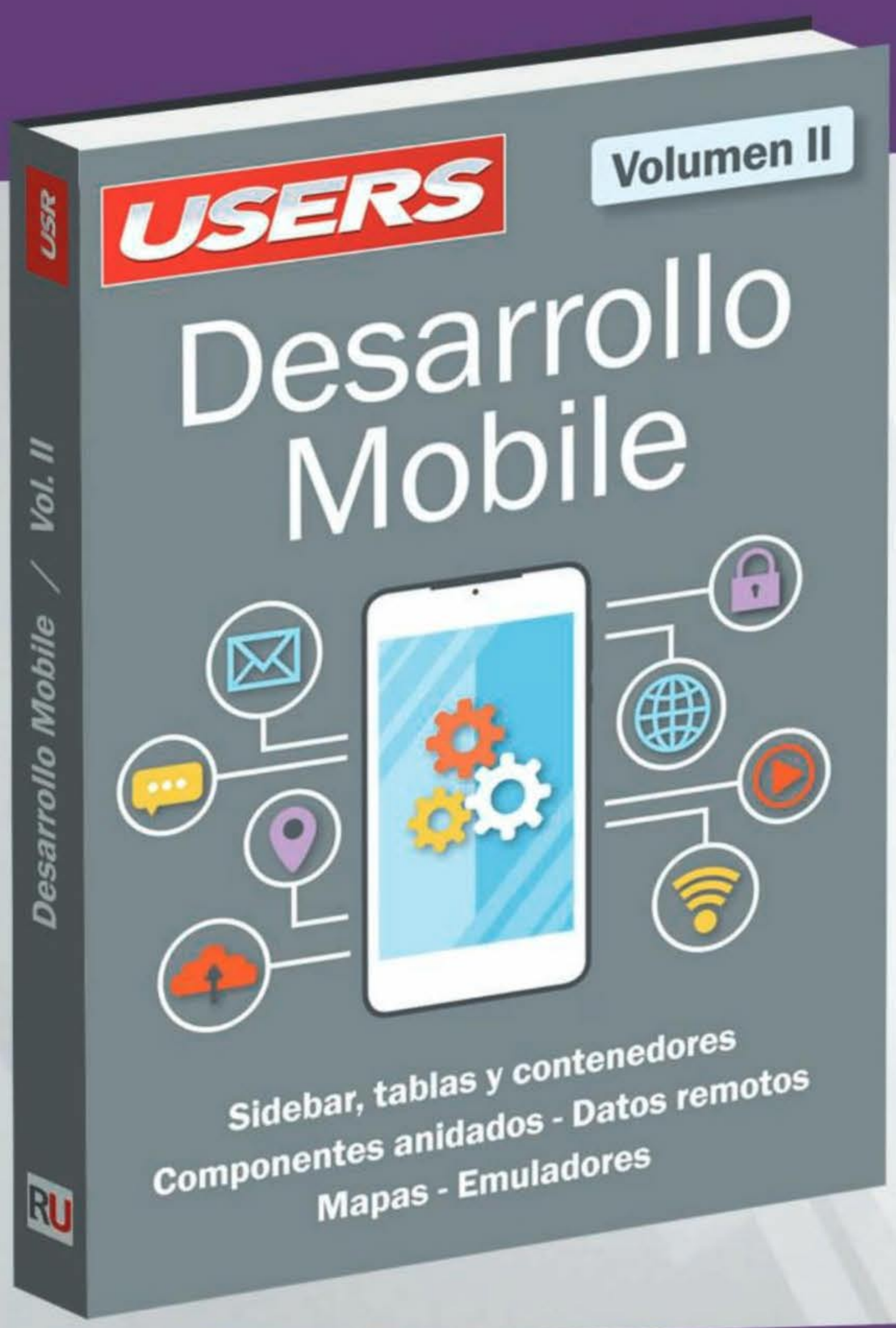
USERS

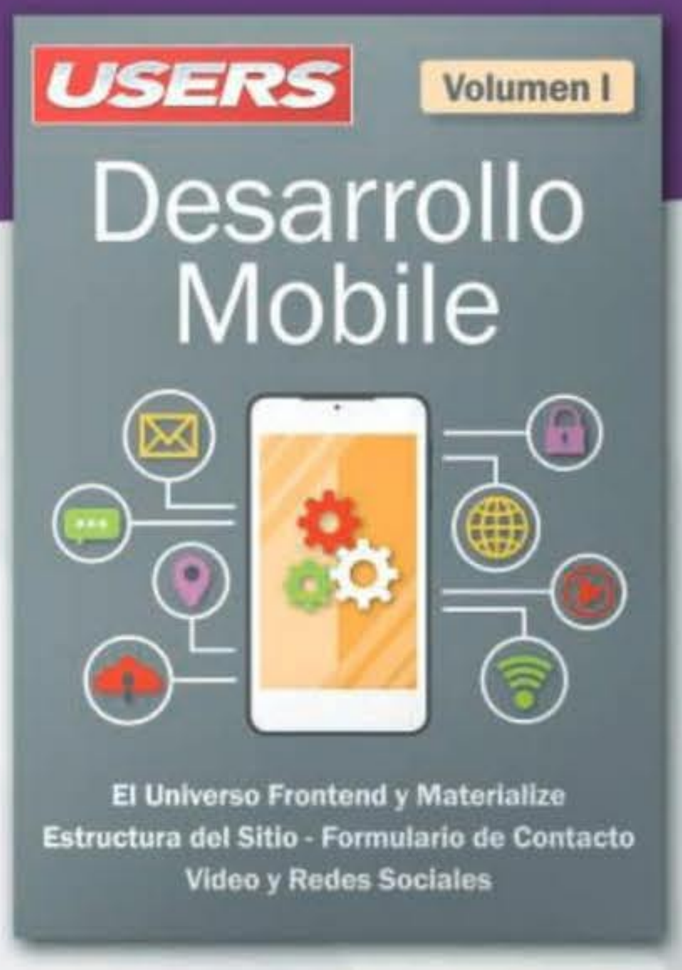
Volumen I

Desarrollo Mobile

El Universo Frontend y Materialize
Estructura del Sitio - Formulario de Contacto
Video y Redes Sociales

CURSO DE DESARROLLO MOBILE





Diseña aplicaciones web Mobile First, desde una web estática hasta 100% dinámica. Convierte tu web en PWA, la próxima generación de aplicaciones móviles y de escritorio.

Desarrollo Mobile

El Universo Frontend y Materialize
Estructura del Sitio - Formulario de Contacto
Video y Redes Sociales

Título: Desarrollo Mobile - Vol. I / **Autor**: Fernando Omar Luna
Coordinador editorial: Miguel Lederkremer / **Edición**: Claudio Peña
Diseño y Maquetado: Marina Mozzetti / **Colección**: USERS ebooks - LPCU292

Luna, Fernando O.
Desarrollo mobile 1 / Fernando O. Luna. - 1a ed . - Ciudad Autónoma de Buenos Aires : Six Ediciones, 2019. Libro digital, PDF - (Desarrollo Mobile ; 1)

Archivo Digital: descarga y online
ISBN 978-987-4958-14-3

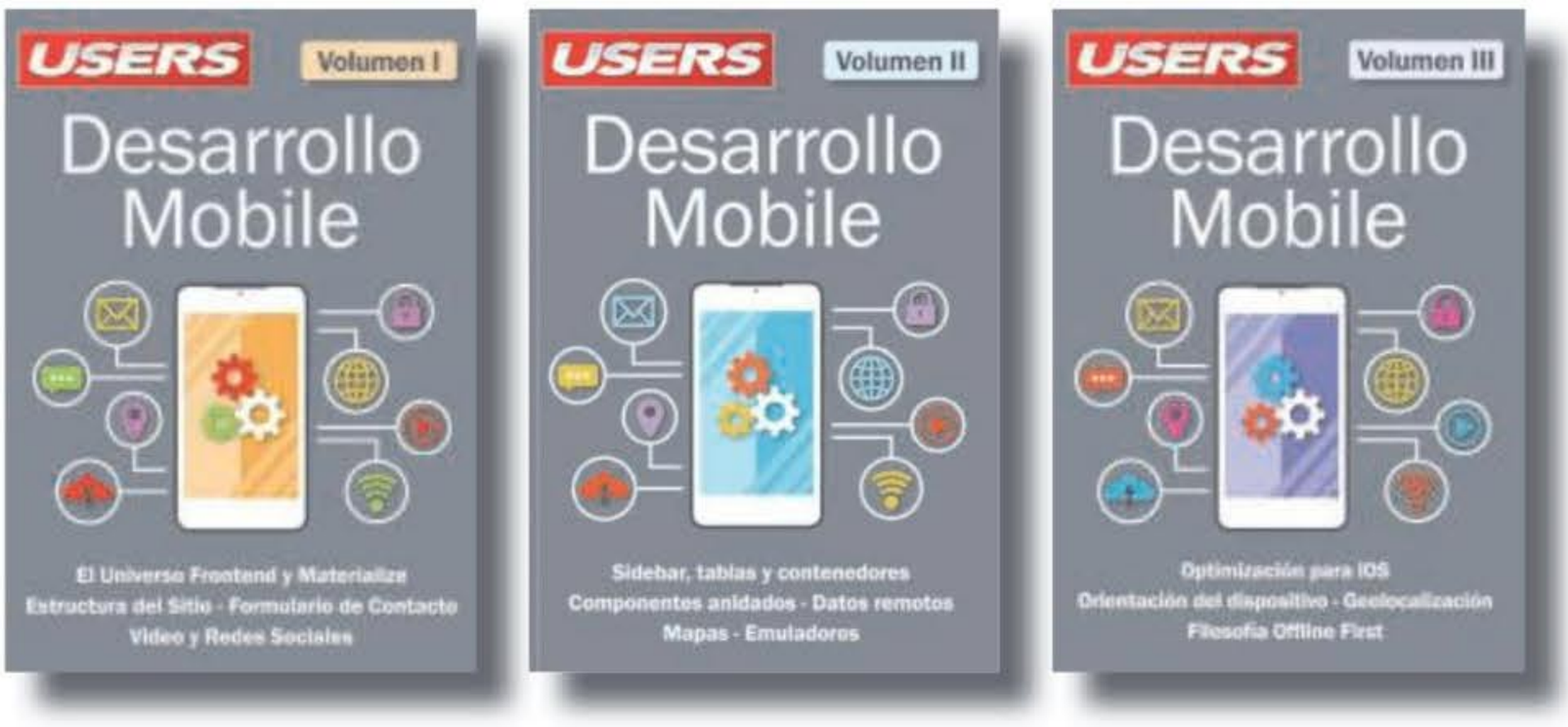
1. Diseño de Software. 2. Aplicaciones Web. I. Título.
CDD 005.384

Acerca de este curso

Este curso se divide en tres e-books. En el **primer** volumen conoceremos Materialize CSS, y comenzaremos a ver cómo implementar la barra de navegación, los títulos, los párrafos y las imágenes en el diseño de un sitio web adaptado para móviles. Luego profundizaremos en la manera de integrar audio y video en nuestros desarrollos, y en la forma de sumar contenido desde y hacia las diferentes redes sociales más populares a la fecha.

En el **segundo** volumen de la colección potenciaremos la barra de navegación, y la integraremos en modo lateral, tal como la utilizan las aplicaciones nativas. Luego aprenderemos las diferentes opciones de botones que podemos aplicar con Materialize, cómo integrar mensajes de tipo tooltip y toast, cómo integrar mapas dinámicos, de qué forma leer y mostrar datos utilizando JSoN, y cómo explotar al máximo las herramientas para desarrollador de Google Chrome y los distintos emuladores móviles del mercado.

En el **tercer** volumen aprenderemos a acondicionar nuestros desarrollos web adaptándolos a las diferentes plataformas móviles. Veremos cómo integrar metatags para la plataforma iOS, qué nos ofrece Onsen.io, cómo convertir nuestra web en una PWA e instalarla en Android e iOS, y de qué modo hacer una web 100% desconectada de Internet. También veremos cómo optimizar las imágenes integradas en una web y la forma de ofrecer una experiencia 360ª con imágenes panorámicas.



Prólogo

La Web es todo. Desde ella no solo nos informamos, sino que también compramos productos y servicios, pagamos impuestos, consumimos películas y música, realizamos reservas en restaurantes y volcamos opiniones sobre lugares visitados. Además leemos mails, ponemos recordatorios en un calendario e interactuamos varias veces al día con las redes sociales, entre un sinfín de cosas más.

Y quienes desarrollamos software sabemos que la Web fue, es y será durante muchísimo tiempo más el puntapié inicial de nuestro día a día. Cada vez estará más comprometida con nosotros y, a su vez, más integrada; y todo esto seguirá ocurriendo desde nuestro bolsillo, desde nuestro dispositivo móvil. Por eso, los invitamos a recorrer la serie de e-books que integran esta obra, para conocer los principales secretos del desarrollo web para dispositivos móviles y saber cómo encarar el diseño e integración de las PWA, la próxima generación de aplicaciones móviles y de escritorio, que ya lleva un tiempo conquistando las pantallas móviles.

Autor

Fernando Omar Luna

Es Analista de Sistemas y Technical Writer. Lleva más de 25 años desarrollando software para diferentes plataformas, y algo más de una década colaborando como autor de publicaciones técnicas orientadas a la programación y las nuevas tecnologías. Su pasión se divide entre la programación, la electrónica y la educación, siendo este último el motivo por el cual se encuentra cursando un profesorado que le permita ejercer profesionalmente su pasión por el desarrollo del software.

Contenido

01 **Introducción al desarrollo frontend y Materialize.....6**

Perfil de un frontend developer / Visual Studio Code / Desarrollo Web / HTML5 / CSS / JavaScript / Ejemplos prácticos

El universo Material Design.....14

Donde profundizar más sobre MD puro / Materialize CSS / Estructura de un proceso Materialize CSS / Estructura HTML / Integración de CSS, fuente tipográfica y JS / Iconografía / Verificar la funcionalidad / CSS Adicional / JavaScript adicional / Personalizar la fuente tipográfica / Más información sobre tipografías / Clases CSS

02 **Barras de navegación.....26**

Navbar / Una barra de navegación funcional / Insertar un logo gráfico / Uso de clases según display / Ejerecicio práctico: Barra de menú superior responsiva / Testear el comportamiento responsivo / Redimensionar la página / Optimización del menú para dispositivos móviles / Material Icons / Testear su comportamiento / Ejemplos prácticos

Footer: construir un pie de página funcional.....34

Diseño del footer / Fijar el footer / Agregar contenido al pie de página

03 **Estructura y estilos Web.....40**

Tipografías / Tipografías Web / Google Fonts / Blockquotes (énfasis e párrafos) / Flow Text / Párrafos editables (content editable) / Imágenes: la clase materialboxed / Breadcrumbs (jerarquía de navegación) / Pagination (estructurar el cúmulo de información)

04 **Formulario de contacto.....50**

Estructura de los text input de Materialize CSS/ Agregar íconos a cada campo / Diseño de un formulario / Validación de datos / Mensajes de confirmación o error personalizados / Botón de envio de formulario / Metatag form / El servicio Mailthis.to / Ejercicio práctico: diseño de una web corporativa / Pautas complementarias para el diseño del ejercicio / Prueba del sitio web en dispositivos móiles.

05 **Audio y Video.....64**

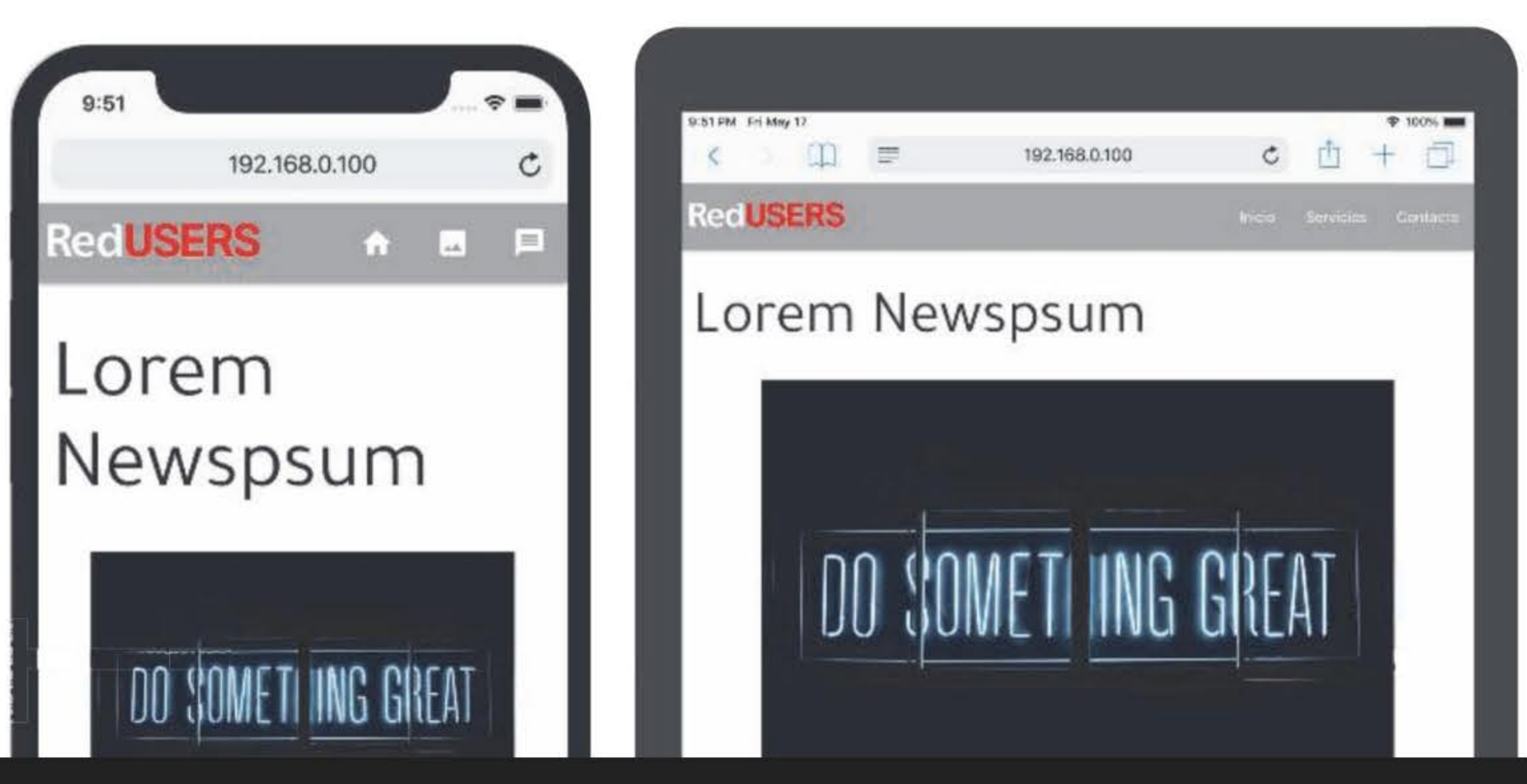
El elemento <audio> / Sintaxis / Propiedades del elemento audio / Controlar el audio desde JavaScript / Enteneder JavaScript / Cómo animar el vúmetro / Función de variables declaradas / Manejo de CSS desde JS / Pausar la reproducción / Visualización según el navegador web / Control multimedia integral desde JS / Eliminar el atributo disabled / Cargar el archivo de audio / Activar los botones de comando / Ejercicio práctico: potenciar el reproductor MP3

El elemento <video>.....82

Mensaje de no compatibilidad / Atributo controls / Atributo autoplay / Atributo loop / Manos a la obra / Ancho del video / Poster / CSS background-color / Múltiples formatos de video / Formatos de video soportados según el navegador / Media Types / Propiedades y atributos del elemento / Picture in Picture / Picture in Picture y JavaScript / Video tracks / Estructura de los archivos VTT / Notas adicionales al archivo / Visualiazación de subtítulos / Múltiples idiomas / Prerequisitos para tener en cuenta / Material de referencia / Ejercicio práctico: video multi-idioma.

06 **Integración de Redes Sociales 96**

Ventajas de la integración de plugins sociales / Sistema de comentarios de Facebook / Requisitos para utilizar Facebook Developer / Configurar nuesto plugin social / Personalizar el lenguaje / Resultado final de comentarios / Total de comentarios al inicio de la publicación / Plugin de páginas / Visualizar íconos de amigos / Cambiar el idioma / Compartir videos de Facebook / YouTube: cómo embeber videos / Insertar publicación de Instagram / Integrar contendo de Twitter / Recursos sociales



01 Introducción al desarrollo frontend y Materialize

Nos zambullimos en el ecosistema del desarrollo de aplicaciones web, repasando las características que hacen a un desarrollador frontend, las herramientas necesarias y las tecnologías actuales requeridas para desenvolvernos en este campo.

El ecosistema web ha cambiado desde su nacimiento, hace casi tres décadas, hasta hoy. Las tecnologías avanzaron a pasos agigantados y, en este espacio de la gran línea de tiempo que enmarca al desarrollo web, nos encontramos en un punto en el cual el acceso a Internet se está volviendo casi transparente para nosotros.

El uso de un navegador web desaparece lentamente, y la disponibilidad de conexión a Internet 24/7 también está siendo condicionada por las innovadoras tecnologías actuales, como son las **Progressive Web Apps**.

TARJETA DE NOTICIAS

TARJETA DESDE UNA TARJETA

GOOGLE CHROME

Perfil de un frontend developer

Definimos a un desarrollador frontend como aquel que se ocupa del aspecto visual que tienen las webs. Y cuando mencionamos el aspecto visual, no solo significa el manejo de **HTML5** y **CSS**. También se incluye aquí el lenguaje **JavaScript**, los frameworks más próximos a él y las capas de desarrollo que existen para potenciar a CSS. Respecto a JavaScript, abarca además la interacción con datos dinámicos que se visualizarán en pantalla, producto del acceso en capas hacia una API o base de datos.

Frontend developer	Core	Otras capas detrás del Core
		
		 
		  

Y, en cuanto al aspecto visual en sí, HTML5 y CSS no se comportan por igual en cualquier navegador, lo cual implica el gran desafío de poder mantener una estética similar que contemple el compendio de browsers y motores de navegación que existen en el mercado informático.


KHTML / Konqueror


Internet Explorer


Dolphin browser


Chomium


Microsoft Edge


Apple Safari


Maxthon browser


Google Chrome


Samsung browser


Opera browser <7


UC browser


Vivaldo browser


Epiphany browser


Opera browser 8>


Tor browser


Mozila Firefox

Visual Studio Code

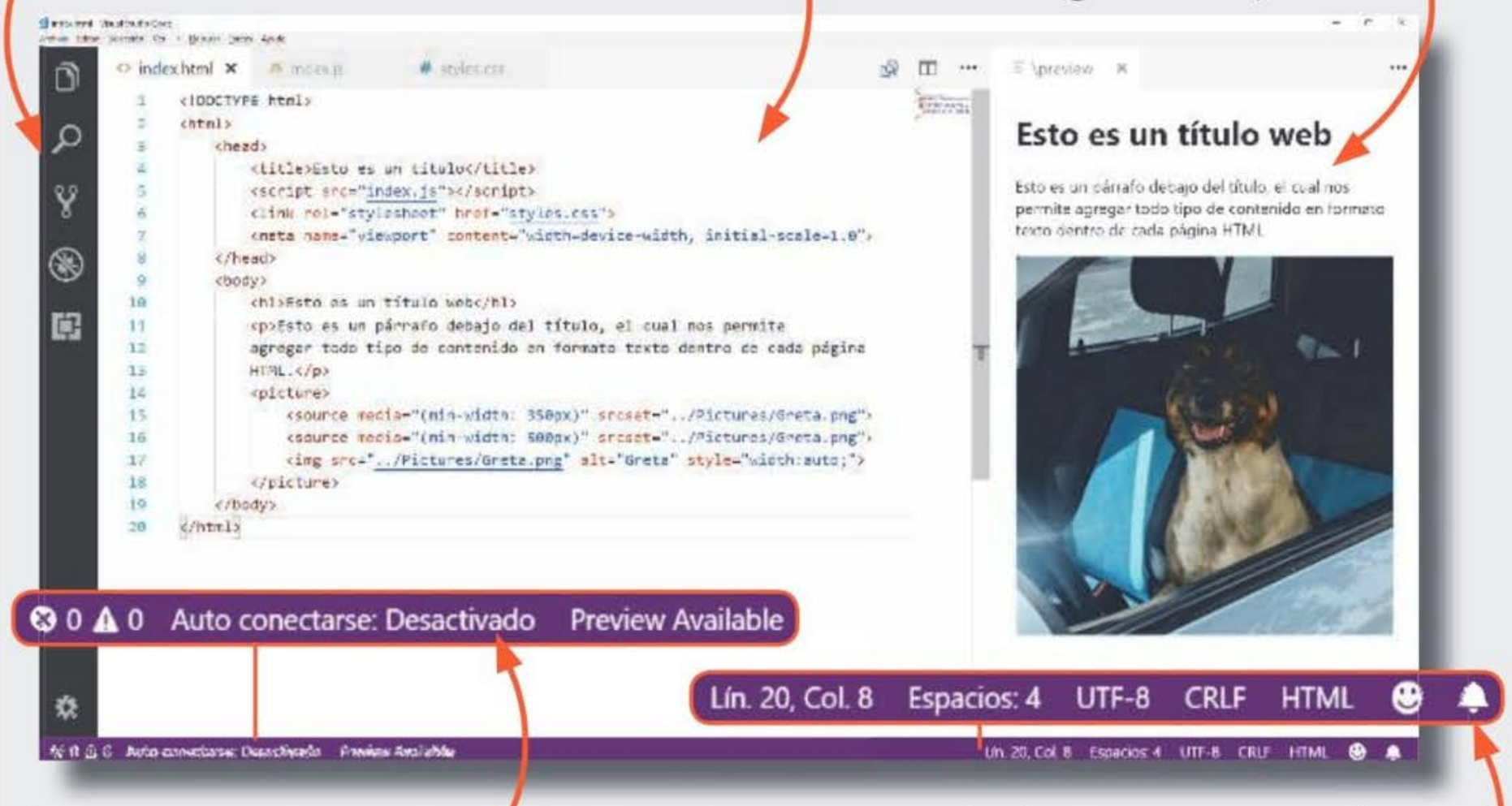
Hace unos años Microsoft lanzó **Visual Studio Code**, un IDE para desarrollo rápido basado en **Atom**. Este IDE será el que utilizaremos a lo largo de la obra para realizar todos los ejercicios relacionados. Veamos a continuación una Guía Visual para saber cómo interactuar con él.

GUÍA VISUAL 1

La barra lateral nos permite:
-Abrir o crear proyectos
-Buscar archivos o carpetas
-Acceder a sistemas de control de código
-Acceder a la consola de depuración
-Ver o instalar extensiones

En el panel central visualizaremos los archivos de código, scripting e imágenes relacionados con nuestro proyecto. Podremos editar los archivos de código, guardarlos y/o renombrarlos.

Mediante extensiones como HTML5 preview, podremos obtener una vista previa de los archivos HTML que editemos en nuestro proyecto, visualizar parcialmente el comportamiento CSS y, de manera eventual, ver la ejecución o respuestas del código JavaScript.



Los errores y advertencias que vayan surgiendo sobre nuestro código serán alertados a través de estos iconos. Podremos desplegar la ventana de depuración y hasta llegar a la línea de bug del archivo en cuestión.

Mediante la combinación de teclas Ctrl + y/o Ctrl -, podemos aplicar zoom sobre el código fuente que trabajamos, para así obtener una mejor visualización.

Aquí podemos ver en qué línea y columna de código nos encontramos, los espacios de tabulación, el código de idioma del archivo que manejamos, el tipo de archivo con el que estamos trabajando, y otras tantas advertencias y notificaciones propias de este IDE.

La descarga e instalación de Visual Studio Code puede realizarse desde su sitio oficial: code.visualstudio.com. Está disponible para Windows, Mac y Linux.

Desarrollo web

Como bien comentábamos al inicio de este capítulo, el desarrollo web concentra una gran porción de la torta estadística relacionada a la programación. En particular, JavaScript se ubicó como el tercer lenguaje de programación más demandado en el año 2018. Y, para poder demostrar todo su esplendor, en la mayoría de los desarrollos requiere combinarse con HTML y CSS, para elaborar una solución de software basada en la Web.

HTML5

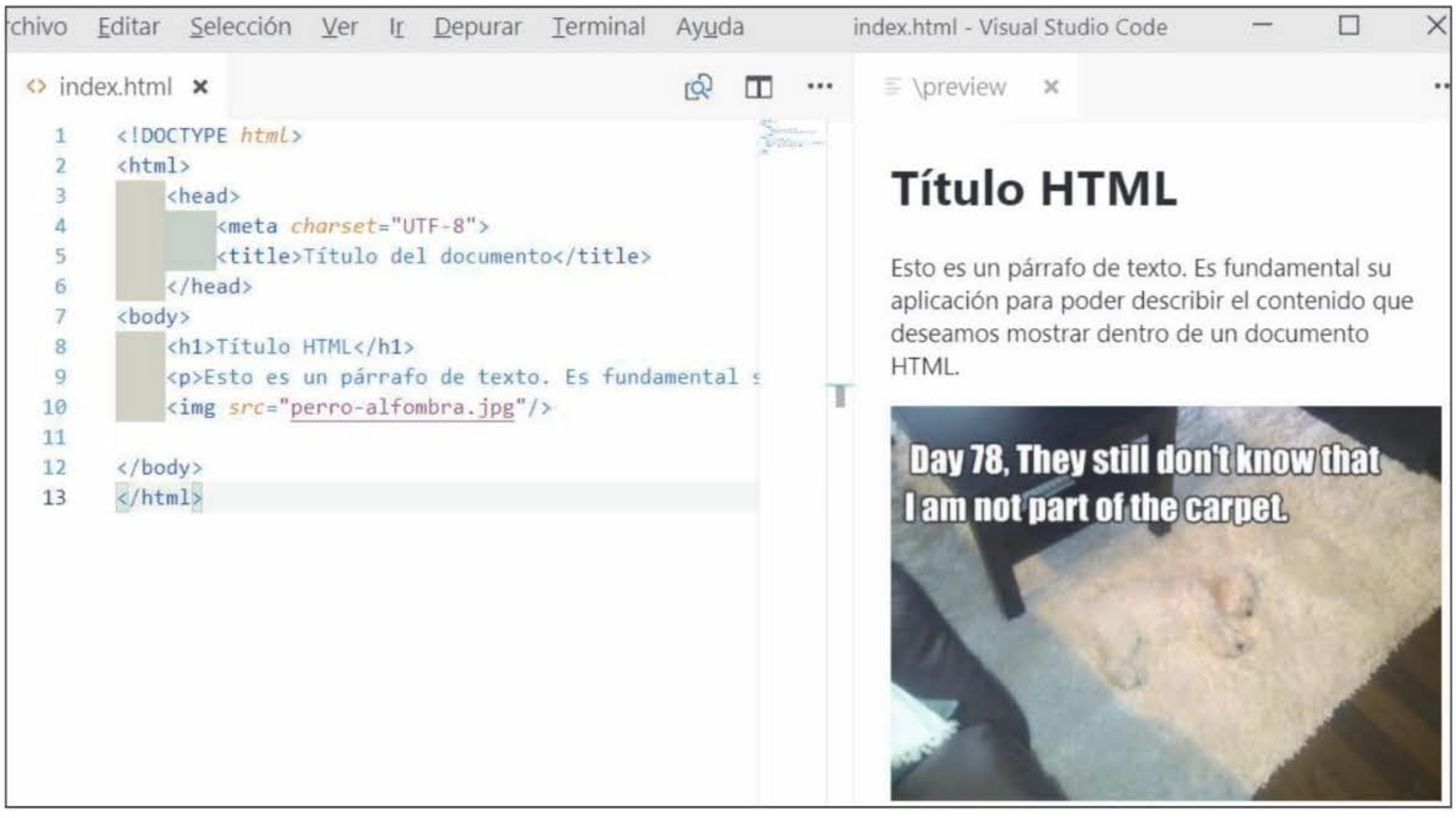
El lenguaje de marcado de hipertexto, o HTML, nació durante el primer lustro de los años 90, para darle vida al contenido de Internet de una manera amigable y legible para cualquier persona. Este lenguaje evolucionó pasando por varios estados, hasta llegar a lo que todos conocemos como HTML5. Si bien este no concibe la línea original evolutiva del lenguaje HTML, fue mucho más aceptado por los motores de navegadores web y, en menos de una década, se convirtió en el lenguaje de marcado web por excelencia.

La adaptación de su estructura de tags a una estructura de tags semántica, de muy fácil entendimiento y aprendizaje, fue la que aportó mucho a esta evolución. Los cambios significativos que trajo en la estructura de títulos, párrafos, imágenes, videos, archivos de audio, y otros contenidos que conforman la Web, son los que realmente facilitaron su entendimiento para el programador y el renderizado por parte del motor de navegación web. Veamos a continuación un ejemplo del lenguaje:

```
01 <!DOCTYPE html>
02 <html>
03     <head>
04         <meta charset="UTF-8">
05         <title>Título del documento</title>
06     </head>
07     <body>
08         ...
09     </body>
10 </html>
```

La estructura principal se divide entre **<head></head>** y **<body></body>**. **!Doctype html** se usa para definir al documento como tal, y así acelerar su renderizado. Todo lo que va encerrado entre **<head>** corresponde al título, la codificación del lenguaje del documento, y los archivos **CSS** y **JS** relacionados. Finalmente, todo lo que se encierra entre **<body>** corresponde al documento HTML en sí, que vemos desplegado en la pestaña del navegador web.

Aquí encontramos títulos, tablas, párrafos, imágenes, videos y cualquier otro contenido HTML visual.



CSS

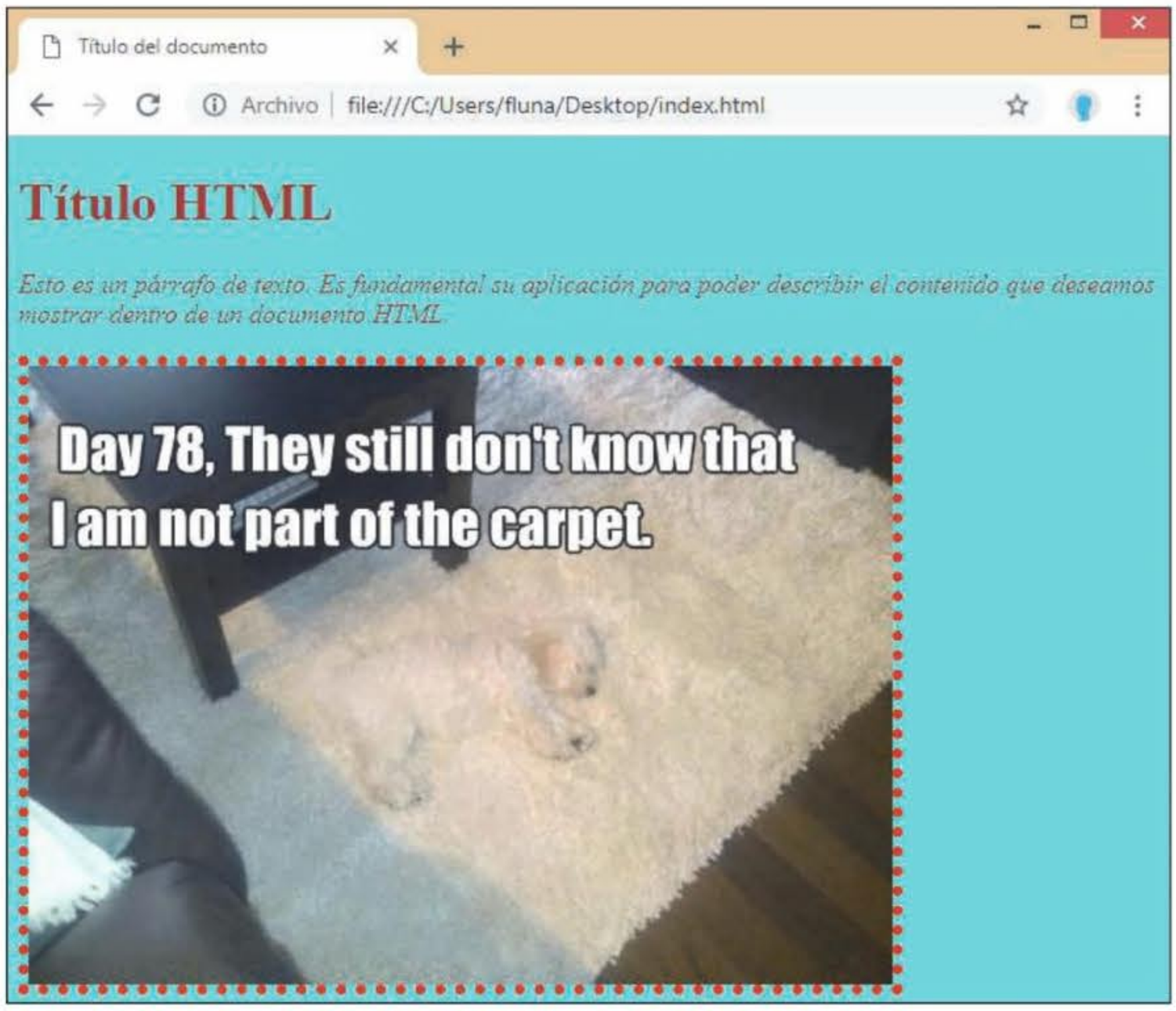
La función de CSS es reforzar la estética de los componentes mayormente visuales que conforman un documento HTML. Su nombre proviene del acrónimo de **Cascade StyleSheet** u **hojas de estilo en cascada**. La sintaxis es muy fácil de aprender, y puede aplicarse de manera general a un documento HTML, como así también de modo individual a cada uno de los componentes de este.

```
body {
    background-color: cyan;
    font-display: Arial, Sans serif;
    color: brown;
}

h1 {
    font-size: 30;
}

p {
```

```
    font-style: italic;
}

img {
    border-style: dotted;
    border-width: 6px;
    border-color: red;
}
```

En este bloque de código utilizamos CSS para modificar las diferentes propiedades de los componentes HTML. Entendemos por propiedades al fondo de página, la tipografía, el tamaño de esta, su color, su estilo, los bordes de un objeto, su ancho y color, entre otras tantas posibilidades más. Para vincular una hoja de estilo CSS a un documento HTML, tenemos que editar este último, agregando, dentro del bloque **<head> </head>**, la referencia hacia el archivo CSS. Veamos a continuación un ejemplo del código:

```
...
<link rel="stylesheet" type="text/css" href="index.css">
</head>
```



En la siguiente figura, observamos el resultado de integrar el ejemplo CSS descripto con anterioridad.

JavaScript

Este lenguaje nació a mediados de los años 90. Si bien al principio fue muy criticado porque consumía muchos recursos de hardware cuando cargaba animaciones o funciones complejas, hoy en día, con el gran avance en procesamiento y velocidades de comunicación, JavaScript ha ganado un protagonismo casi impensado en el terreno web.

Su forma de utilizarse puede ser directamente dentro del código HTML a través del tag **<script></script>**, o en un archivo separado (como los documentos CSS), vinculándose a la página web dentro del apartado <head>. Veamos un ejemplo de este:

```
<script>
    alert("Hola. Esto es JavaScript.");
</script>
```

index.html ●

```
1   <!DOCTYPE html>
2   <html>
3       <head>
4           <meta charset="UTF-8">
5           <title>Titulo del documento</title>
6           <link rel="stylesheet" type="text/css" href="index.css">
7       </head>
8   <body>
9       <h1>Título HTML</h1>
10      <p>Esto es un párrafo de texto. Es fundamental su aplicación para poder describir el con
11      <img src="perro-alfombra.jpg"/>
12      <script>
13          alert("Hola. Esto es JavaScript.");
14      </script>
15  </body>
16  </html>
```

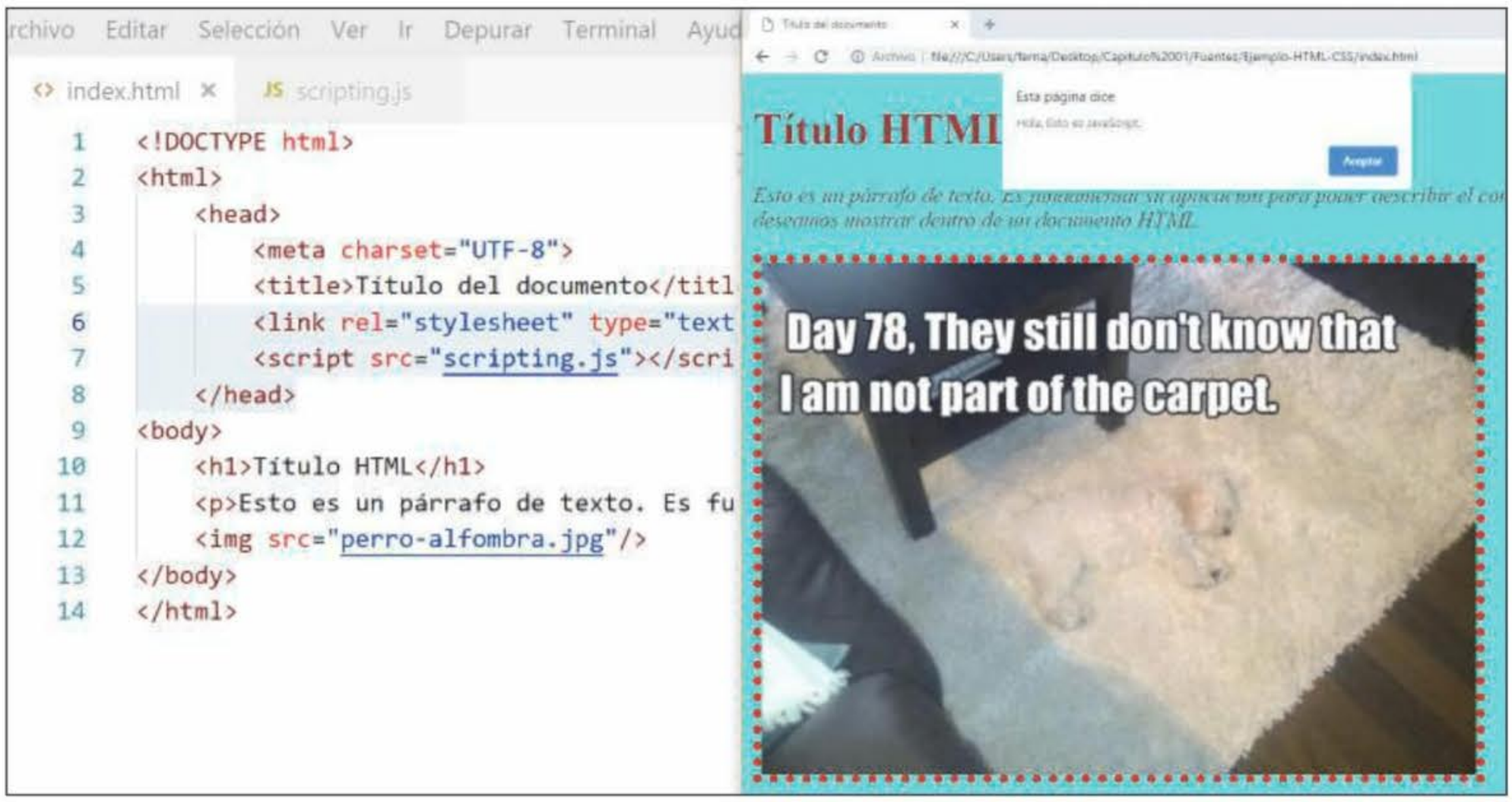
En la figura anterior se muestra cómo se integra JS dentro del código HTML. Veamos ahora cómo realizar lo mismo, integrando JavaScript en un archivo externo:

Archivo **scripting.js**:

```
document.addEventListener("DOMContentLoaded", function() {
    alert("Hola. Esto es JavaScript.");
})
```

Bloque de código en el documento HTML:

```
  ...
  <link rel="stylesheet" type="text/css" href="index.css">
  <script src="scripting.js"></script>
</head>
```



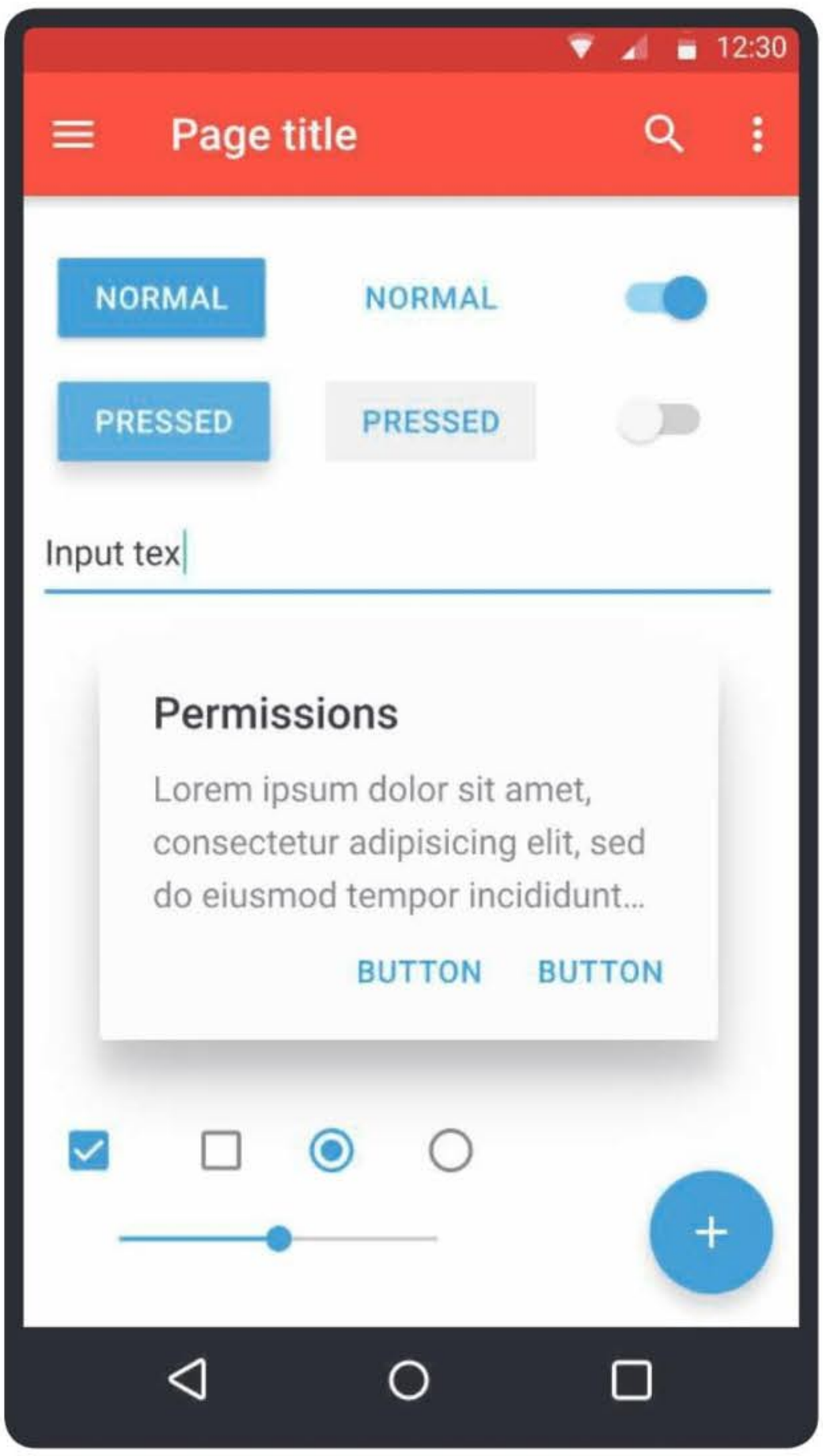
Como podemos notar, casi que no hay diferencia entre utilizar JS en el desarrollo web, dentro del código HTML, o hacerlo en un archivo por separado. Lo mejor de escribir el código de esta última forma es mantener una estructura clara de trabajo y no tener que estar modificando los archivos principales de cada proyecto.

Ejemplos prácticos

Para ver en acción lo que hemos explicado, es posible acceder al repositorio de archivos que acompaña a esta obra, y descargar: **Ejemplo-HTML.zip, Ejemplo-HTML-CSS.zip y Ejemplo-HTML-CSS-JS.zip**. Allí se representa cada uno de los casos abordados hasta aquí.

01 El universo Material Design

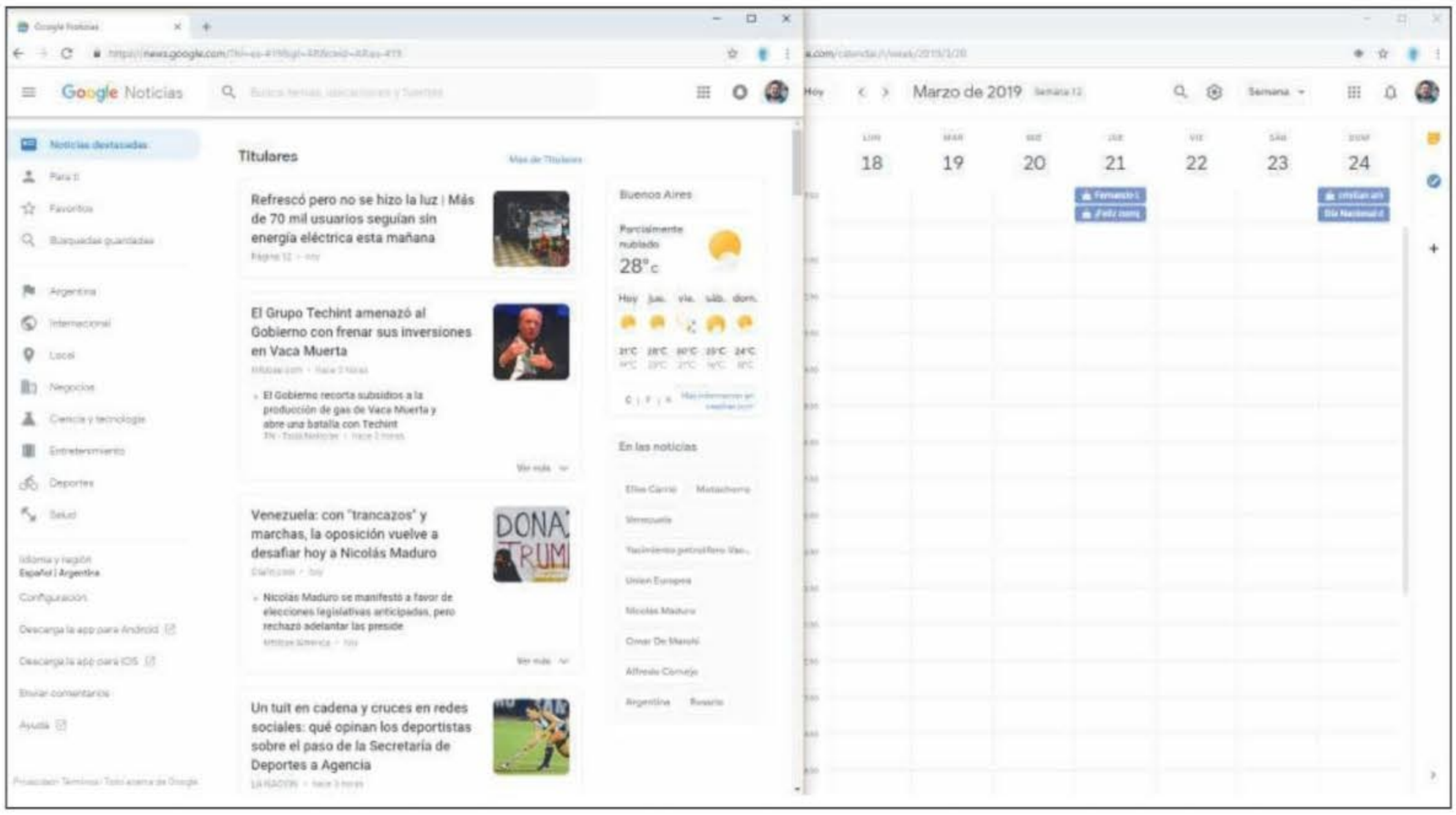
Comencemos a transitar por el camino del diseño de web apps con Materialize CSS. Conozcamos sus orígenes, las distribuciones más populares y cómo sacar partido de esta fabulosa tecnología.



Con la llegada de **Android Lollipop** en 2014, Google presentaba en su conferencia **I/O** una normativa de diseño focalizada en el estilo visual de su sistema operativo móvil, bautizada **Material Design**. Esta llegó para reemplazar la antigua estética de **Android 4** y siguientes.

Esta normativa, completamente visual y llena de patrones de diseño en cuanto a curvas, sombras, fuentes, animaciones e iconos para Android, alcanzó una gran popularidad, lo cual hizo que, con el correr de los años, terminara usándose no solo en sus aplicaciones nativas y sistema operativo, sino también en el resto de las herramientas web de la firma.

La gran aceptación de Material Design llevó a generar diversas alternativas en el mercado que pudiesen ser aprovechadas tanto por los desarrolladores de la plataforma Android Studio, como por quienes llevan adelante aplicaciones móviles híbridas, con **Phonegap**, **Apache Cordova o Ionic**,

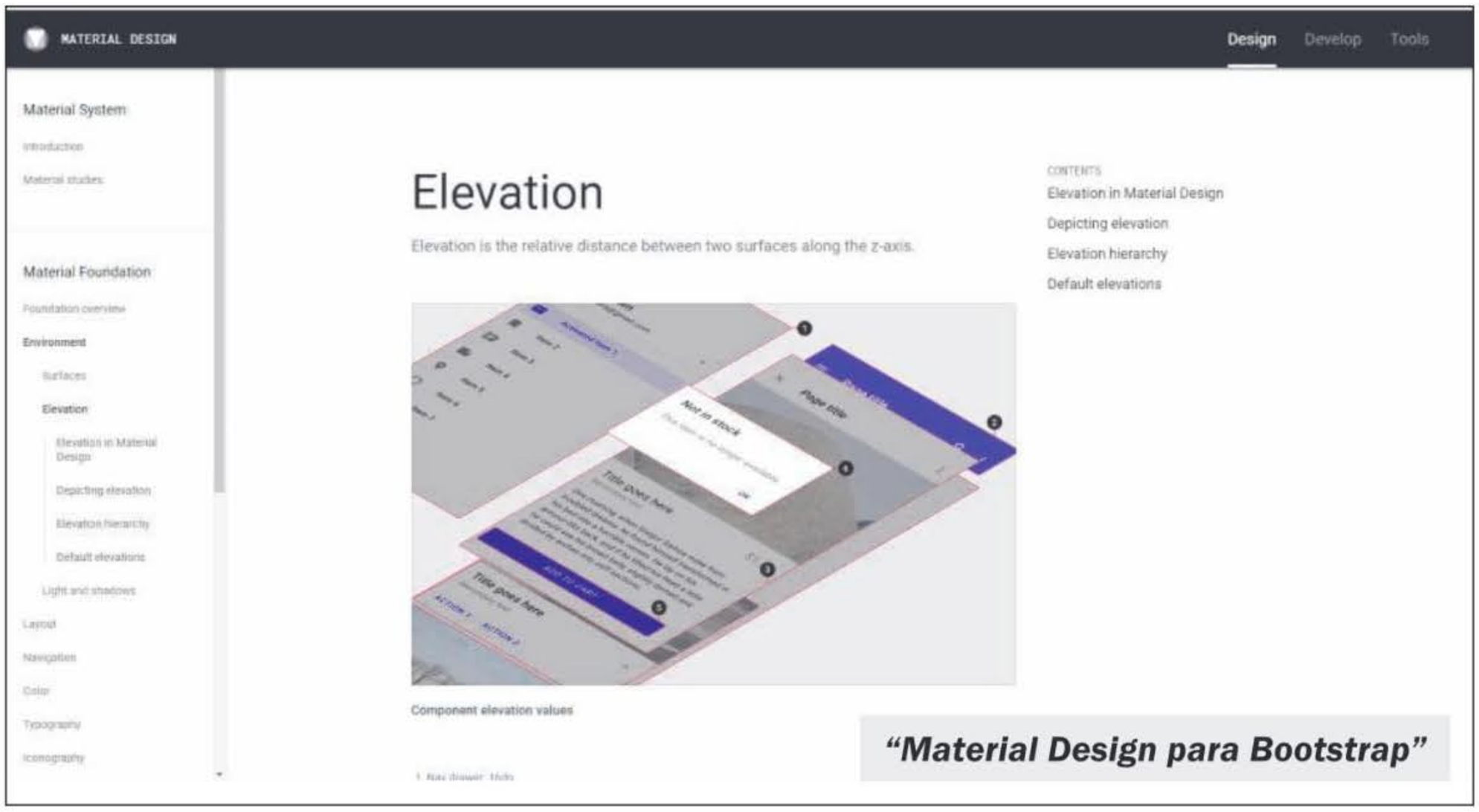


como también por aquellos desarrolladores web que buscaban innovar sus sitios, poniendo en ellos un "estilo Google".

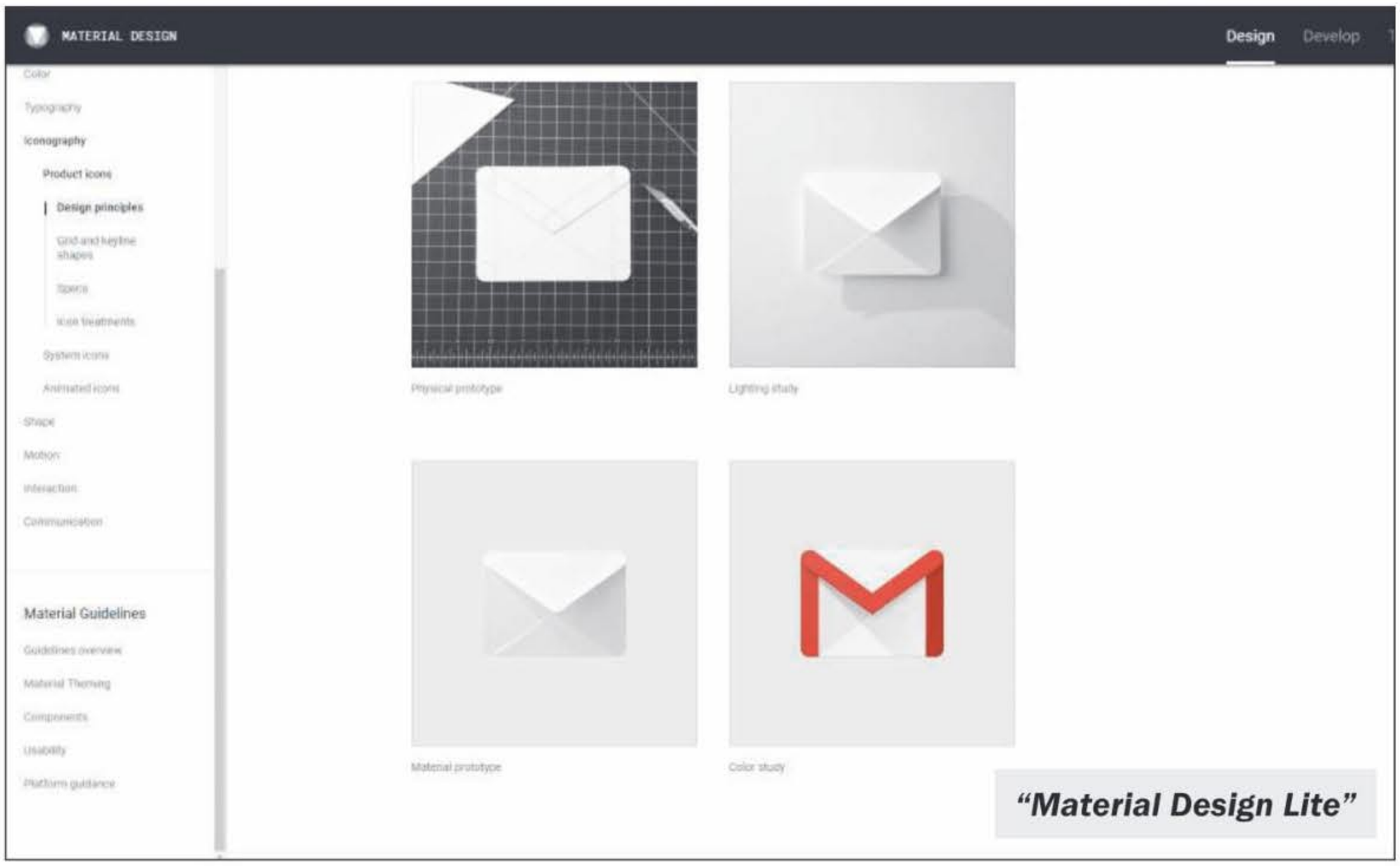
Si hacemos una recorrida por la galería de aplicaciones que dependen de Google, podremos apreciar que todas ya cuentan con el look and feel propio de Material Design. Finalmente, el tiempo logró que una idea pensada para el terreno Mobile terminara acaparando el terreno web e, incluso, el ecosistema iOS.

Dónde profundizar más sobre MD puro

La principal fuente de contenido respecto a la normativa de MD puede encontrarse en los sitios https://design.google y https://material.io/design. En ellos se ofrece la información concerniente a todos los comunicados oficiales y datos técnicos sobre los patrones de diseño.



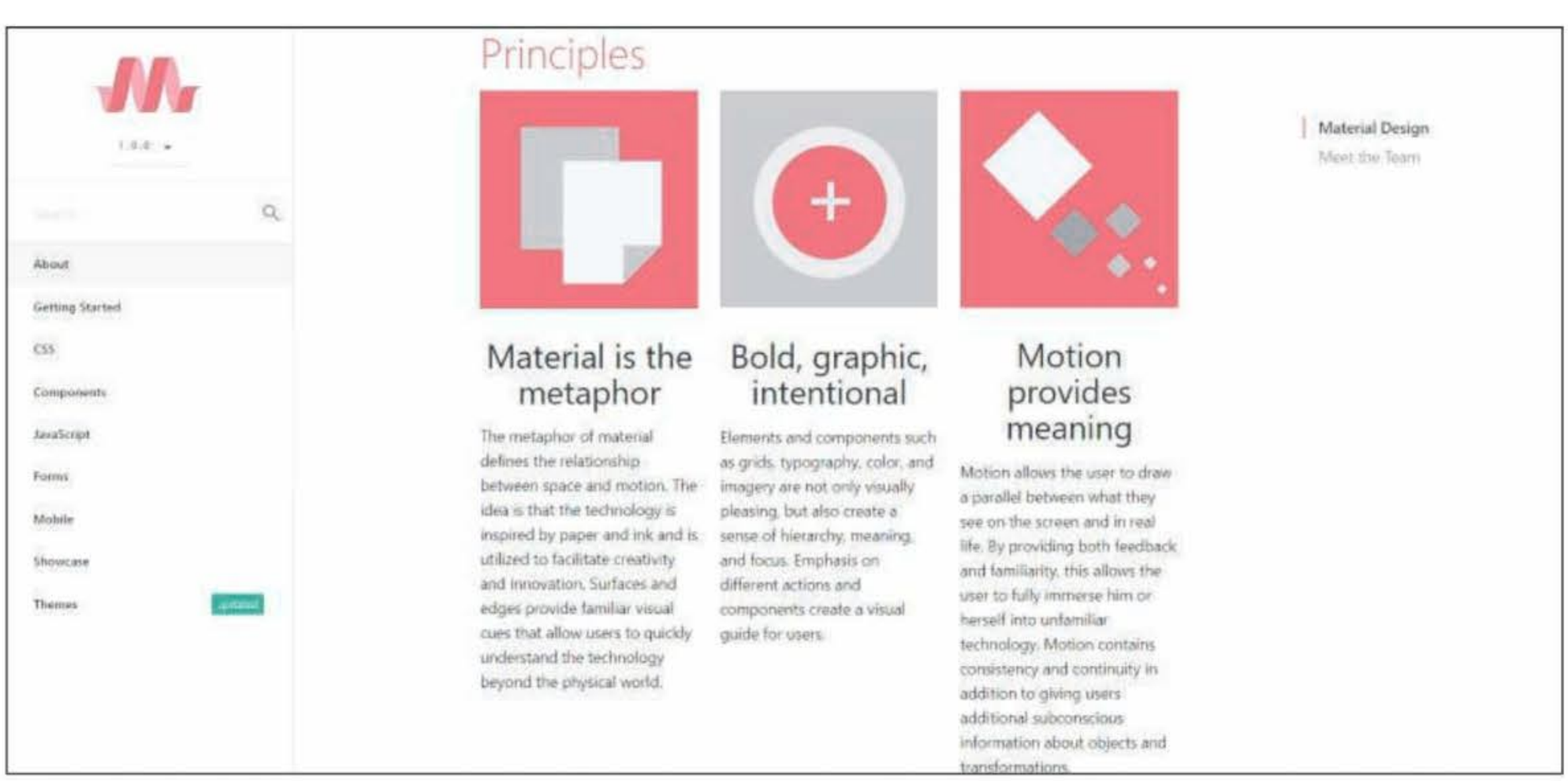
"Material Design para Bootstrap"



"Material Design Lite"

Materialize CSS

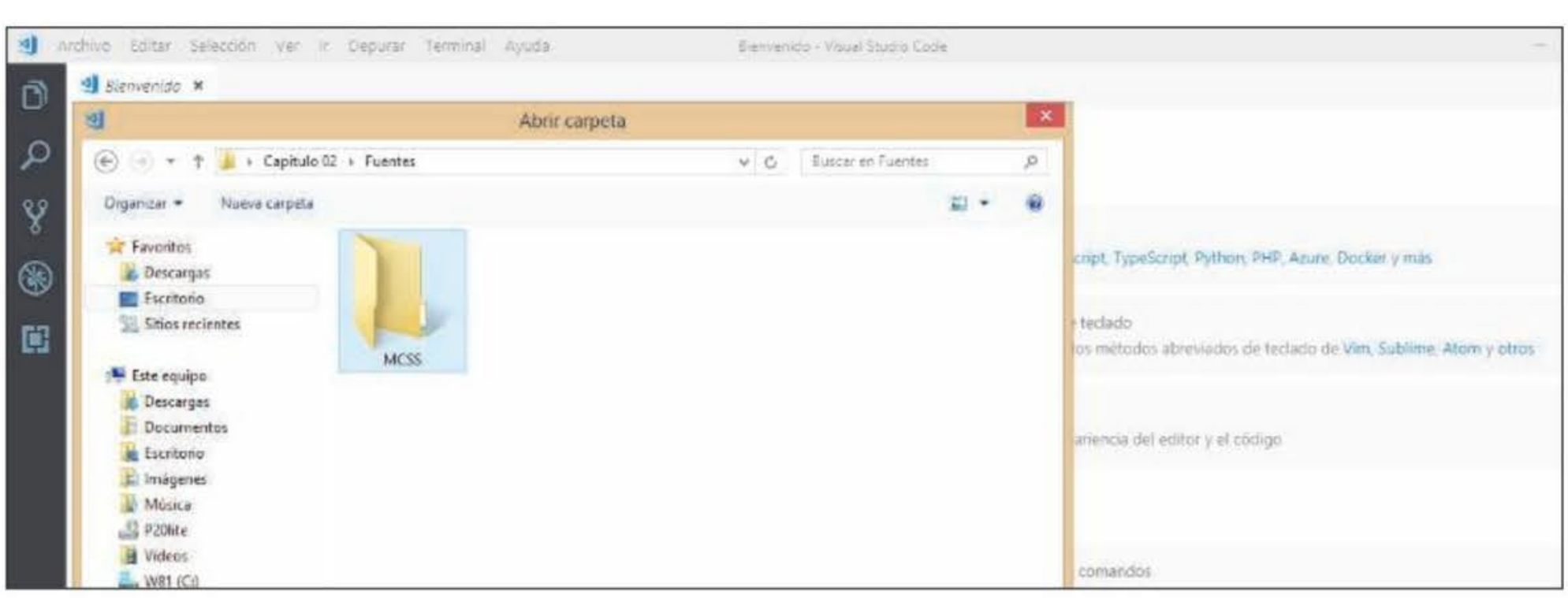
Una de las plataformas más elegidas, por sobre **MD for Bootstrap** y **Material Design Lite**, es Materialize CSS. Este framework nace de la mano de un equipo de estudiantes de la **Universidad de Carnegie Mellon, USA**, que dio vida al proyecto Materialize en tiempo récord como parte de su tesis de grado.



En base a su rápida aceptación y adaptación en el mercado por su estructura simple de implementar, este último framework es el que elegimos para desarrollar todos los ejemplos que encontraremos a lo largo de esta obra.

Estructura de un proyecto Materialize CSS

Comencemos a crear nuestros diseños aprovechando esta plataforma de desarrollo web. Para hacerlo, ingresamos en Visual Studio Code y generamos un nuevo proyecto, presionando **Ctrl + K Ctrl + O**. Buscamos o creamos una subcarpeta en nuestro equipo, donde se agregarán todos los archivos vinculados al desarrollo.



Estructura HTML

Acto seguido, pulsamos la combinación de teclas **Ctrl + N**, la cual nos permite crear un nuevo archivo. Una vez hecho esto, y antes de escribir una línea de código, presionamos **Ctrl + S** y lo guardamos con el nombre **index.html**.

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <meta name="viewport" content="width=device-width,
user-scalable=no"/>
        <title>Nuestro primer proyecto M-CSS</title>
        ...
    </head>
    <body>
    </body>
</html>
```

Integración de CSS, fuente tipográfica y JS

Dentro de este código HTML ubiquemos visualmente los tres puntos que están agregados en el apartado **<head>**, y reemplacémoslos por el siguiente bloque de código:

```
<link href="https://fonts.googleapis.com/icon?family
=Material+Icons" rel="stylesheet">
<link rel="stylesheet" href="https://cdnjs.cloudflare.com
/ajax/libs/materialize/1.0.0/css/materialize.min.css">
<script defer src="https://cdnjs.cloudflare.com/ajax/
libs/materialize/1.0.0/js/materialize.min.js"></script>
```

Con esto, ya incorporamos las referencias necesarias a nuestro documento HTML, para así poder utilizar dentro de él cada uno de los componentes que conforman el framework Materialize CSS. Por cada documento HTML que incluyamos en nuestro proyecto web, debemos replicar esta misma estructura de código, que nos garantizará el correcto funcionamiento de Materialize CSS. Veamos en la siguiente Guía Visual qué función cumple cada una de estas líneas agregadas en el apartado **<head>**.

GUÍA VISUAL 2

Esta referencia apunta a los iconos de acción oficiales de Google, utilizados en todas sus aplicaciones creadas con Material Design. Para conocer con mayor detalle cuáles son los iconos disponibles, es posible acceder al siguiente link: https://material.io/tools/icons/?style=baseline.

Esta referencia al archivo CSS contiene todas las clases que darán vida a nuestros componentes HTML, para que estos consigan el look and feel propio de Material Design.

Toda referencia al framework Materialize CSS y a las fuentes externas, como también a nuestros propios archivos CSS y JS, deben estar incluidas dentro del apartado ***<head>*** *y* ***</head>*** *de nuestros documentos HTML.*

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="UTF-8">
        <link href="https://fonts.googleapis.com/icon?family=Material+Icons"
        rel="stylesheet">
        <link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/
        materialize/1.0.0/css/materialize.min.css">
        <script defer src="https://cdnjs.cloudflare.com/ajax/libs/materialize/
        1.0.0/js/materialize.min.js"></script>
        <meta name="viewport" content="width=device-width, user-scalable=no"/>
        <title>Nuestro primer proyecto M-CSS</title>
    </head>
    <body>

    </body>
</html>
```

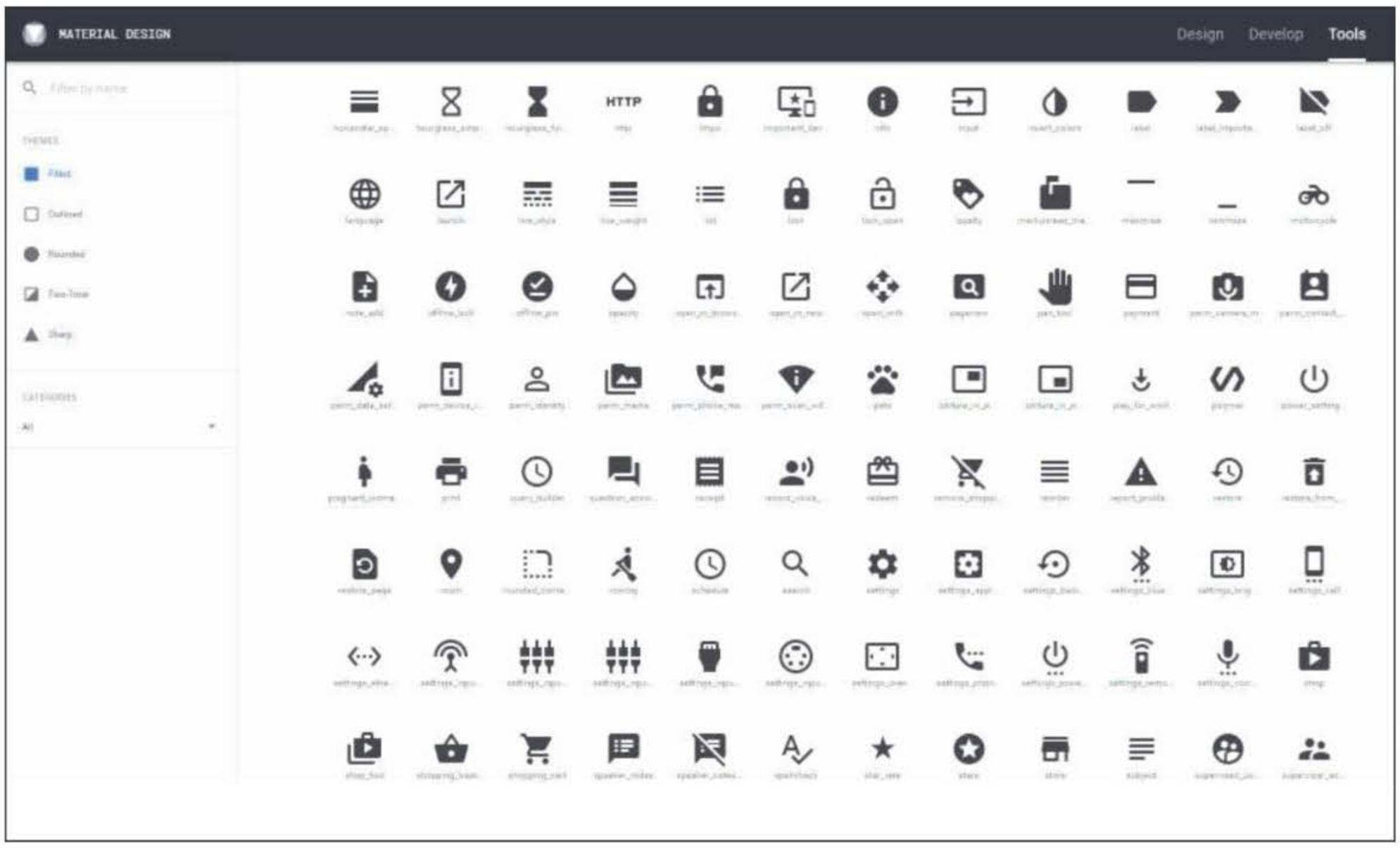
Algunos de los comportamientos y/o animaciones de los componentes de Materialize CSS dependen del accionar de JavaScript. Es por eso que se agrega esta librería de scripting.

Utilizamos el atributo ***defer*** *en su declaración para poder incluirlo al inicio del documento HTML, y que su comportamiento no se active hasta tanto este documento se haya cargado por completo.*

Todos nuestros proyectos se realizarán utilizando las referencias remotas (CDN) a Materialize CSS. Si no queremos depender constantemente de la conexión a Internet, podemos descargar los archivos oficiales y alojarlos en forma local, desde el siguiente link:
https://github.com/Dogfalo/materialize/releases/download/1.0.0/materialize-v1.0.0.zip.

Iconografía

Como bien explicamos en la infografía, los iconos que utilizaremos a lo largo de nuestros desarrollos provienen de las fuentes iconográficas oficiales creadas por Google para la plataforma Material Design.

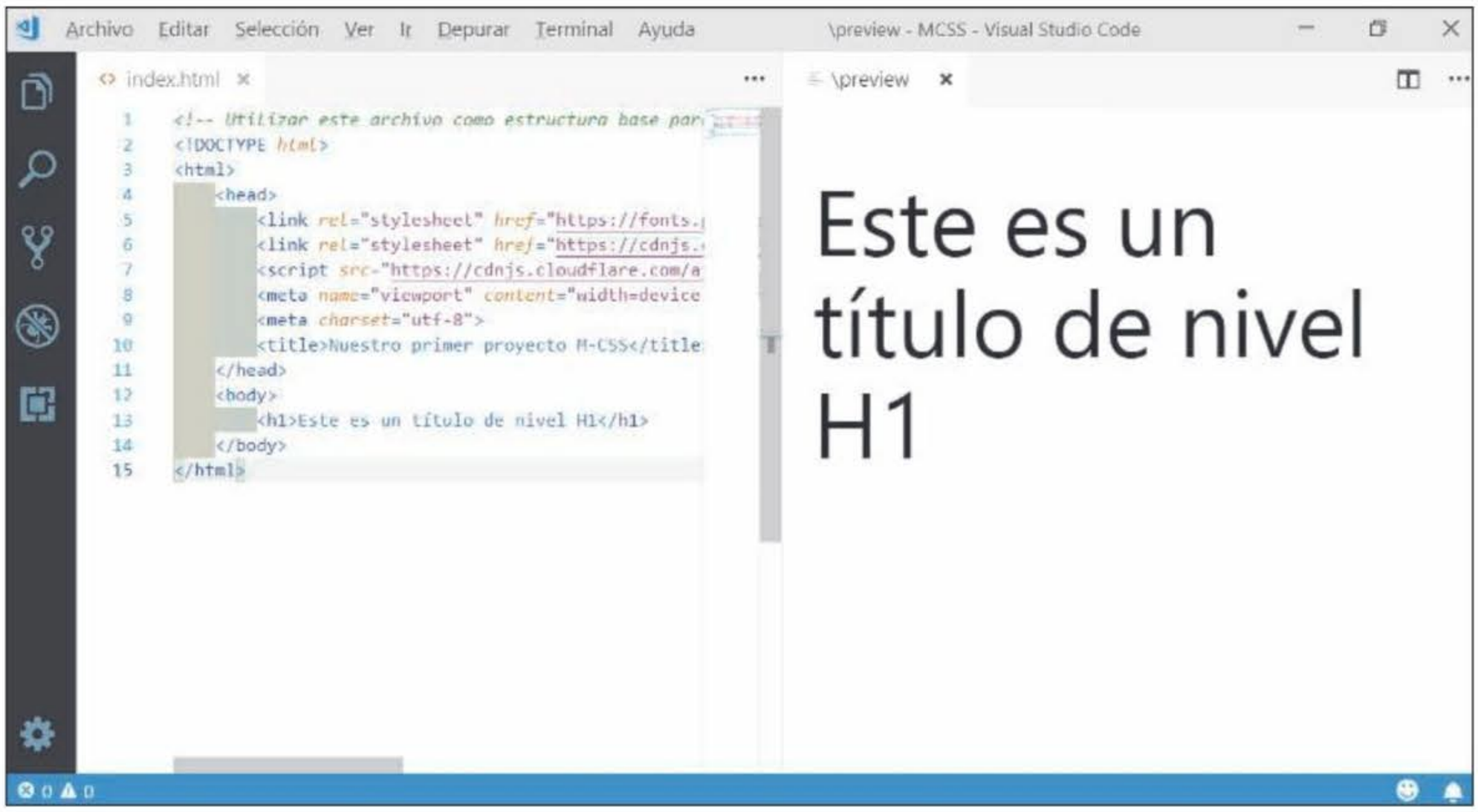


Verificar la funcionalidad

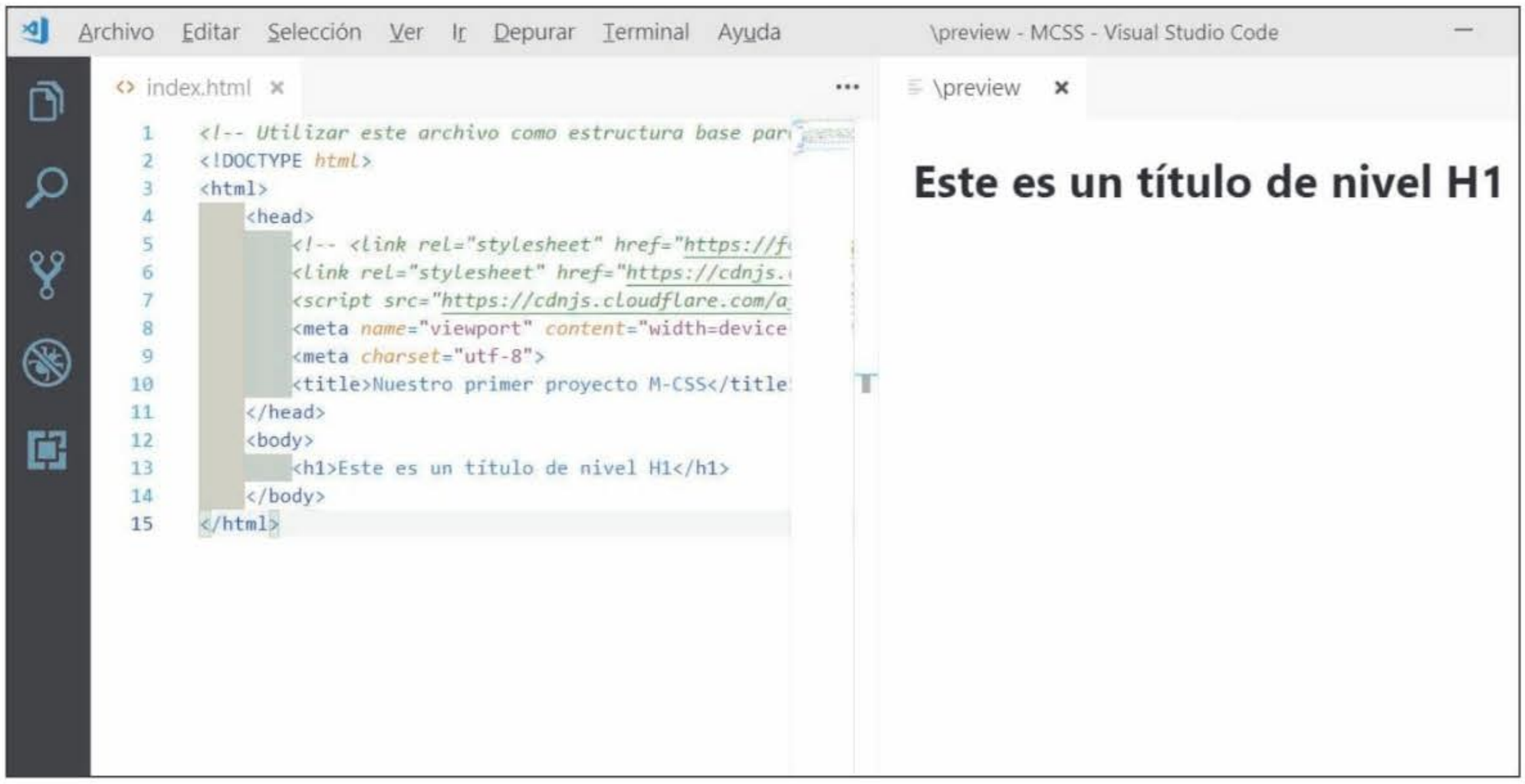
Veamos a continuación cómo verificar la funcionalidad de Materialize CSS en nuestro desarrollo. Nos ubicamos dentro del apartado **<body> </body>** del documento HTML y agregamos la siguiente línea de código:

```
<h1>Este es un título de nivel H1</h1>
```

Cargamos el documento HTML en nuestro navegador web o dentro de la vista previa de Visual Studio Code, y analizamos su estética, que tiene que ser similar a la siguiente figura:



Luego renombramos todas las referencias que agregamos de Materialize CSS en el apartado **<head>**, seleccionándolas y presionando la combinación de teclas **Ctrl + }**. Guardamos el documento HTML, y volvemos a cargar o refrescar la página dentro del navegador web o la vista previa. El resultado ahora debe ser semejante al de la siguiente figura:



Con esta prueba, podemos certificar que, con el cambio de estilo en los títulos, Materialize CSS está funcionando correctamente.

CSS adicional

Si necesitamos trabajar con hojas de estilo adicionales, debemos agregar la o las líneas de referencia, justo por debajo de las líneas añadidas anteriormente. De esta forma, nos aseguramos de que nuestra hoja de estilos CSS personalizada aplique el diseño que deseamos a cualquier componente HTML, por sobre lo que dictamina la hoja de estilos propia de Materialize CSS.

JavaScript adicional

De la misma manera en que agregamos una hoja de estilos personalizada a nuestro documento HTML, debemos hacerlo con cualquier archivo del tipo JS que deseemos incluir en un proyecto. Podemos añadirlos en una línea anterior al tag de cierre **</body>** o agregarlos después del JS correspondiente a Materialize CSS, junto con el atributo **defer**:

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

Personalizar la fuente tipográfica

Por defecto, Materialize CSS incluye la fuente tipográfica **Roboto** embebida en todas las clases CSS creadas en el archivo de hoja de estilos de este framework. Si por algún motivo necesitamos personalizar el tipo de fuente, dejando de lado la predeterminada, solo debemos agregar una hoja de estilos como la referenciada anteriormente en este capítulo, e incluir en ella el siguiente bloque de código:

```
body {
    font-family: 'Courier New', 'Courier', 'monospace';
}
```

¡Atención!

En muchos ejemplos relacionados con la declaración CSS de fuentes, estas se representan de la siguiente manera:

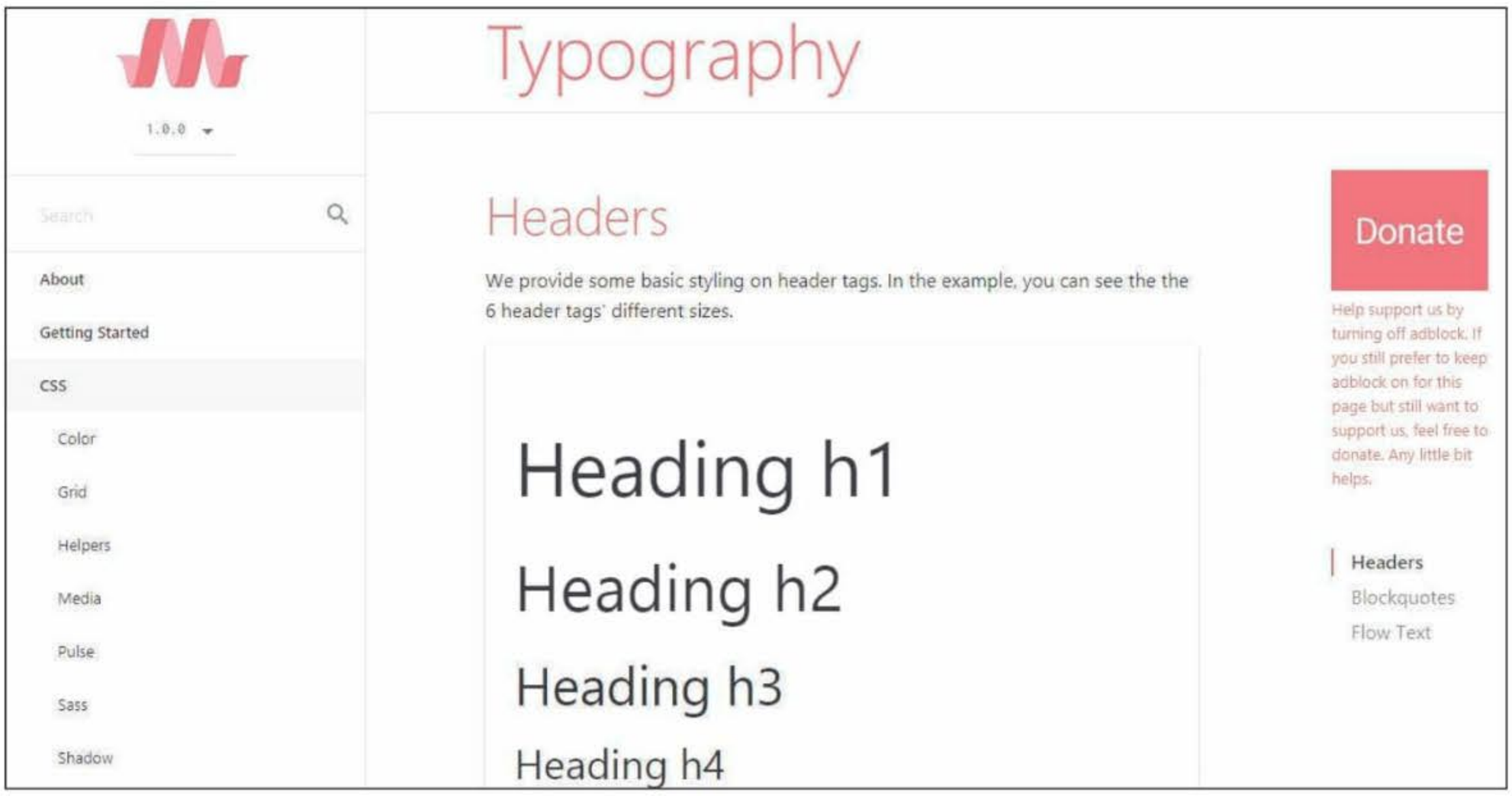
```
font-family: Courier New, Courier, monospace;
```

Como podemos visualizar en el bloque de código anterior, una de las tres posibles fuentes contiene un nombre compuesto: **Courier New**. En algunos casos, esto puede provocar que el navegador web no la interprete, al igual que las otras

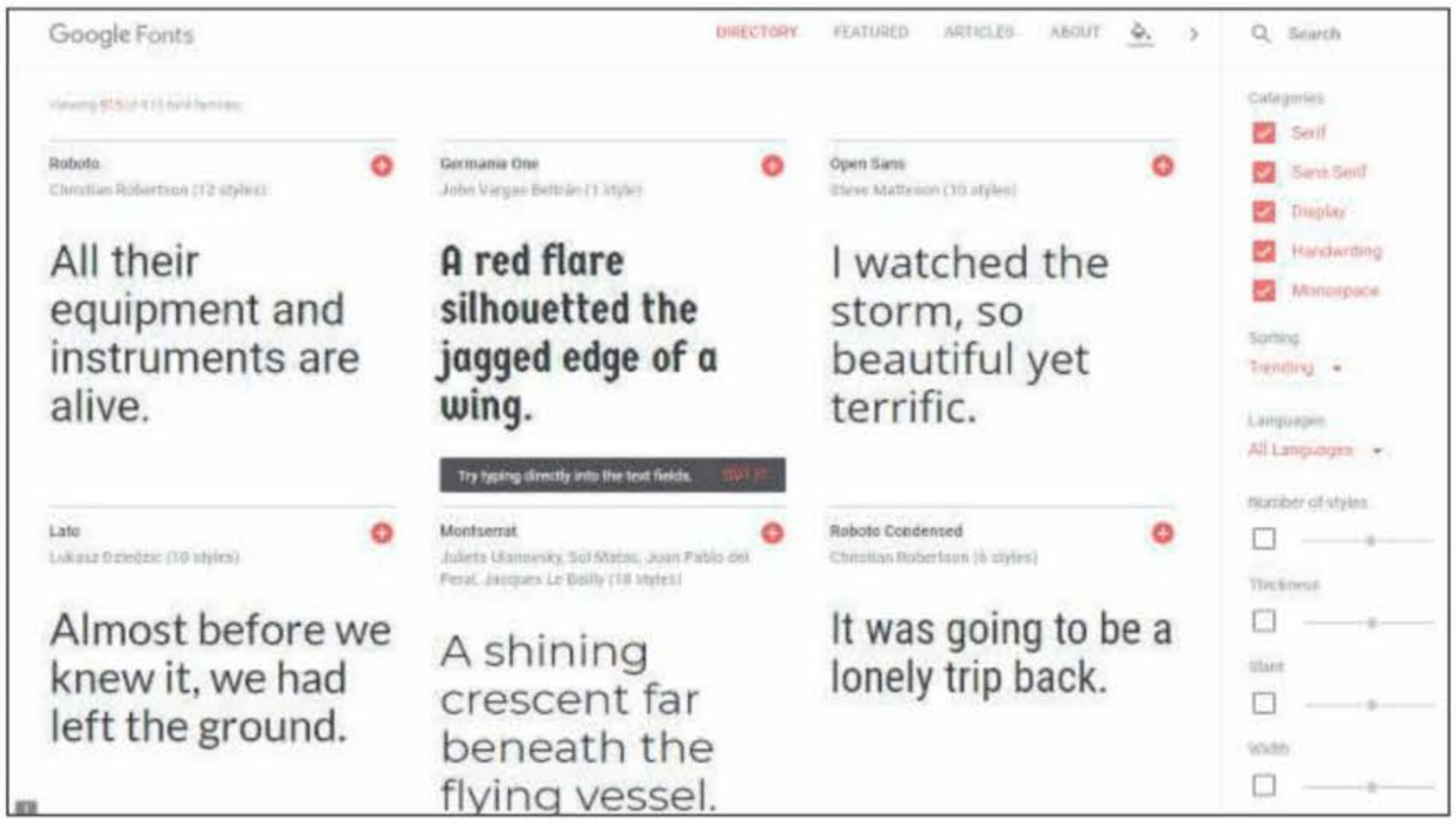
que le siguen. Por lo tanto, siempre recomendamos agregar la comilla simple al inicio y final de cada tipografía en cada una de las familias de fuentes declaradas en los bloques de código CSS, tal como lo representamos en el primer bloque de código de ejemplo. Esto nos garantizará que el motor del navegador web interprete y tome la familia de tipografía de fuentes de manera correcta.

Más información sobre tipografías

Para conocer con más detalle la información y la manera en que Materialize CSS maneja la fuente tipográfica por defecto que se usa dentro de este framework, podemos acceder al apartado oficial del sitio web: https://materializecss.com/typography.html.



Google Fonts también es un portal de referencia para buscar y personalizar el uso de fuentes tipográficas específicas destinadas a algún proyecto en particular que las requiera.



Clases CSS

Materialize CSS moldea la estética de cada componente, aplicándole una clase en la referencia HTML. Esta clase está declarada en la referencia CSS propia de Materialize que vimos con anterioridad, y se incluye, simplemente, añadiendo el atributo **class** dentro del componente en cuestión que queremos personalizar. Veamos un ejemplo:

```
<h1>Este es un título</h1>
```

El código anterior es una forma simple de declarar un componente HTML. Ahora, si le agregamos la clase CSS **red**, esta hará que el título sea de color rojo:

```
<h1 class="red">Este es un título</h1>
```

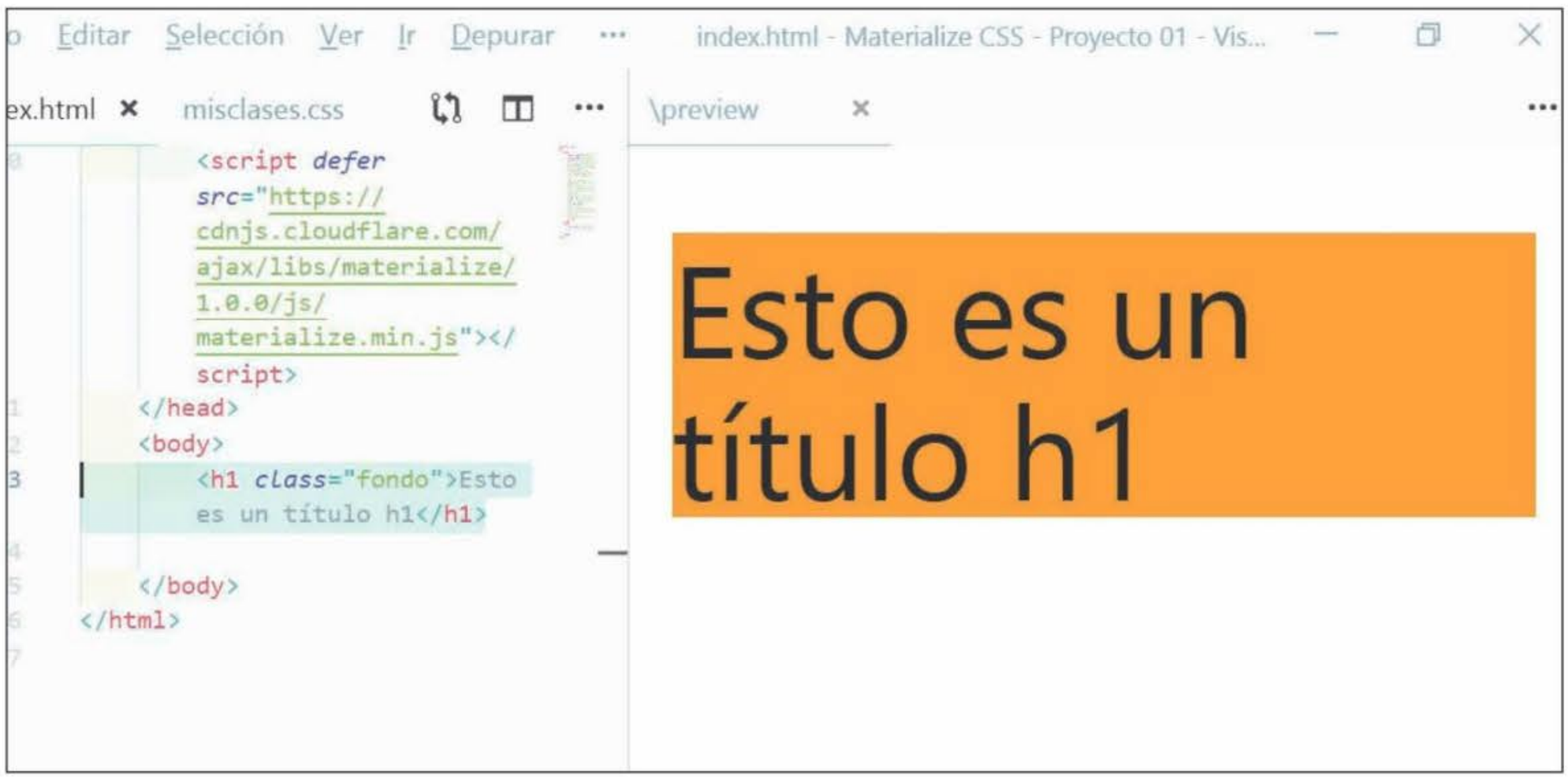
Si queremos crear nuestras propias clases por afuera de Materialize CSS, debemos generar una hoja de estilos y declararla en el apartado **<head>** del documento HTML, por debajo de la hoja de estilos propia de Materialize. Luego, en ella, incorporamos nuestro propio código CSS, como el del siguiente ejemplo:

```
.fondo {
    background-color: orange;
}
.cursiva {
    font-style: italic;
    color: White;
}
```

Cada clase CSS personalizada que declaremos debe llevar un punto (.) antepuesto a su nombre, y luego, dentro de las llaves, las propiedades detalladas que deseamos modificar. Para invocarla cuando creamos un tag HTML, tenemos que especificar el atributo **class** en él, de la siguiente manera:

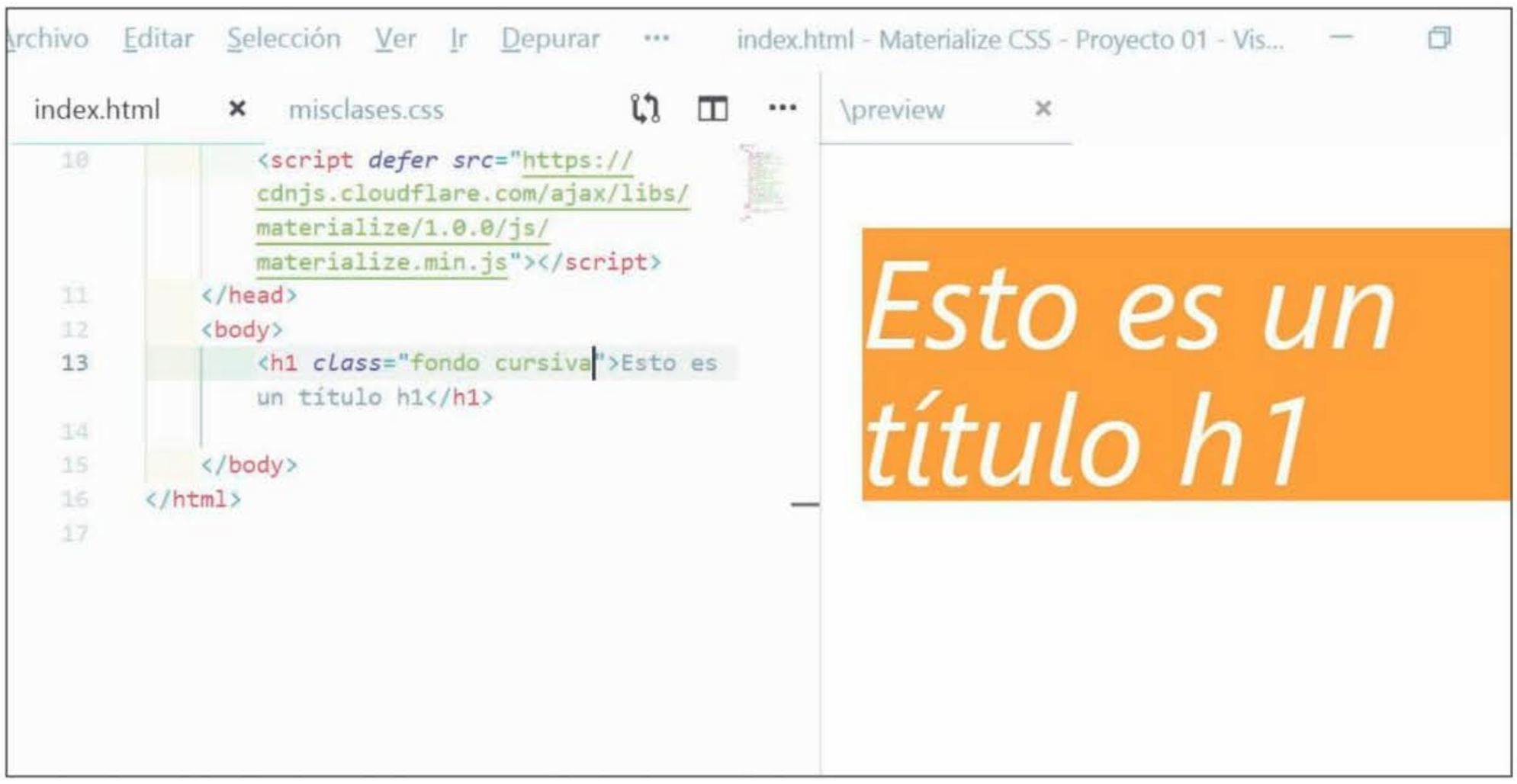
```
<h1 class="fondo">Esto es un título h1</h1>
```

Así la clase será aplicada en dicho archivo.



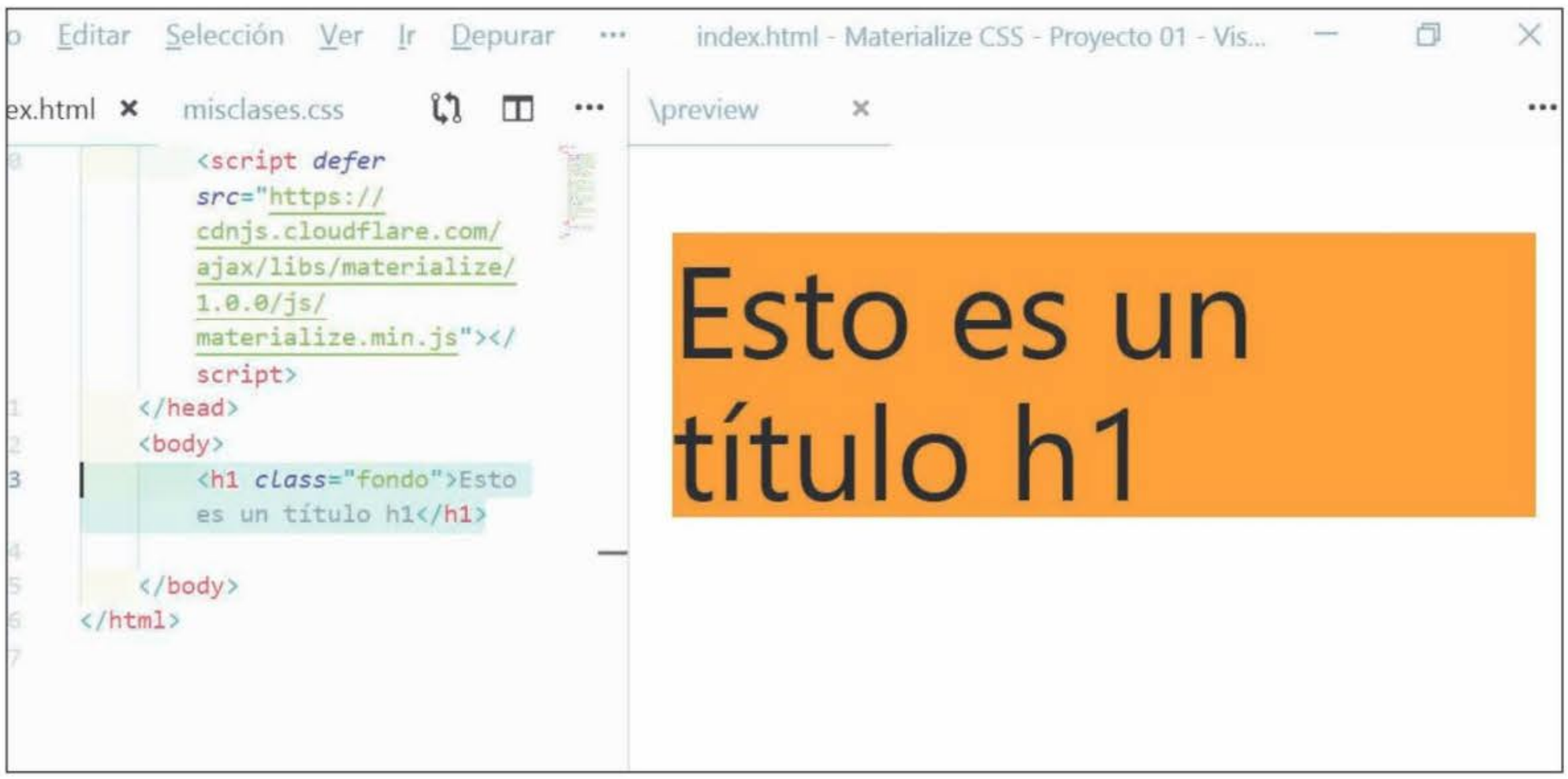
Si queremos aplicar en un estilo dos o más clases con diferentes formatos, las agregamos al atributo **class**, separándolas por un espacio.

```
<h1 class="fondo cursiva">Esto es un título h1</h1>
```



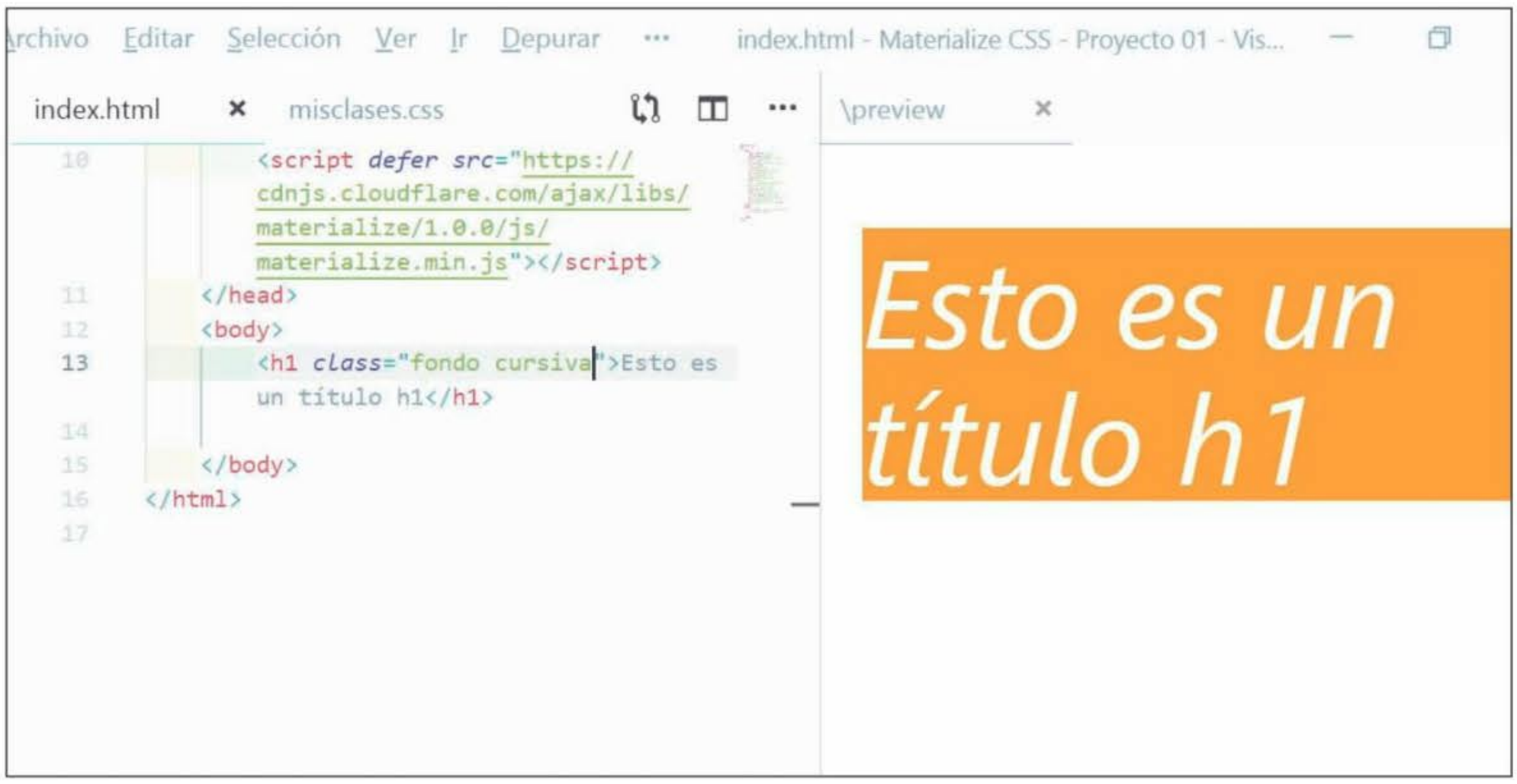
En el apartado Recursos de esta obra se presenta este ejemplo bajo el nombre: **Materialize CSS - Proyecto 01.zip**.

Así la clase será aplicada en dicho archivo.



Si queremos aplicar en un estilo dos o más clases con diferentes formatos, las agregamos al atributo **class**, separándolas por un espacio.

```
<h1 class="fondo cursiva">Esto es un título h1</h1>
```



En el apartado Recursos de esta obra se presenta este ejemplo bajo el nombre: **Materialize CSS - Proyecto 01.zip**.

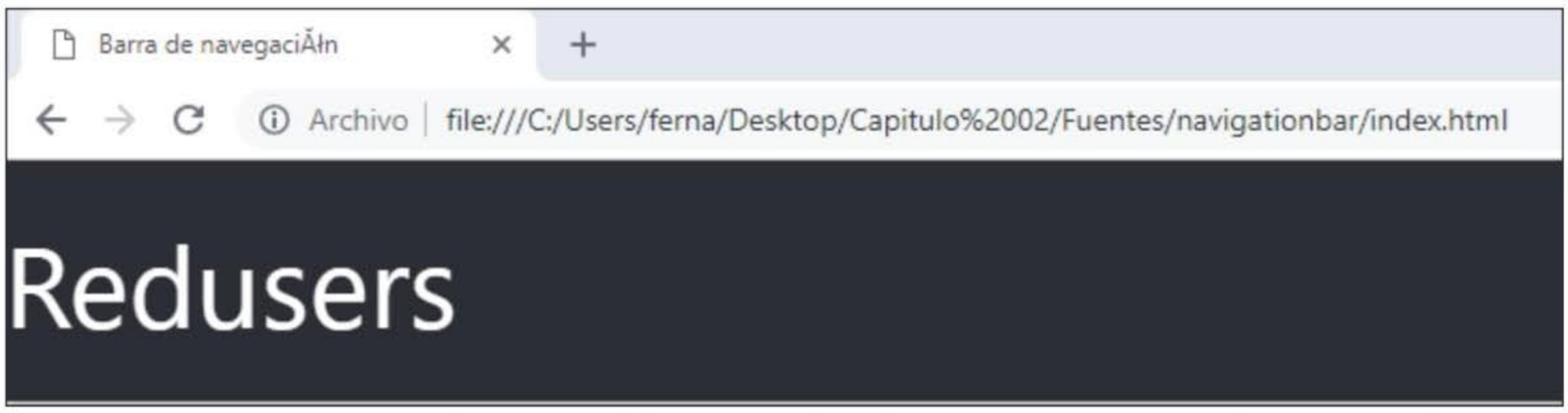
02 Barras de navegación

El sistema de navegación en un sitio o aplicación web debe ser intuitivo y de fácil interacción. Veamos a continuación las opciones que nos propone Materialize CSS a través de su componente navbar.

El primer paso en este capítulo para aprender a utilizar las barras de navegación consiste en armar un nuevo proyecto. Lo creamos en Visual Studio Code bajo el nombre **navigationbar**, o cualquier otro que nos guste.
A continuación, agregamos un documento HTML con la estructura base explicada en el capítulo anterior y, dentro del apartado **<body>**, incluimos el siguiente bloque de código:

```
<nav>
    <div class="nav-wrapper black">
        <a href="#" class="brand-logo">Redusers</a>
    </div>
</nav>
```

Guardamos el documento HTML y lo ejecutamos en el navegador.



Navbar

En el paso anterior, creamos una barra de navegación que muestra un logo, título o texto, sobre el extremo derecho. El código escrito nos indica que, simplemente, creamos el elemento **<nav>**, propio de HTML5. Dentro de él, generamos un ** ** con las clases **nav-wrapper** y **black** asociadas: **nav-wrapper** agrupa el div sobre el extremo superior de la página ocupando el 100% de ancho, mientras que **black** se encarga del color de fondo. Dentro de este **div**, encontramos la etiqueta

<a>, cuya clase **brand-logo** agrupa un texto de dimensión mediana sobre la derecha de la página.

Agreguemos el siguiente bloque a continuación de la etiqueta **<a>**:

```
<ul id="nav" class="right">
    <li><a href="pagina1.html">Bienvenidos</a></li>
    <li><a href="pagina2.html">Empresa </a></li>
    <li><a href="pagina3.html">Productos </a></li>
</ul>
```

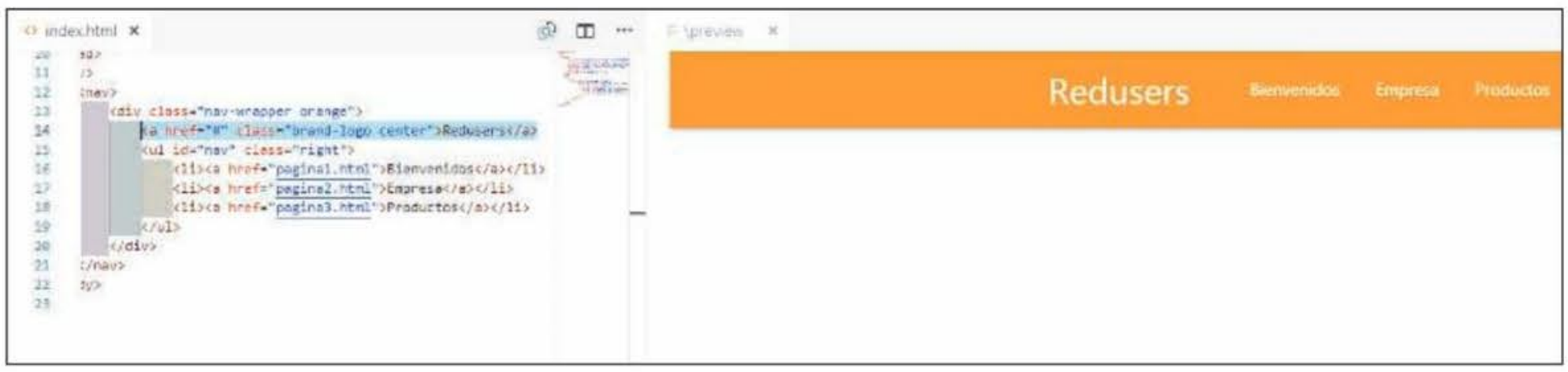
Una barra de navegación funcional

Navbar es el elemento HTML que mejor se adapta en el despliegue de menús de cualquier sitio web. Como podemos notar, su implementación es muy sencilla y, a su vez, con muchas opciones que nos permiten flexibilizar su estética y comportamiento. Continuemos ahora aplicando otras opciones. Vimos que el color de fondo establecido es negro. Para cambiarlo, solo debemos reemplazarlo por cualquier otro que nos guste: **yellow**, **red**, **blue**, **brown**, etcétera.

```
<div class="nav-wrapper orange">
```

Por ejemplo, si queremos alinear el **logo** en el centro de la página, o sobre la derecha en vez de la izquierda, reemplazamos el atributo **left** (de alineación) por **center** o **right**:

```
<a href="#" class="brand-logo center">Redusers</a>
```



Para invertir las posiciones del logo con los botones, además de cambiar la clase del primero de ellos, debemos hacer lo propio con la clase **right**, definida al inicio del bloque de código de los botones, ingresando la clase opuesta a la que elijamos para el logo:

```
<ul id="nav" class="left">
    ...
```

Insertar un logo gráfico

Reemplacemos a continuación el título **Redusers**, escrito en **navbar**, por una imagen gráfica que represente mejor nuestra barra de herramientas. Para hacerlo, simplemente eliminamos la palabra del ejemplo anterior y, en su lugar, colocamos la siguiente línea:

```
<img src="images/logo_redusers.png" />
```

Si vemos que el tamaño del logo no concuerda con las dimensiones promedio que tiene una barra de navegación, podemos reducir su ancho desde HTML, agregando el atributo **width** a la línea de código anterior:

```
... width="50%" />
```

Uso de clases según el display

Todos los componentes de Materialize CSS pueden mostrarse u ocultarse según nuestras necesidades y de acuerdo al tipo de pantallas donde se visualiza nuestra solución web. Para esto, existe una serie de **clases** que nos permiten administrarlos de manera eficiente. Veamos cuáles son a través de la **TABLA 1**.

De acuerdo con las pruebas responsivas realizadas con el **navegador web Chrome**, aquellas pantallas de hasta **992 píxeles** de ancho se interpretan como pantallas de media-baja resolución, mientras que los displays superiores a este ancho son tomados como pantallas de alta resolución.

TABLA 1

CLASE	FUNCIÓN
hide	Oculta el elemento en cualquier tipo de pantalla.
hide-on-small-only	Oculta el elemento solo en pantallas chicas.
hide-on-med-only	Oculta el elemento solo en pantallas medianas.
hide-on-med-and-down	Oculta el elemento en pantallas medianas y pequeñas.
hide-on-med-and-up	Oculta el elemento en pantallas medianas y grandes.
hide-on-large-only	Oculta el elemento solo en pantallas grandes.
show-on-small	Muestra el elemento solo en pantallas chicas.
show-on-medium	Muestra el elemento solo en pantallas medianas.
show-on-large	Muestra el elemento solo en pantallas grandes.
show-on-medium-and-up	Muestra el elemento en pantallas medianas y grandes.
show-on-medium-and-down	Muestra el elemento en pantallas medianas y chicas.

Entendemos por pantallas chicas a smartphones y tablets de 7 pulgadas; por pantallas medianas a las tablets de baja/media gama (8 pulgadas o más); y por pantallas grandes a las tablets de 10 pulgadas o más, displays portátiles y monitores en general.

Ejercicio práctico: Barra de menú superior responsiva

Ya tenemos dominada una parte del aspecto de navbar en el terreno web, pero como desarrolladores, debemos tener siempre presente la filosofía **Mobile First**. Para esto, realizaremos ahora un ejercicio práctico que nos permita seguir entendiendo el uso de una barra de navegación, adaptándola para que sea compatible con las plataformas móviles.

A continuación, adaptamos la barra de herramientas integrando en ella una clase específica para el ámbito mobile. Para lograrlo, ubicamos el elemento **<ul>** y, en su atributo **class**, agregamos la clase **hide-on-med-and-down**. Como seguramente cuenta con otra clase agregada, recordemos dejar un espacio de separación entre una y otra.

```
<ul id="nav" class="hide-on-med-and-down right">
```

El elemento **<ul>** contiene el menú de navegación de nuestro sitio.

Con esta clase agregada, cuando este documento HTML se cargue en pantallas de tablets o smartphones, se ocultará, actuando correctamente bajo la modalidad responsiva. Pero como dependemos de que se muestre un menú acorde a las pantallas de mediana y baja resolución, necesitamos crearlo y hacer que se vea solo en estos casos. Entonces, agregamos otro elemento **<ul>** a continuación del que ya estaba creado, con el siguiente código:

```
<ul id="mobile-menu" class="hide-on-large-only right">
    <li><a href="#">Menú</a></li>
</ul>
```

Como podemos ver, en este otro elemento utilizamos una clase diferente, que solo lo visualizará cuando el documento HTML se cargue en pantallas de mediana o baja resolución. El menú que se mostrará, por el momento, no tiene ninguna interacción.

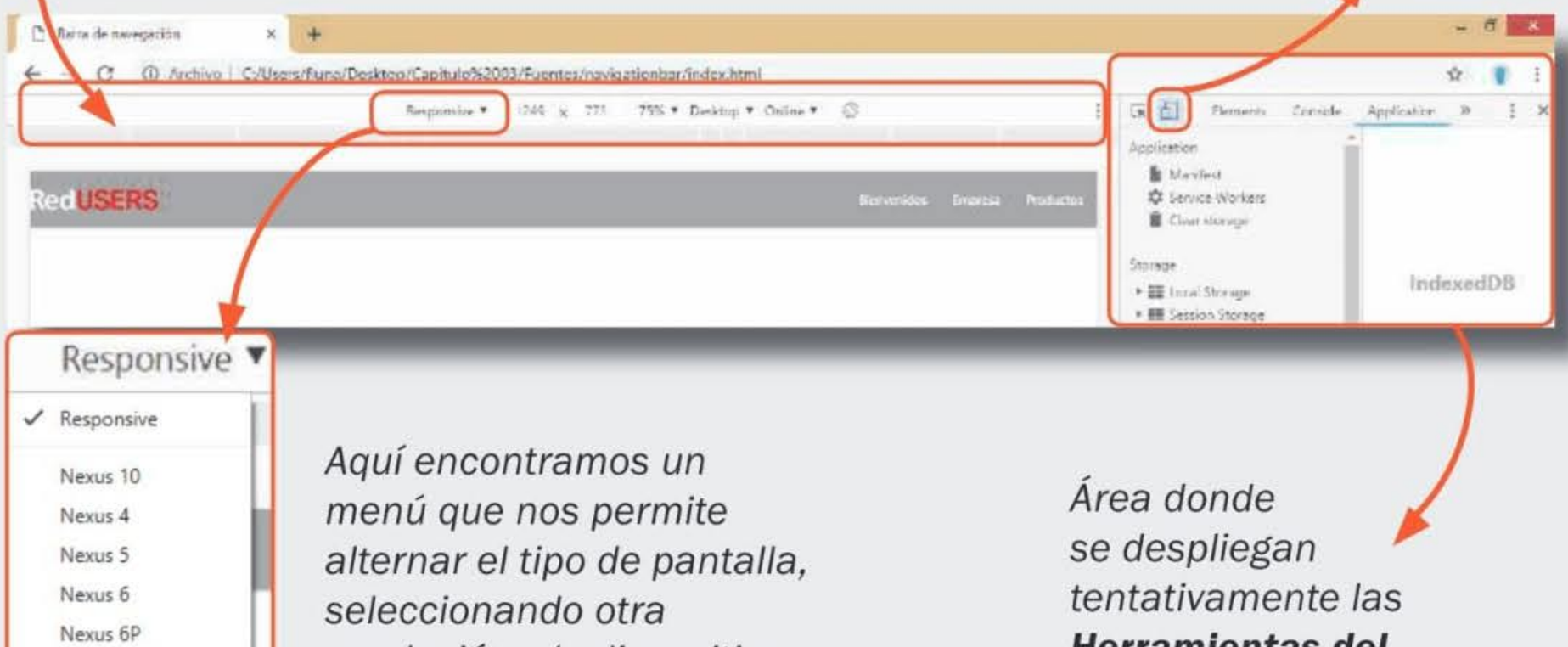
Testear el comportamiento responsivo

Guardamos las modificaciones del documento HTML y lo cargamos en Google Chrome. Para corroborar la correcta respuesta del framework, presionemos la tecla **F12**, que acciona las **Herramientas del desarrollador**. Ubicamos el icono **Toggle Device Toolbar** y lo pulsamos. Esta acción activa el **Modo Dispositivo**, que nos permitirá testear lo necesario. Veamos en la **Guía Visual** cómo debemos proceder:

GUÍA VISUAL 1

Visualizaremos una barra de herramientas sobre nuestro documento HTML.

*Pulsamos el botón **Toggle Device Toolbar**, también accionable mediante la combinación de teclas **Ctrl + Shift + M**.*



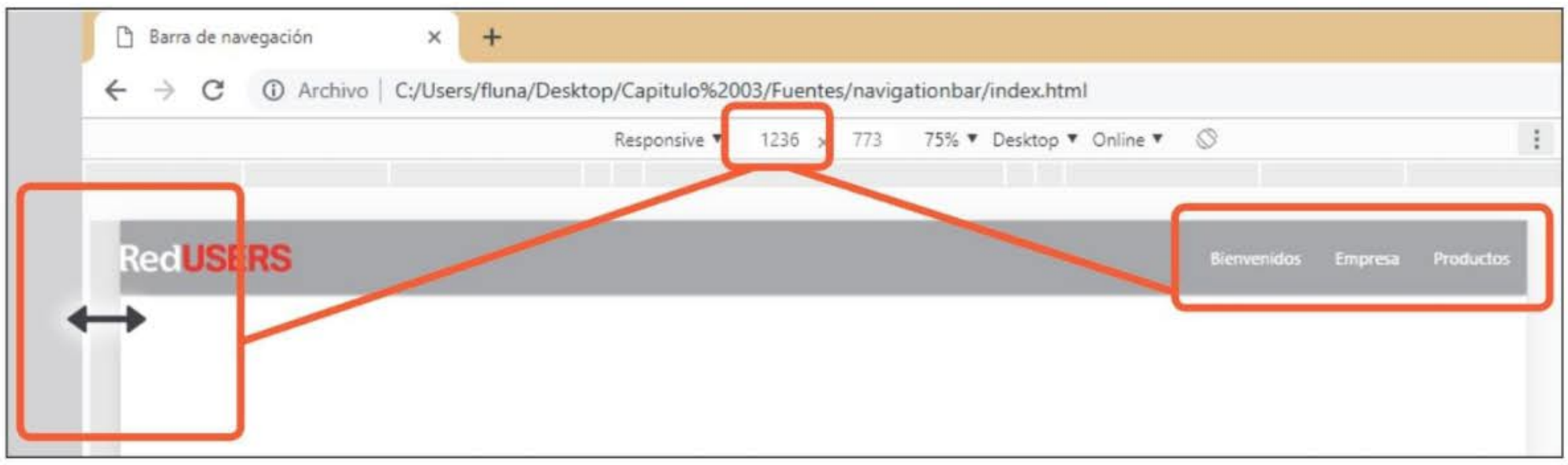
Aquí encontramos un menú que nos permite alternar el tipo de pantalla, seleccionando otra resolución y/o dispositivo.

*Nos aseguramos de que, por el momento, esté seleccionada la opción **Responsive**.*

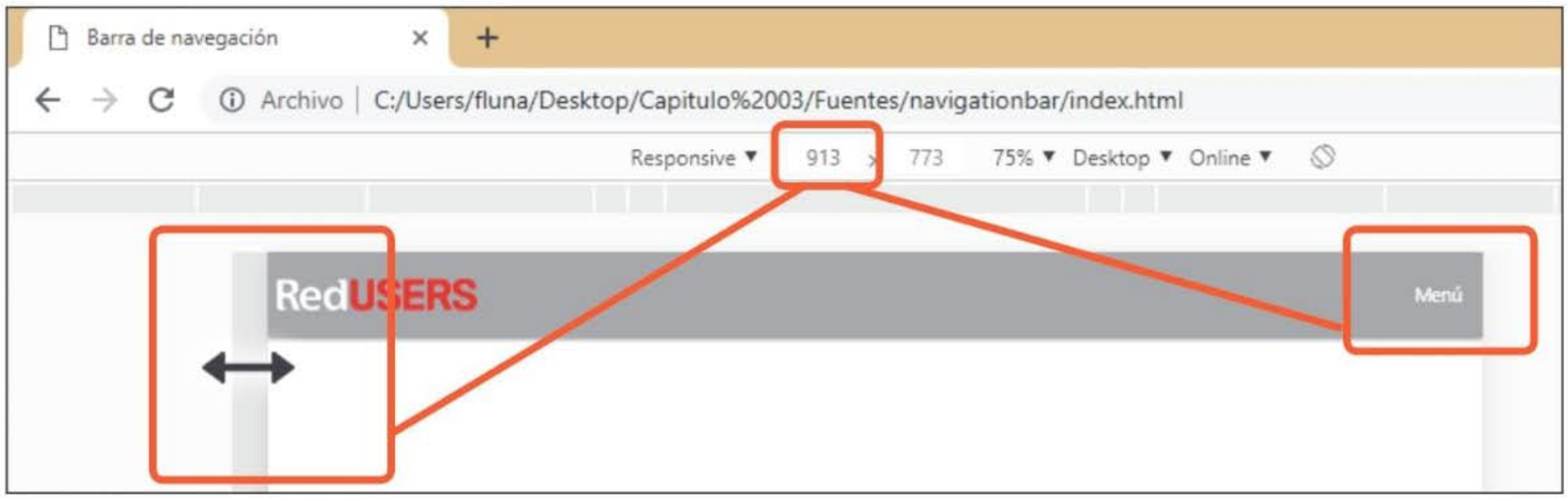
*Área donde se despliegan tentativamente las **Herramientas del desarrollador**, o **Chrome Developer Tools**, cuando pulsamos la tecla **F12**.*

Redimensionar la página

Acercando el puntero del mouse hacia uno de los laterales del documento HTML, este cambiará su icono por el de redimensionamiento. Al hacer clic y mantenerlo pulsado, podremos cambiar el tamaño de la página HTML. El redimensionamiento se verá reflejado en la barra superior, donde cambiará el valor numérico de **width**.



A su vez, al reducir el ancho de la pantalla, veremos que el menú principal creado en la barra de navegación desaparece, y se activa el menú **Mobile** generado.



Optimización del menú para dispositivos móviles

Luego de haber resuelto el comportamiento responsivo de nuestro menú de navegación, profundicemos en el diseño de un menú alternativo orientado a dispositivos móviles. Para esto, vamos a eliminar, en primera instancia, el elemento ** Menú** que creamos para probar el modo responsivo. Ahora, en su lugar, agregamos el siguiente código:

```
<li><a href="#"><i class="material-icons">person</i></a></li>
<li><a href="#"><i class="material-icons">business</i></a></li>
<li><a href="#"><i class="material-icons">business_center</i></
a></li>
```

Este menú dedicado a dispositivos móviles se desplegará visualizando iconos en vez de etiquetas. Veamos a continuación cómo funciona.

Material icons

Como comentamos en el capítulo anterior, Materialize CSS incluye más de 900 iconos, los cuales corresponden a la opción oficial integrada por Google en Material Design. El acceso a ellos se configura al agregar las referencias oficiales de este framework en el encabezado de los documentos HTML:

```
<link rel="stylesheet" href="https://fonts.googleapis.com/
icon?family=Material+Icons">
```

Su implementación es muy fácil. Veamos la siguiente Guía Visual para comprender la forma de hacer uso de un icono:

GUÍA VISUAL 2

Para visualizar un icono, iniciamos su llamada usando los elementos **** *e* ****

La clase ***material-icons*** *es la que llama al estilo propio con el cual visualizamos el elemento gráfico "icono".*

Indicamos entre los elementos **** *e* ****, *el nombre del icono que deseamos visualizar.*

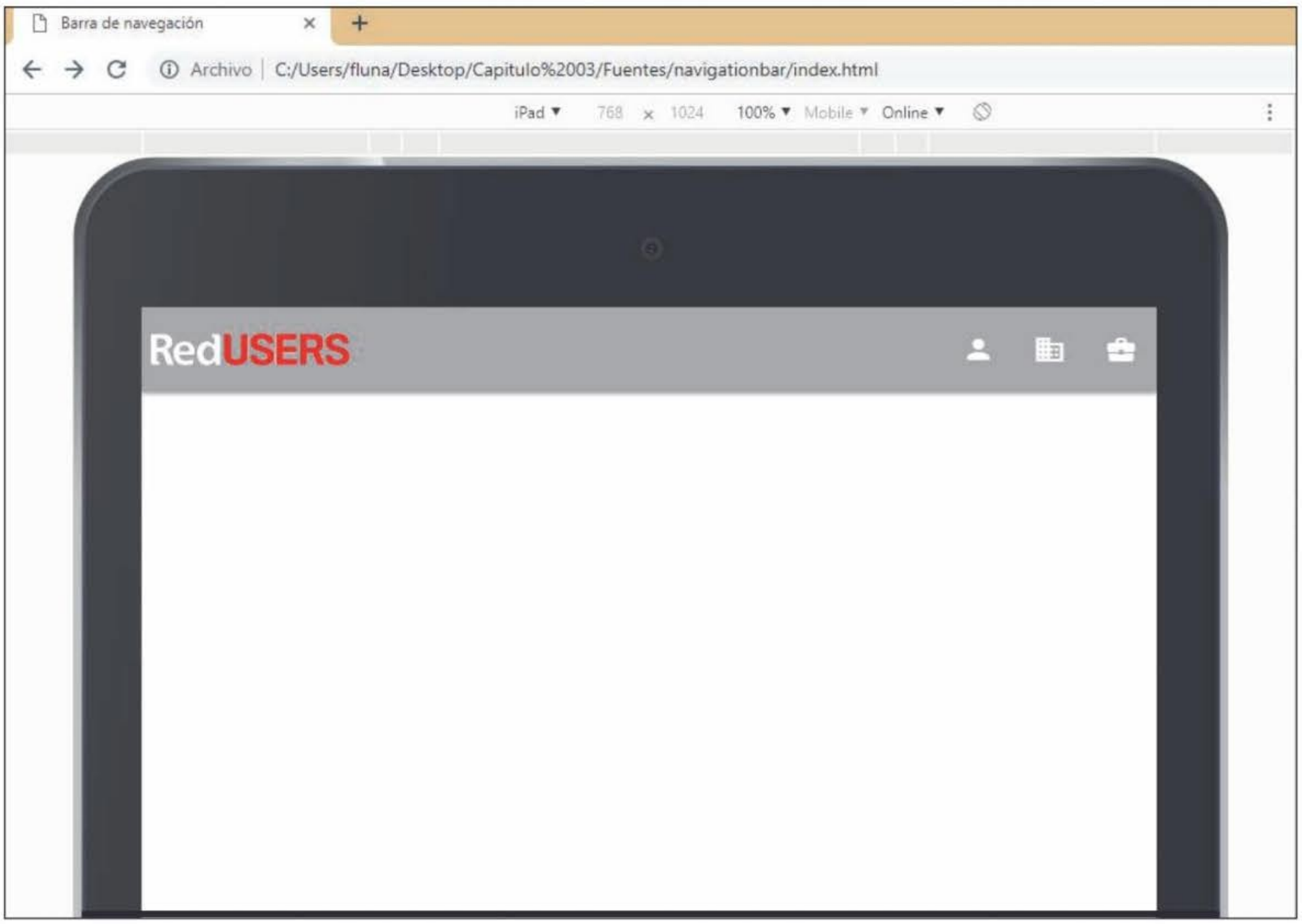
```
<i class="material-icons">person</i>


<a href="#"><i class="material-icons">person</i></a>
```

Si deseamos que el icono cuente con un hipervínculo, debemos encerrarlo dentro del elemento HTML **<a>** *y* ****, *y detallar el hipervínculo en el atributo* **href** *de este elemento.*

El uso de iconos se puede aplicar de manera independiente en casi cualquiera de los componentes de Materialize CSS que utilicemos. Solo debemos tener en cuenta que, si lo usamos dentro de un componente **button** (el cual veremos más adelante), debemos desplegar el hipervínculo que este botón tenga asociado, a nivel botón y no a nivel icono. Esto impedirá que el componente **button** quede limitado en cuanto a su funcionamiento.

Testear su comportamiento

Solo nos resta desplegar en Chrome los cambios aplicados al menú para displays grandes o displays de dispositivos móviles. Si todo va bien, la respuesta que tendremos, simulando el mismo test hecho anteriormente, nos deberá mostrar en pantalla móvil los iconos, en vez de los botones de menú basados en texto.



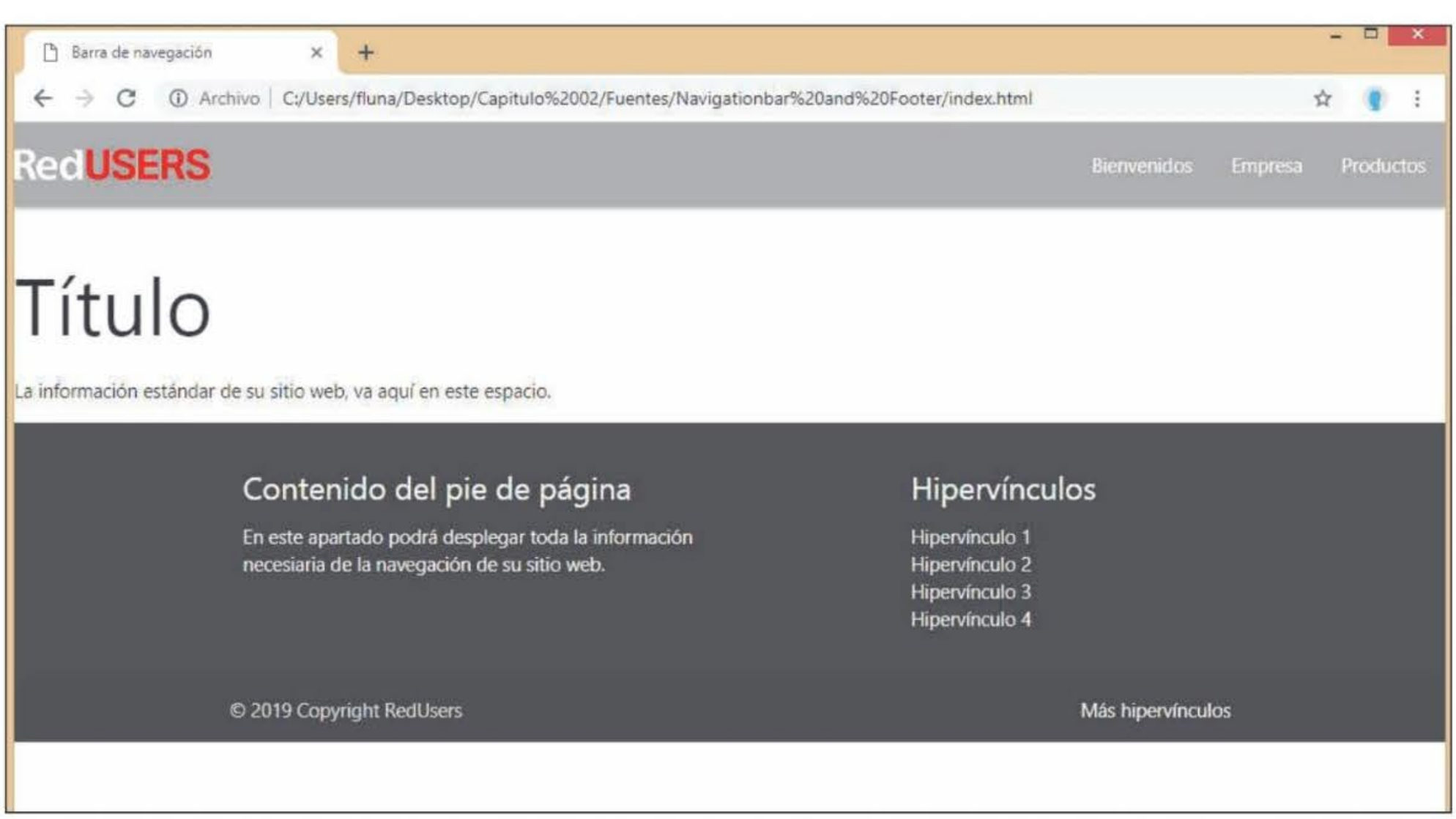
Podemos elegir en el menú superior algún dispositivo móvil para lograr una aproximación más certera de cómo se visualizará nuestra barra de navegación responsiva. Cabe recordar que en el repositorio de archivos que acompaña a esta guía, está este ejercicio resuelto para descargar y comparar, bajo el nombre **Ejercicio-navigationbar-Responsiva-resuelto.zip**.

Footer: construir un pie de página funcional

Todo sitio web dispone de un pie de página con información importante relacionada a él. Veamos a continuación la propuesta de Materialize CSS, que nos ayudará a construir un footer acorde a nuestra necesidad real.

Un footer o pie de página es el elemento HTML que nos permite ordenar la navegación de todo el sitio, como así también mostrar información importante referente a él. Tal como su nombre lo indica, se ubica al pie de cada uno de los documentos HTML que conforman un sitio.

Para comenzar a elaborarlo, vamos a copiar nuestro último proyecto, **navigationbar**, a un nuevo archivo, para así modificarlo e incluir un footer acorde.



Diseño del footer

Veamos el concepto que debemos aplicar para crear la estructura HTML básica del **footer**. El elemento HTML que se utiliza es **<footer>**, y en él debemos integrar la clase **page-footer** junto con el color que queramos:

```
<footer class="page-footer grey darken-3">
    ...
</footer>
```

Este bloque de código HTML debe agregarse justo antes de la etiqueta de cierre del elemento **<body>**. Si miramos el resultado que nos devuelve este código, encontraremos una línea de color que está pegada por debajo del resto de los elementos que componen el documento HTML, tal como se observa en la siguiente imagen:

Si nuestra web contiene mucho texto e imágenes, el elemento **footer** aparecerá al final de ellos. De lo contrario, lo veremos ubicado en un lugar poco conveniente. Podemos solucionar este problema integrando código CSS para que el elemento footer se ubique al final de la página.

Fijar el footer

Crearemos a continuación un archivo CSS con el nombre **estilo.css** y lo referenciaremos dentro del apartado **<head>** de documento HTML, justo debajo del CSS que corresponde a Materialize:

```
<link rel="stylesheet" href="css/estilo.css">
```

Ahora agregamos dentro del elemento **<body>**, inmediatamente debajo de la etiqueta de cierre **</nav>**, el siguiente código:

```
<main>
    <article>
        <h1>Título</h1>
        <p>La información estándar de su sitio web va aquí, en
este espacio.</p>
    </article>
</main>
```

Vamos al código CSS del archivo **estilo.css** generado anteriormente, y allí añadimos este otro bloque de código:

```
body {
    display: flex;
    flex-direction: column;
    min-height: 100vh;
  }
main {
    flex: 1 0 auto;
  }
```

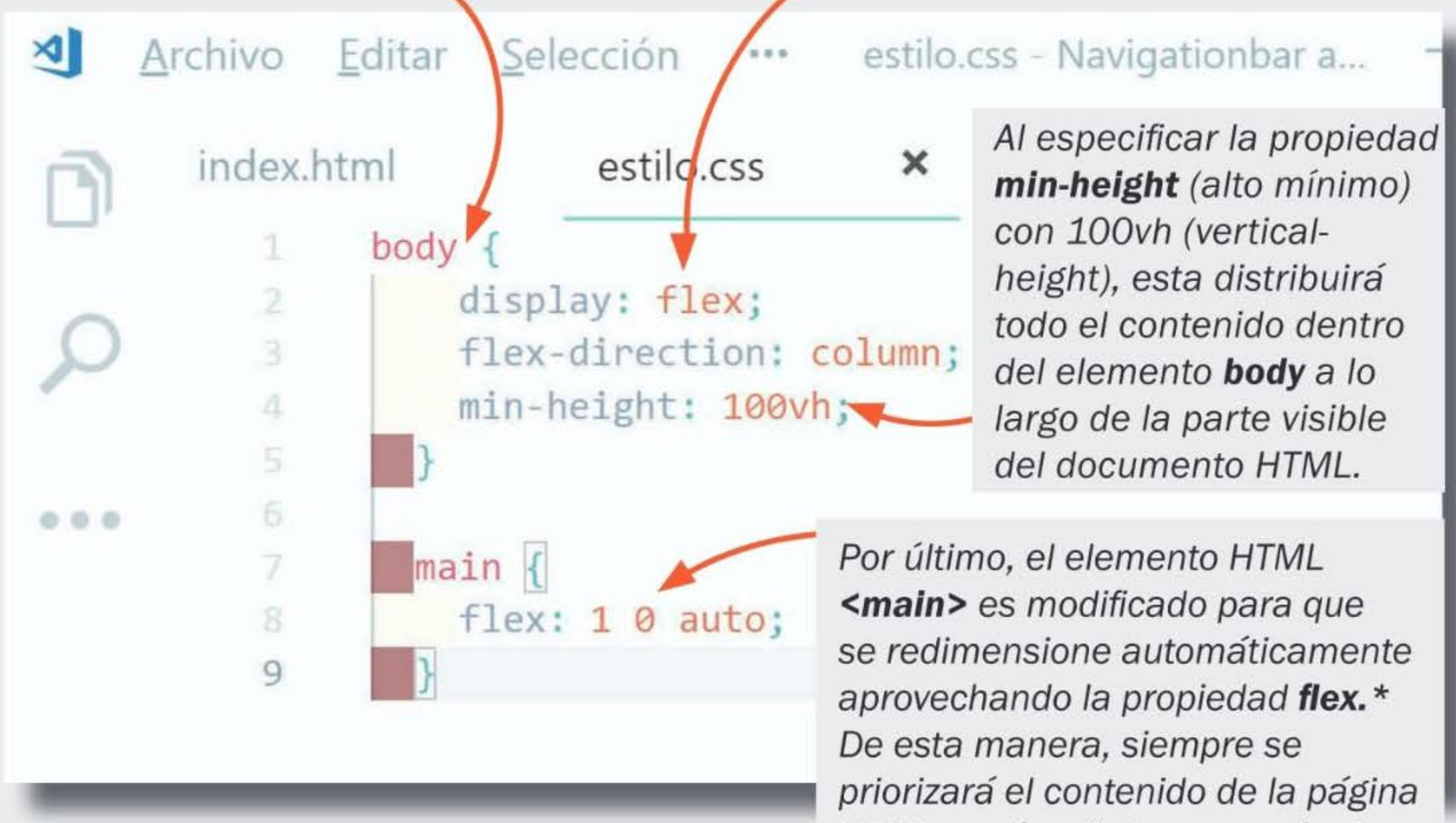
Veamos en la siguiente Guía Visual, la explicación del código CSS incorporado:

GUÍA VISUAL 3

CSS3 incluye un modo de diseño denominado flexbox, que permite alterar los elementos de una página web para que se comporten de forma predecible cuando el diseño de dicha página deba redimensionarse según la pantalla del dispositivo que invoca nuestro sitio.

*La primera declaración involucra y modifica el elemento **body**, para que sea completamente flexible, como también los elementos HTML que tenga anidados.*

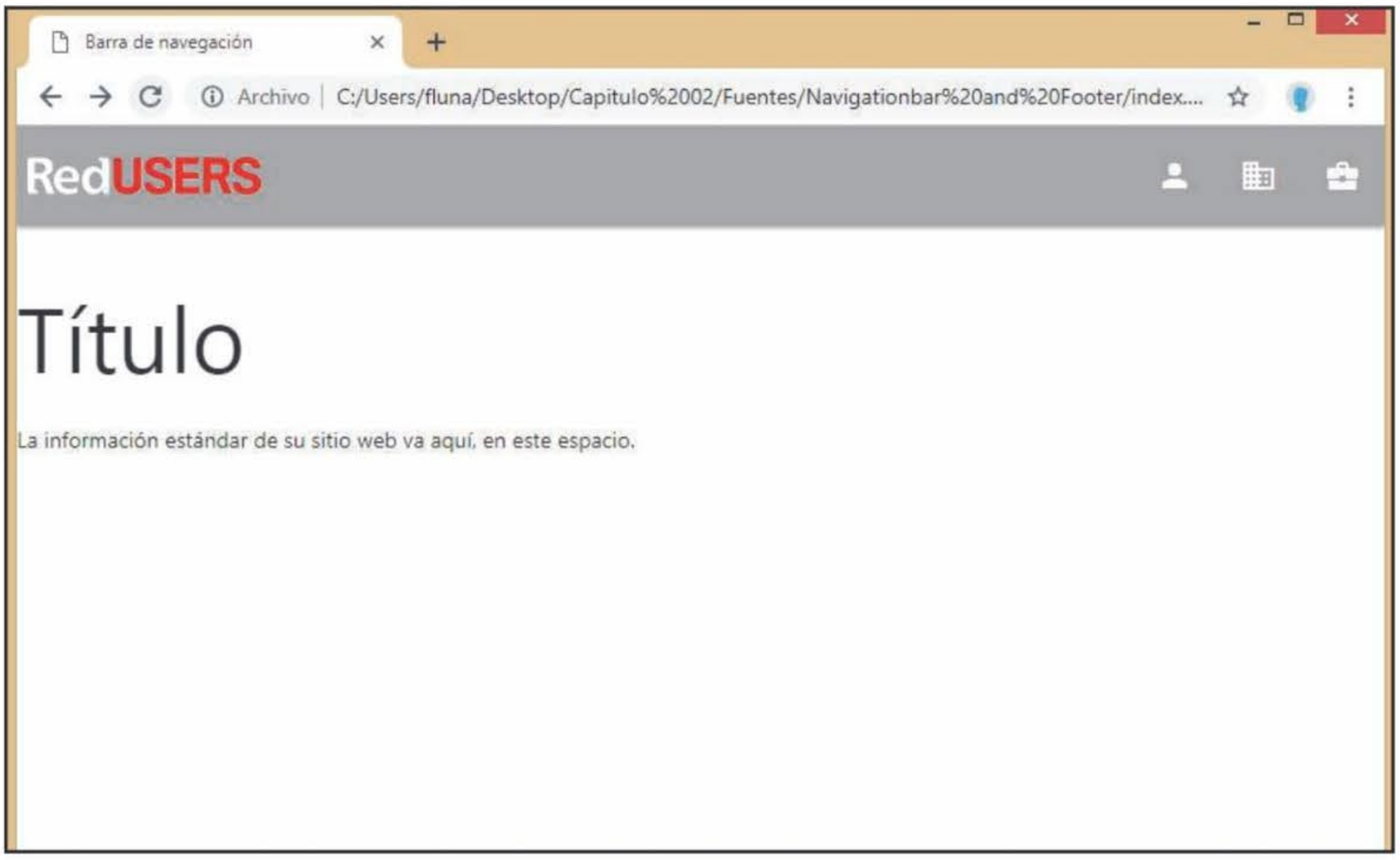
*El atributo **flex**, aplicado a la propiedad **display**, permite flexibilizar el comportamiento de este elemento HTML.*



*Al especificar la propiedad **min-height** (alto mínimo) con 100vh (vertical-height), esta distribuirá todo el contenido dentro del elemento **body** a lo largo de la parte visible del documento HTML.*

*Por último, el elemento HTML **<main>** es modificado para que se redimensione automáticamente aprovechando la propiedad **flex.** * De esta manera, siempre se priorizará el contenido de la página HTML en sí, y el elemento **<footer>** se ubicará hacia el pie de página.*

El resultado de la integración del código CSS debe ser similar al de la siguiente imagen:



El código CSS que involucra la propiedad **flex** se ocupará de enviar el elemento **footer** hacia el final del documento HTML, haya o no contenido dentro de dicho documento.

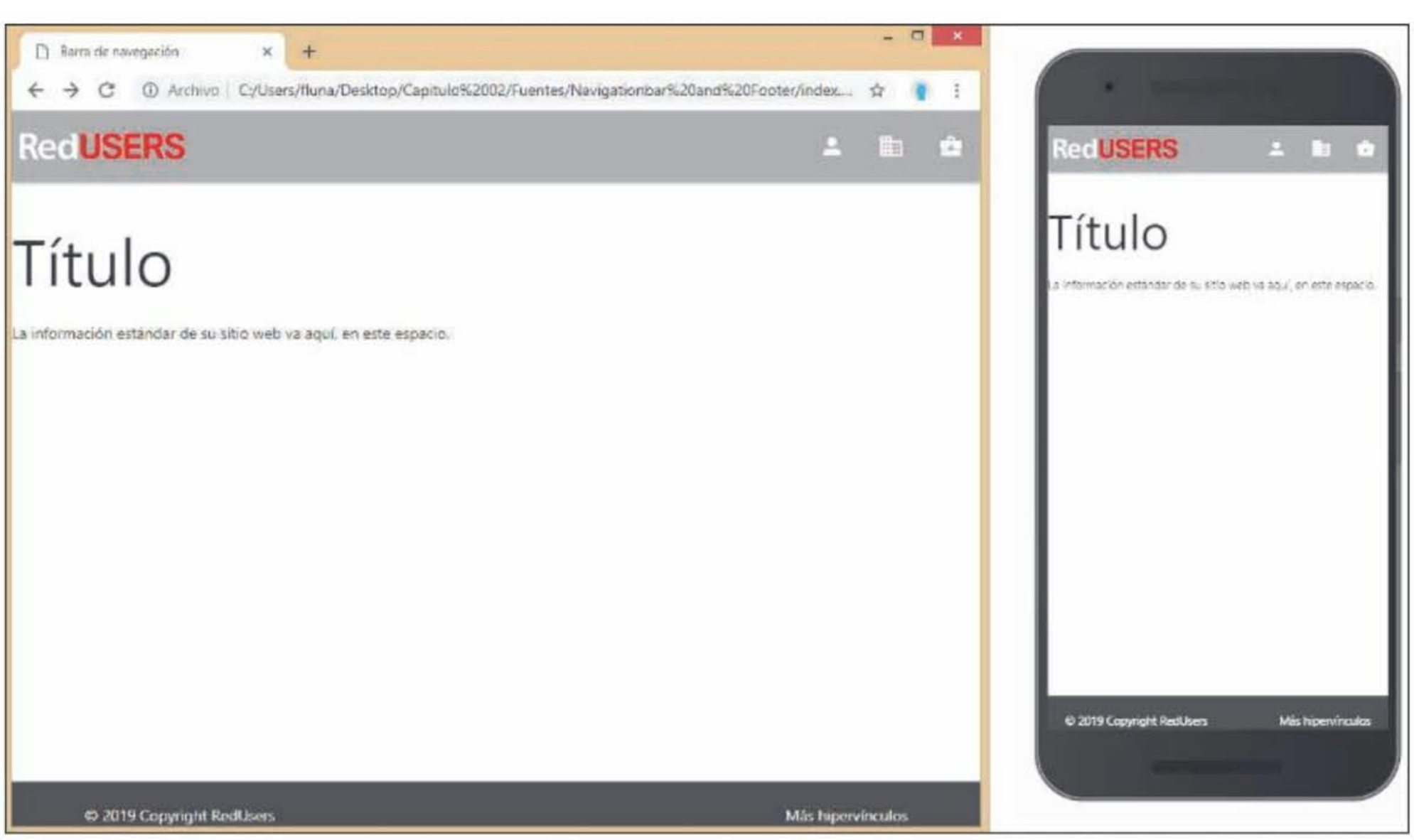
Agregar contenido al pie de página

Sigamos agregando contenido a nuestro elemento **footer**. Como ejercicio, siempre trabajando sobre el código anterior, que ya incluye la barra de navegación, agregaremos en el pie de página información de copyright relacionada con esta web, y un hipervínculo, a modo de ejemplo, para entender cómo se comportan los elementos allí contenidos.

Para hacerlo, nos ubicamos entre los tags **<footer>** y **</footer>** y agregamos el siguiente código:

```
<div class="container">
    © 2019 Copyright RedUsers
    <a class="grey-text text-lighten-4 right" href="http://www.
redusers.com/noticias/contacto/" target="_blank">Escríbanos</a>
</div>
```

Dentro del elemento **footer**, incluimos un **div** del tipo contenedor (**container**). En él declaramos un texto referente al copyright y sumamos también un hipervínculo que se abrirá, a través del atributo **target**, en una nueva ventana. A su vez, en la definición **class** del elemento **<a>** utilizamos la alineación hacia la derecha de la pantalla. El resultado debe ser similar al de la siguiente figura:



Haciendo uso de Google Chrome, podemos comprobar que el comportamiento del pie de página de nuestro documento HTML sea igual tanto en el web browser de escritorio como en un dispositivo móvil.

También podemos optar por un pie de página o footer mucho más completo, que nos permita desplegar más información sobre la web. Veamos el código a continuación, y luego el resultado de este en la figura final:

```
<footer class="page-footer grey darken-3">
    <div class="container">
        <div class="row">
            <div class="col l6 s12">
                <h5 class="white-text">Contenido del pie de página</h5>
                <p class="grey-text text-lighten-4">En este apartado
podrá desplegar toda la información necesaria de la navegación de su
sitio web.</p>
            </div>
            <div class="col l4 offset-l2 s12">
```

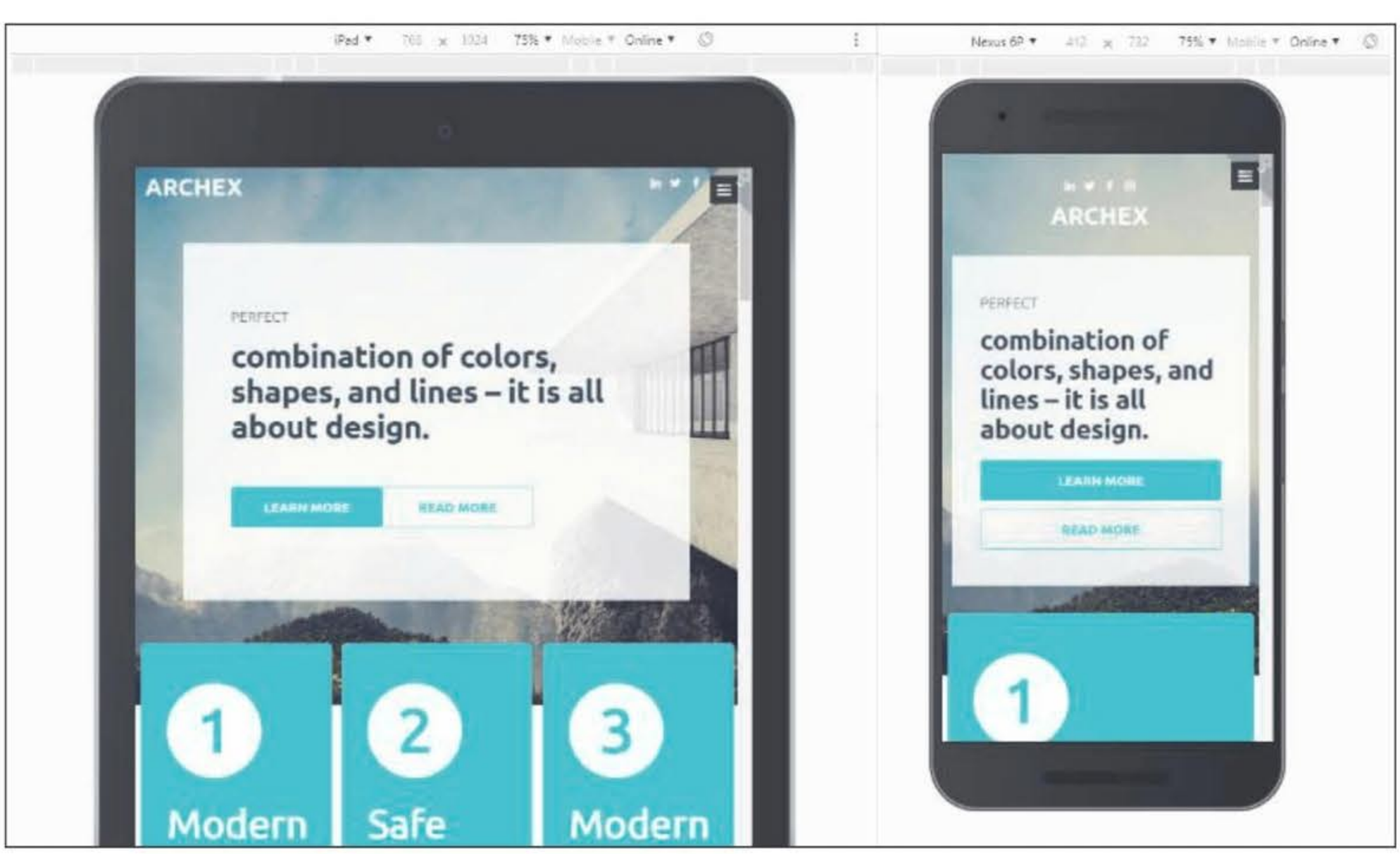
```
                <h5 class="white-text">Hipervínculos</h5>
                <ul>
                <li><a class="grey-text text-lighten-3"
href="#!">Hipervínculo 1</a></li>
                <li><a class="grey-text text-lighten-3"
href="#!">Hipervínculo 2</a></li>
                </ul>
            </div>
        </div>
        <div class="container">
            © 2019 Copyright RedUsers
            <a class="grey-text text-lighten-4 right" href="http://www.
redusers.com/noticias/contacto/" target="_blank">Escríbanos</a>
        </div>
    </div>
</footer>
```

¿Se animan a modificar este footer para darle vida con sus propios contenidos?

03 Estructura y estilos Web

La estética de una web juega hoy un papel importante, ya que es la que genera en los usuarios el primer impacto visual. Veamos algunos trucos y estilos que podemos aplicar en nuestros diseños para conseguir un efecto notorio ante los visitantes.

El diseño de un sitio web debe darle, desde su idea inicial, una fuerte importancia al impacto visual. En algunos casos, este se puede aplicar recurriendo al diseño gráfico o animado, o con imágenes atractivas; en otros casos, debemos pensar cómo llamar la atención de quien navega la web desde las palabras o el estilo de los textos y títulos. Sumado a esto, cuando trabajamos para el terreno mobile, es preciso sumar, al desarrollo y el diseño, un fuerte poder de síntesis en cada título, párrafo o texto escrito con el cual los visitantes se encontrarán.



Para lograr este objetivo, veamos a continuación qué opciones nos ofrece Materialize CSS y otros jugadores de este terreno, que nos ayudarán a impactar visualmente a través del diseño del sitio o aplicación web.

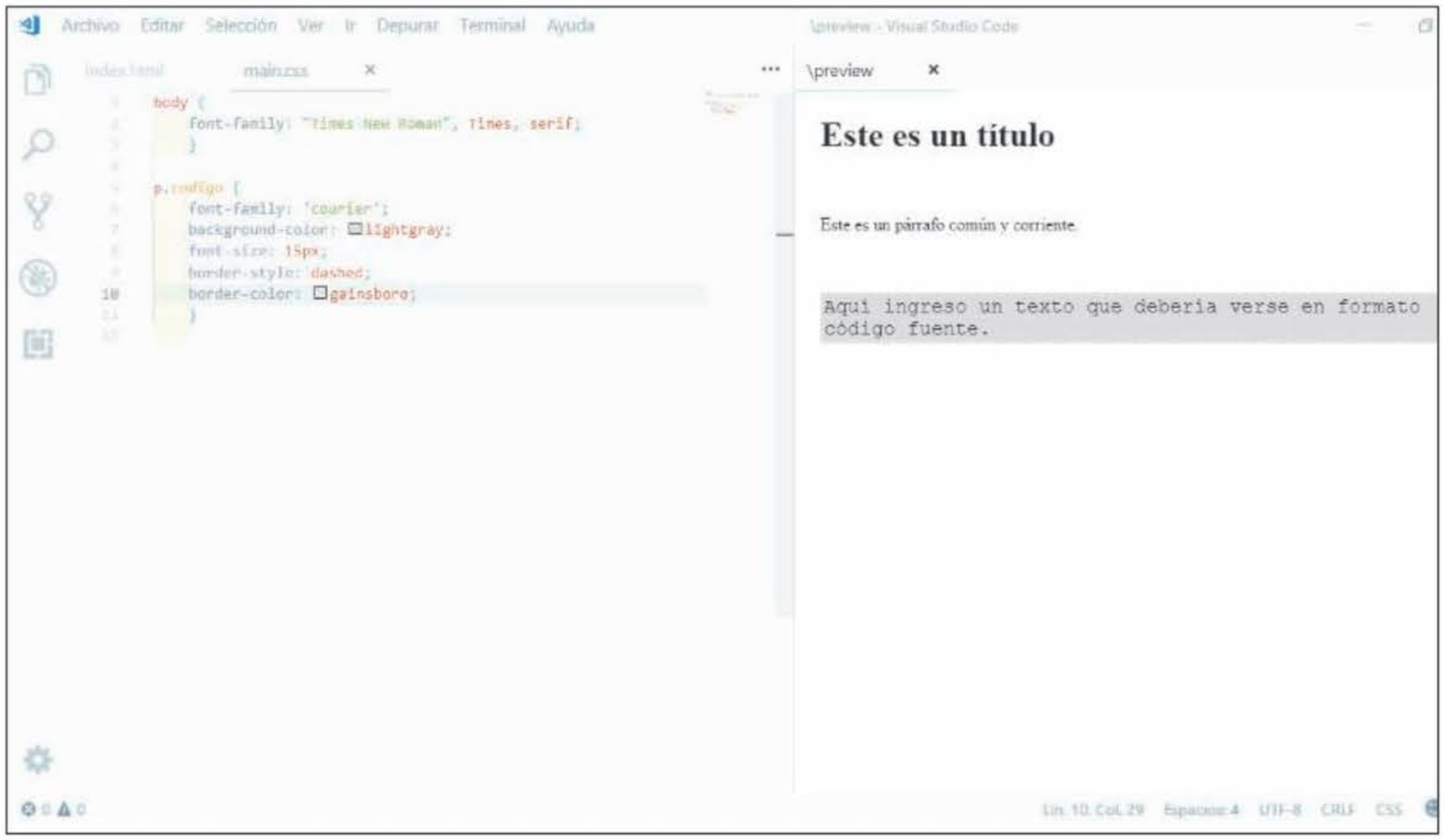
Tipografías

Las tipografías o fuentes determinan el diseño de los textos que componen un sitio web. Por defecto, los navegadores en general traen incluido el soporte de tipografías estándar, el cual abarca, por ejemplo: **Courier**, **Arial**, **San Serif** y **Verdana**, entre otras. Para usarlas en un proyecto, podemos declararlas dentro de la hoja de estilos, de esta manera:

```
body {
    font-family: "Times New Roman", Times, serif;
    }
```

De acuerdo con este código, la familia de fuentes será aplicada a cada uno de los elementos contenidos en el **body**. Ahora, si queremos utilizar una fuente específica para un determinado componente web, lo hacemos de la siguiente forma:

```
p.codigo {
    font-family: "courier";
    background-color: lightgray;
    font-size: 15px;
}
```

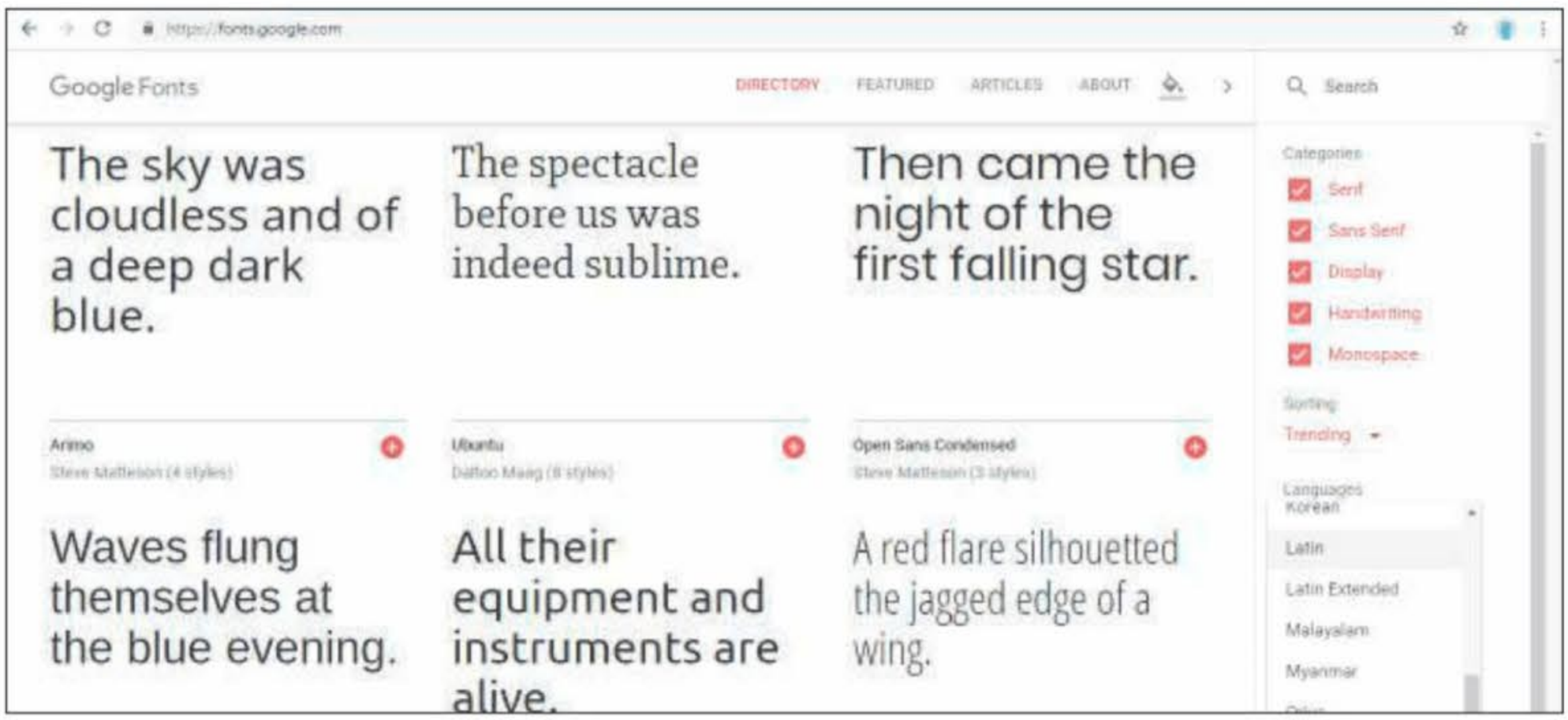


Tipografías web

En los inicios del desarrollo web, solo podíamos utilizar las llamadas "fuentes seguras", que garantizaban la correcta visualización de un sitio. Estas fuentes son las que especificamos al inicio de este capítulo. La evolución de HTML y CSS permitió, con el tiempo, incluir otro tipo de fuentes, las llamadas **web fonts** o **fuentes web**. Están construidas a partir de una hoja de estilos del tipo **CSS3**, y no es necesario tenerlas instaladas en el servidor web, ya que se pueden utilizar referenciando el sitio que las contiene.

Google Fonts

Uno de los repositorios de fuentes web más utilizados actualmente es **Google Fonts**. Allí podemos encontrar una variedad increíble de tipografías basadas en **CSS3**, e incluirlas sin costo en nuestros diseños web o mobile.



Cuando repasamos las bases necesarias para diseñar integrando Materialize CSS, aprendimos a agregar una tipografía web, referenciándola en el apartado **<head>** del documento HTML a través de CSS. Por ejemplo:

```
<link href="https://fonts.googleapis.com/css?
      family=PT+Sans" rel="stylesheet">
```

Ahora, si queremos integrar múltiples fuentes web en un diseño, tenemos que realizar la referencia hacia ellas de esta manera:

```
<link href="https://fonts.googleapis.com/css?
      family=PT+Sans|Raleway" rel="stylesheet">
```

GUÍA VISUAL 1

A través del carácter **|**, *conocido como* ***Pipe***, ***Bar*** *o* ***barra vertical***, *agregamos cada una de las fuentes web que deseamos utilizar en un sitio.*

Fuente tipográfica ***web PT+Sans****: es la primera fuente que se visualizará al cargar la página.*

Fuente tipográfica ***Raleway****: es la segunda fuente que se visualizará al cargar la página, en caso de que la primera no esté disponible.*

```
<link href="https://fonts.googleapis.com/css?
      family=PT+Sans|Raleway" rel="stylesheet">
```

Para utilizar ambas fuentes en diferentes componentes web de un proyecto, debemos invocarlas utilizando la propiedad CSS ***font-family****, y seleccionando el orden que deseamos, o incluyendo para un elemento HTML específico una de las fuentes web.*

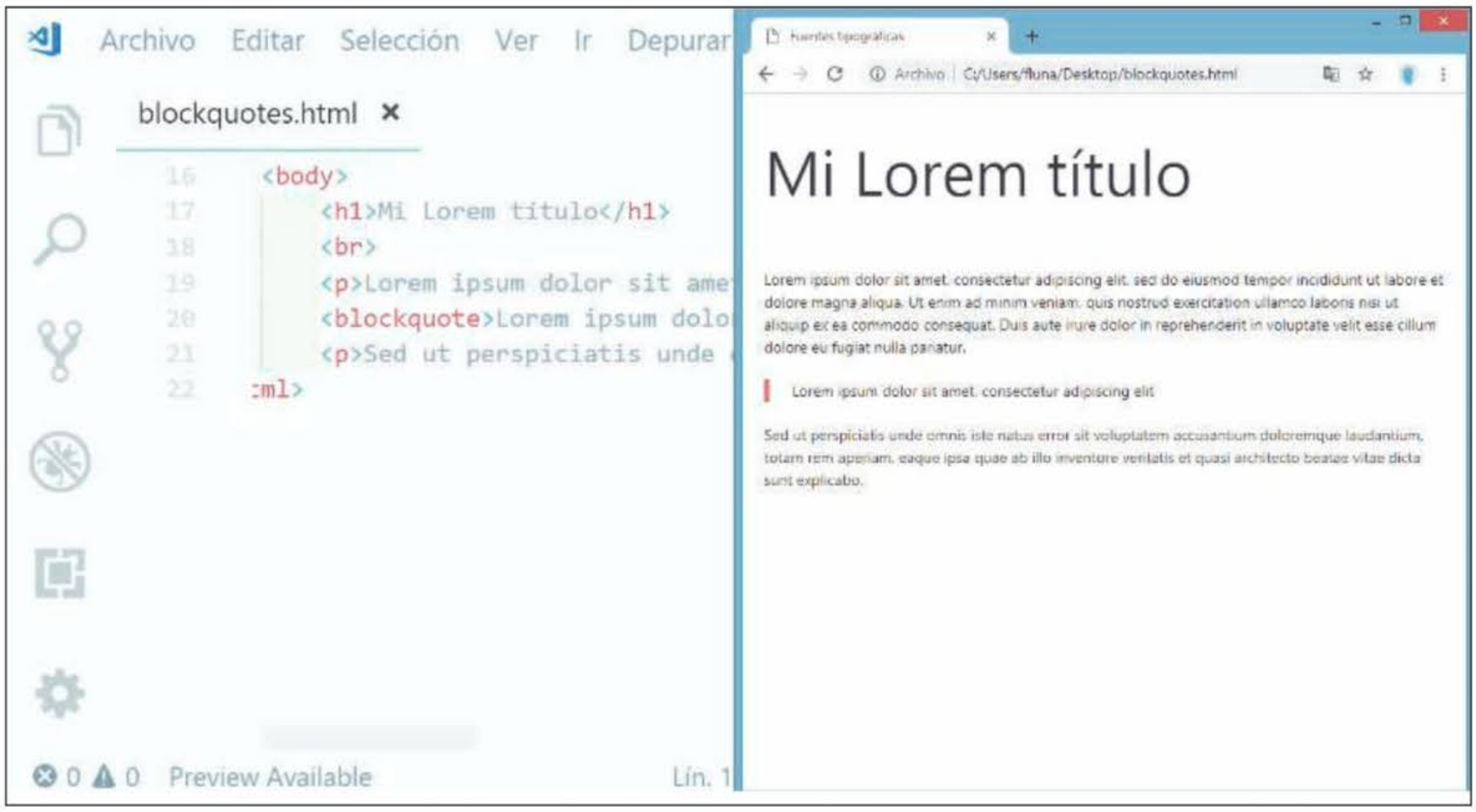
Blockquotes (énfasis en párrafos)

Si hablamos de fuentes y diseño, no podemos dejar pasar que, ante determinadas situaciones en las que existen textos de lectura mediana o extensa, el resaltado de una frase u oración que resuma un tema o concepto siempre es una muy buena práctica. Para este tipo de necesidad, Materialize CSS propone el uso del tag HTML **blockquotes**. Este se encuentra preformateado por Materialize a través de CSS. Veamos un ejemplo:

```
<p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do
eiusmod tempor incididunt ut labore et dolore magna aliqua...</p>
<p>Sed ut perspiciatis unde omnis iste natus error...</p>
```

Tenemos dos párrafos (mediante los elementos html **<p>**) con textos extensos, y queremos resaltar una oración de uno de ellos. Agregamos el elemento HTML **<blockquote>** englobando la oración por destacar:

```
<blockquote>Lorem ipsum dolor sit amet, consectetur adipiscing elit</
blockquote>
```



Flow Text

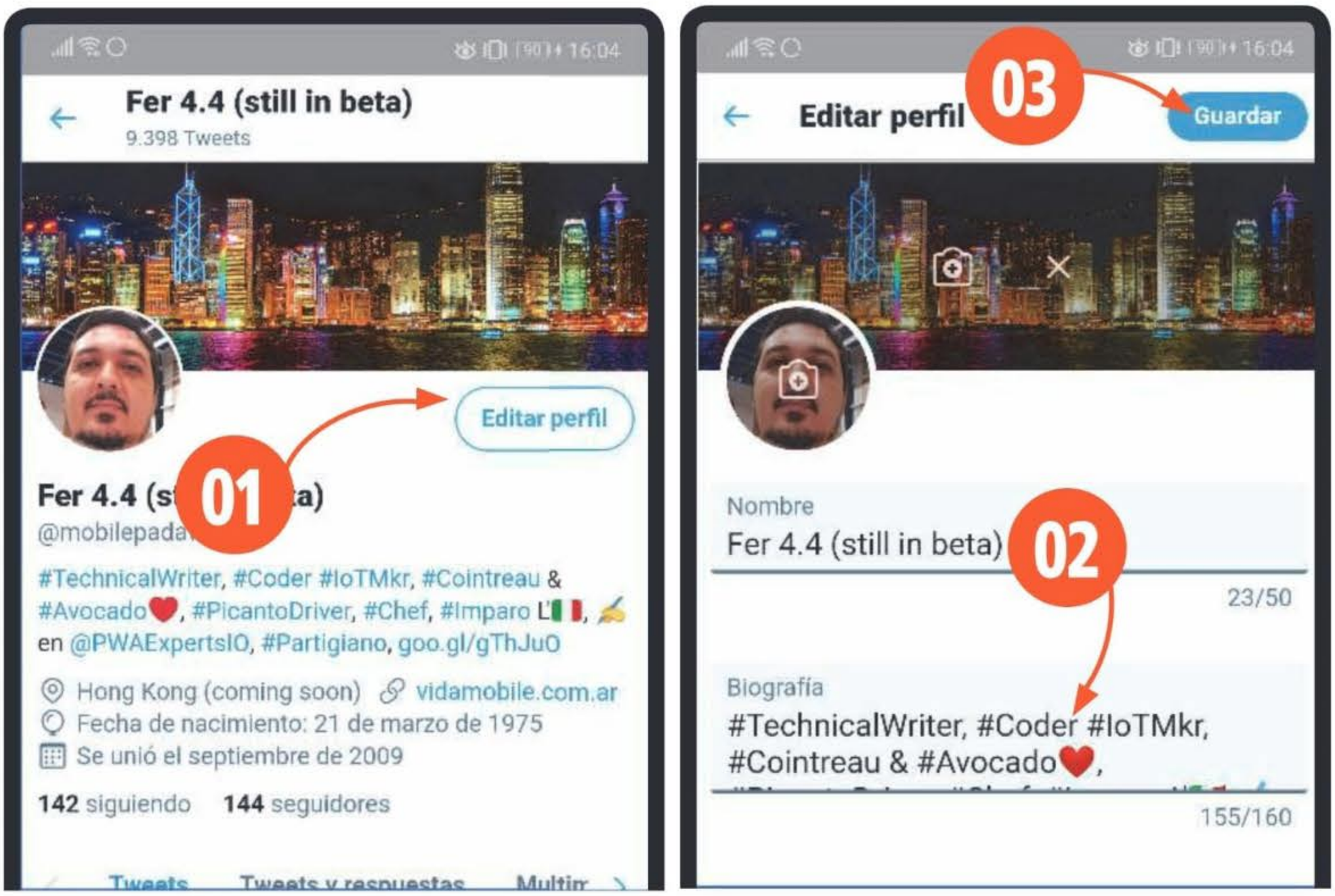
Seguramente habremos encontrado estos últimos años diversos sitios que adaptaron sus sistemas de lectura incluyendo en algún rincón la posibilidad de aumentar o disminuir el tamaño de la fuente utilizada. La clase **Flow-Text** de Materialize CSS propone algo similar, expandiendo o contrayendo un determinado párrafo dentro de su elemento HTML contenedor, para que la lectura resulte mucho más amena.

La forma de uso más común para esta clase es combinarla con JavaScript, la cual invoca la propiedad **ClassList** y su método **Toggle()**, agregando o quitando dicha clase del elemento HTML que deseamos modificar.

En el apartado de recursos que acompaña a esta obra encontrarán los ejemplos funcionales del uso de **fuentes comunes**, **blockquotes** y **flow text**, explicados hasta aquí.

Párrafos editables (content editable)

La simplificación de la interactividad con una web o una app es muy valorada por los usuarios, y HTML5 cuenta con algunas funcionalidades de fácil y rápida implementación. Si pensamos, por ejemplo, en la edición de datos de nuestra **bio** de **Twitter**, debemos ir primero a nuestro perfil, presionar el botón **Editar Perfil** y finalmente llegar al lugar de edición de la info personal.



Esta última interacción puede ser perfectamente evitable, si agregamos sobre el elemento HTML que contiene nuestra bio el atributo **ContentEditable**. Este acepta un valor booleano que determina si el texto se puede modificar o no:

```
<p id="mibiografia" contenteditable="true">Lorem Ipsum Dolor Sit Amet...</p>
```

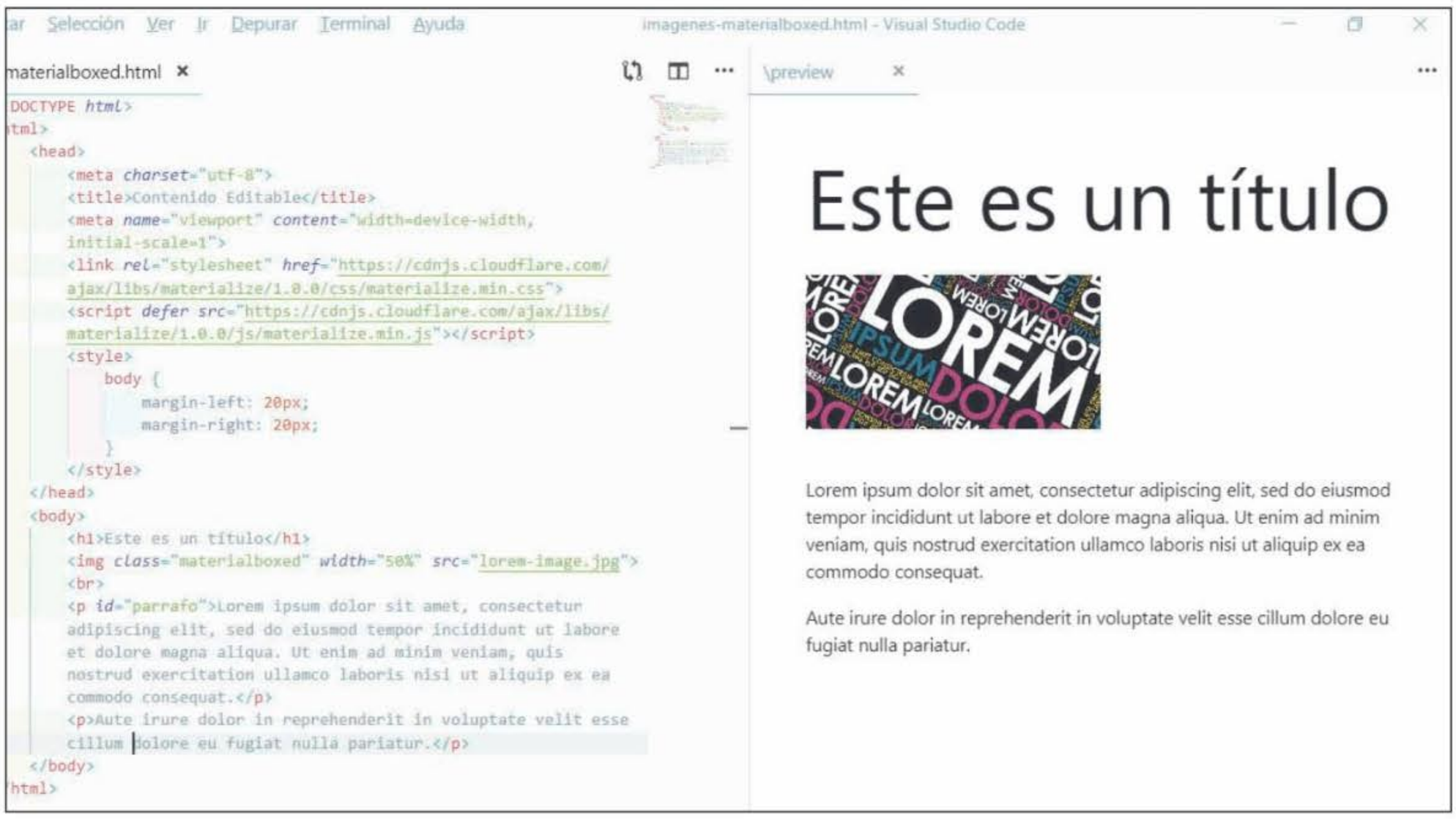
La edición del párrafo permite no solo cambiar el texto, sino también aplicarle un formato rápido de negrita, cursiva o subrayado, entre otros. Luego, para que este persista, se deberá buscar un mecanismo efectivo de guardado de dichas modificaciones.

Imágenes: la clase materialboxed

El tag HTML **<img>** cuenta con una clase Materialize denominada **materialboxed**. Su función es modificar ligeramente una imagen clásica insertada con el tag html, agregándole un leve efecto de contraste cuando deslizamos el dedo o cursor sobre ella. Su uso es el siguiente:

```
<img class="materialboxed" width="100%" src="lorem-image.jpg">
```

A su vez, esta clase CSS se combina con JavaScript para generar un efecto zoom al pulsar sobre ella. Ese efecto modifica el tipo de cursor por defecto (en computadoras de escritorio) y muestra la imagen en pantalla completa.



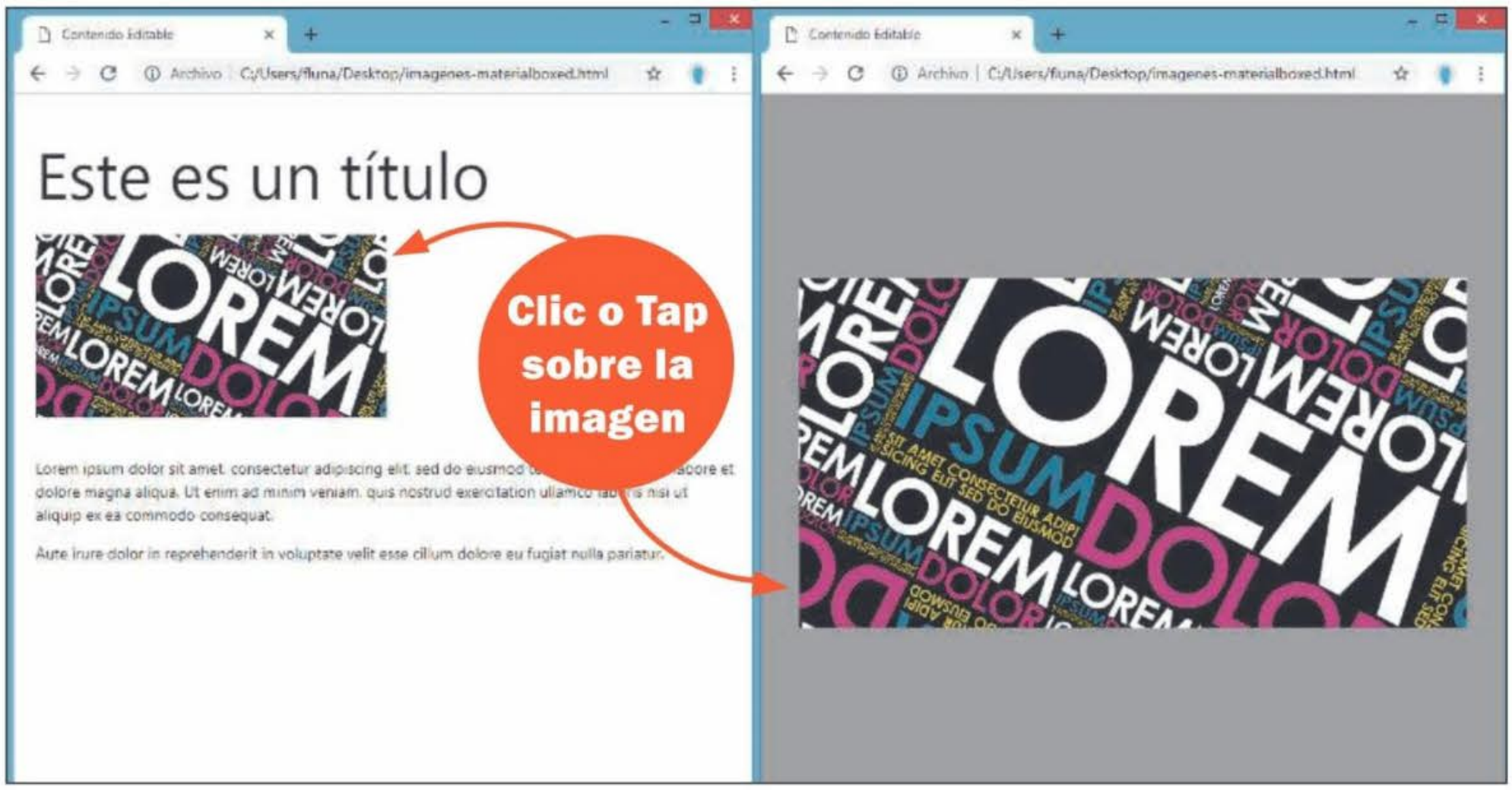
A continuación, exploremos de qué forma JavaScript nos permite modificar el comportamiento de las imágenes con la clase **materialboxed**, en pos de ampliarla en pantalla mediante el efecto de zoom. Descarguemos el ejemplo **Materialboxed.zip**, del repositorio de archivos que acompaña a esta obra.

Descomprimimos el archivo y abrimos el proyecto con Visual Studio Code. Hacemos **Ctrl + clic** sobre la referencia al archivo **mb.js**, y seleccionamos **Create File**. Ya en el archivo JS, escribimos el siguiente código:

```
document.addEventListener("DOMContentLoaded", function() {
    var elems = document.querySelectorAll(".materialboxed");
    var options = "inDuration: 300, outDuration: 600";
    var instances = M.Materialbox.init(elems, options);
  });
```

La variable **elems** permite seleccionar, a través de **querySelectorAll()**, todos aquellos elementos HTML cuya clase CSS es **materialboxed**. Luego, la variable **options** almacena dos propiedades que se le pueden asignar a la imagen: en este caso, ponemos un delay de 300 milisegundos para que la imagen se abra a pantalla completa (**inDuration**), y 600 milisegundos para cuando la cerramos (**outDuration**). Finalmente, iniciamos la instancia de este efecto mediante **M.Materialbox.init()**, pasándole a este método los dos parámetros anteriormente creados en las respectivas variables.

Para terminar, probamos el código ejecutando el documento HTML en un navegador web. Este tendrá que mostrar la imagen a pantalla completa al hacer tap o clic sobre ella.



En la web de Materialize CSS, https://materializecss.com/media.html, dentro del apartado **Options**, encontramos todas las opciones que podemos pasarle como parámetro para que se generen diversos efectos y comportamientos sobre la imagen en cuestión.

Breadcrumbs (jerarquía de navegación)

Por lo general, cuando damos con sitios web que contienen una cantidad importante de información, vemos que esta se va estructurando de manera jerárquica, con el uso de categorías o secciones y subcategorías/subsecciones de las primeras. Materialize CSS nos propone el uso de la clase **breadcrumbs**, la cual nos permite dejar un rastro visible desde el punto inicial de navegación.

Su uso es muy simple:

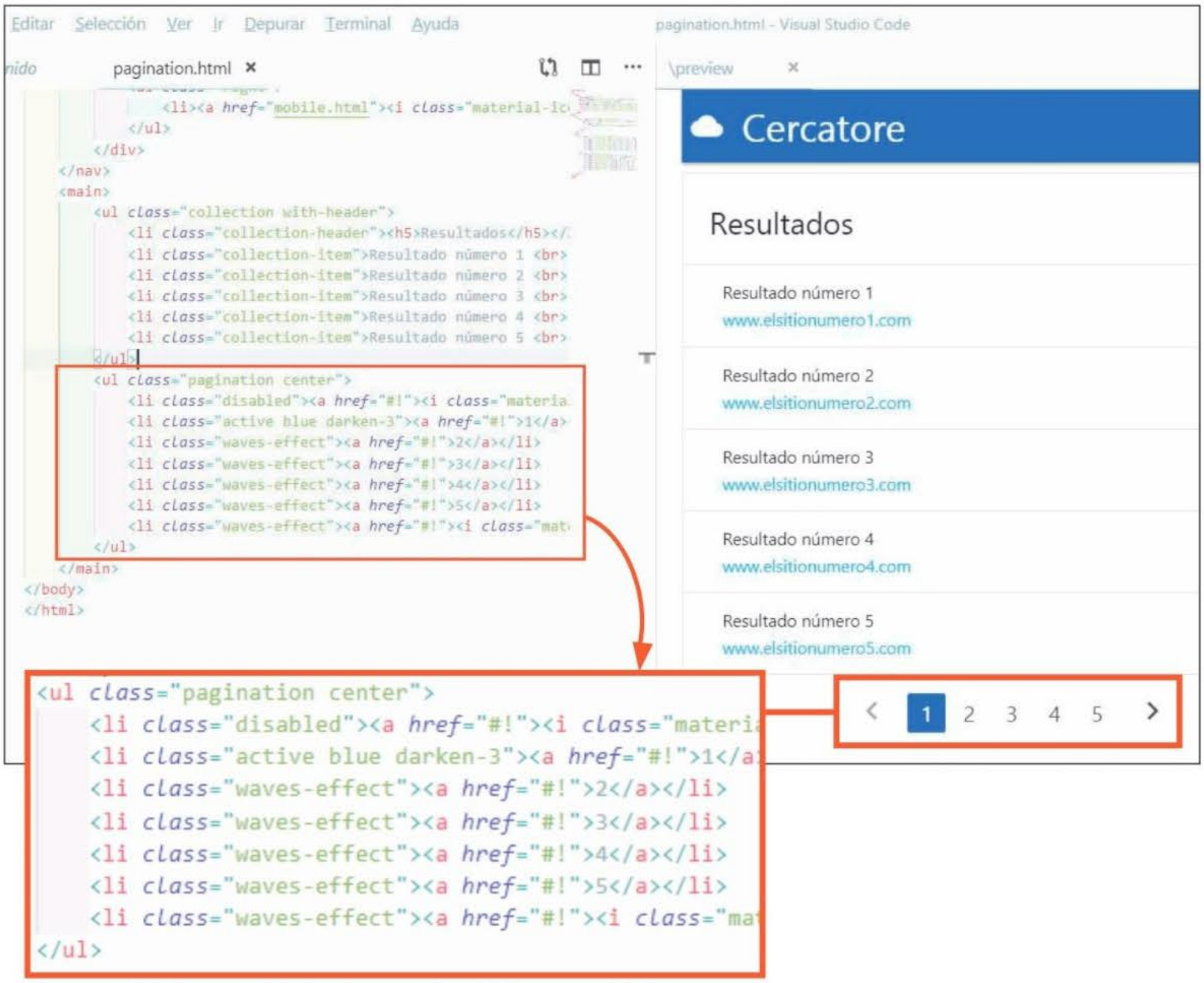
```
<a href="#!" class="breadcrumb">Home</a>
<a href="#!" class="breadcrumb">Nosotros</a>
<a href="#!" class="breadcrumb">Valores</a>
```

Como vemos en el código anterior, sobre cualquier elemento HTML **<a>**, podemos aplicar la clase en cuestión, para que este y los mismos elementos subsiguientes muestren un **path** o ruta desde el inicio de la navegación, hasta el punto en que nos encontremos en este momento. Cada elemento **<a>** contará por supuesto con su hipervínculo asociado, que le permitirá al usuario volver rápidamente uno o más niveles anteriores.

Pagination (estructurar el cúmulo de información)

El mejor ejemplo del uso de la clase **pagination** es el pie de página de resultados de cualquier búsqueda que realizamos en un **buscador**. A diferencia de breadcrumbs, el uso de esta clase se complementa a través de los elementos HTML **<ul>** y **<li>** (listas), especificando al primero de ellos el tipo de clase que debe adoptar:

```
<ul class="pagination">
    <li class="active"><a href="#!">1</a></li>
    <li class="waves-effect"><a href="#!">2</a></li>
    <li class="waves-effect"><a href="#!">3</a></li>
    <li class="waves-effect"><a href="#!">4</a></li>
</ul>
```



04 Formulario de contacto

Los formularios de contacto o suscripción son de uso cotidiano en el universo web y web móvil. Y, para diseñar formularios aptos para este ecosistema, en primer lugar debemos comprender los diferentes campos de entrada adaptados para smartphones y tablets.

El campo de entrada de un formulario HTML estándar se basa originalmente en el tag HTML **<input type>**.

GUÍA VISUAL 1

Aquí detallamos el tipo de campo que se va a mostrar: texto, número, fecha, etcétera (ver Tabla 1).

*A través del atributo **pattern**, podemos limitar el tipo de dato a ingresar (texto, número, solo mayúsculas, etcétera).*

```
<input type="tipodecampo" id="nombre" pattern="formato condicional" placeholder="Texto de ayuda">
```

*El **ID** es el nombre habitual que se les da a los componentes HTML para poder manipularlos desde un lenguaje como JS.*

*El atributo **placeholder** permite agregar un texto de ayuda en el fondo del campo, el cual se borra cuando se ingresa o selecciona un valor para él.*

Esto nos permite crear campos de texto, números, fechas, y demás. Y con la llegada de HTML5, las opciones de campos aumentaron mucho más, para poder sacar partida y minimizar el error en el ingreso de un tipo de dato específico.

Materialize CSS cuenta con componentes HTML denominados **text input**, los cuales utilizan y potencian el **<input type>** estándar de HTML5, agregándole un formato estético más agradable, y otros controles y componentes adicionales que enriquecen la experiencia de uso de estos campos.

TABLA 1

<input type>	Descripción
text	Permite ingresar cualquier tipo de caracteres alfanuméricos.
number	Permite ingresar solo números (positivos, negativos y/o con decimales).
date	Permite ingresar una fecha compuesta por día, mes y año.
email	Permite ingresar una dirección de correo electrónico.
range	Permite ingresar un rango numérico con atributos mínimos y máximos definibles por el desarrollador.
month	Permite ingresar un mes del año calendario.
password	Permite el ingreso de una contraseña o clave.
search	Despliega un campo de búsqueda.
tel	Permite ingresar un número de teléfono, aceptando espacios, guiones, numeral y asterisco como caracteres extendidos.
time	Permite ingresar un dato del tipo HH:MM:SS.
url	Permite el ingreso de una URL, forzando el ingreso de HTTP / HTTPS, los dos puntos y contrabarras, y obviando los caracteres extendidos que no pueden componer una URL.
week	Permite ingresar el número de una semana del año, limitando los valores entre 1 y 53.
checkbox	Despliega un campo del tipo **check**, pudiendo funcionar de forma independiente o agrupado con otros.
radio	Despliega un campo del tipo radio, pudiendo funcionar de forma independiente o agrupado con otros.
color	Despliega en algunos navegadores web la paleta de colores del sistema, para seleccionar uno determinado.
file	Habilita el cuadro de diálogo del sistema para seleccionar uno o más archivos del dispositivo o computadora.

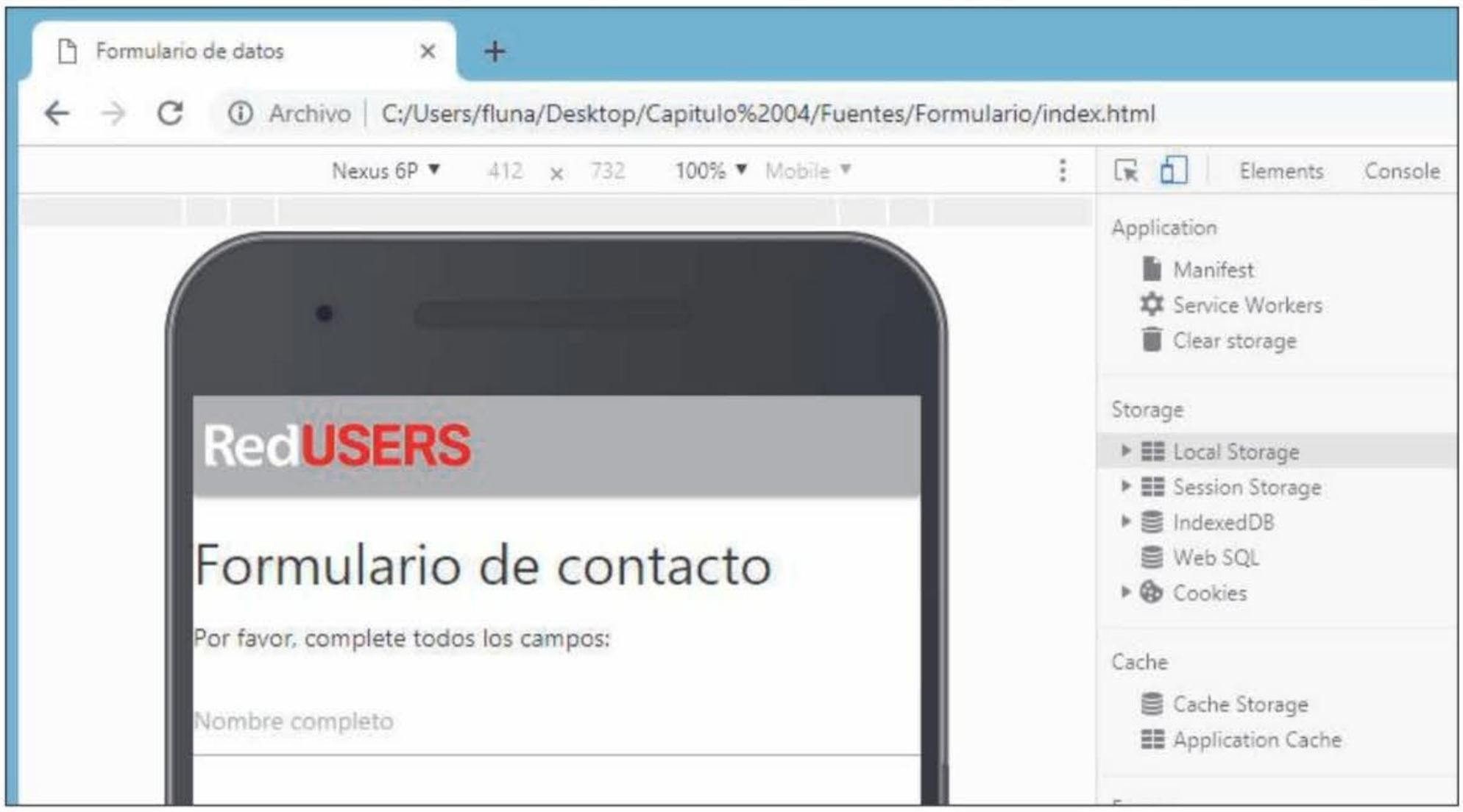
Estructura de los text input de Materialize CSS

El código para generar un campo de entrada o **text input**, se realiza de la siguiente manera:

```
<div class="row">
    <div class="input-field ">
        <input id="nombre" type="text" class="validate">
        <label class="active" for="nombre">Nombre completo</label>
    </div>
</div>
```

Materialize hace diferencia en la creación de un campo del tipo input type. Se incluyen una serie de etiquetas **div** para estructurar el contenido. La primera de ellas crea una fila mediante el atributo **class="row"**; el campo de texto ocupará por defecto el ancho del documento web donde se despliega. Luego, se incluye el div con la clase **input-field**. A continuación viene el input type en cuestión con el atributo **class="validate"**, clase que maneja el color de la barra inferior del campo.

El metatag **<label>** define un nombre para el campo, reemplazando al atributo **placeholder** visto anteriormente, e interactúa como este último hasta que el usuario hace foco en el campo. Cuando esto ocurre, este metatag pasa a ser un título externo. Para relacionar el metatag label con el campo, hay que sumar el atributo **for=""** y, como valor de este, el nombre del campo en cuestión.

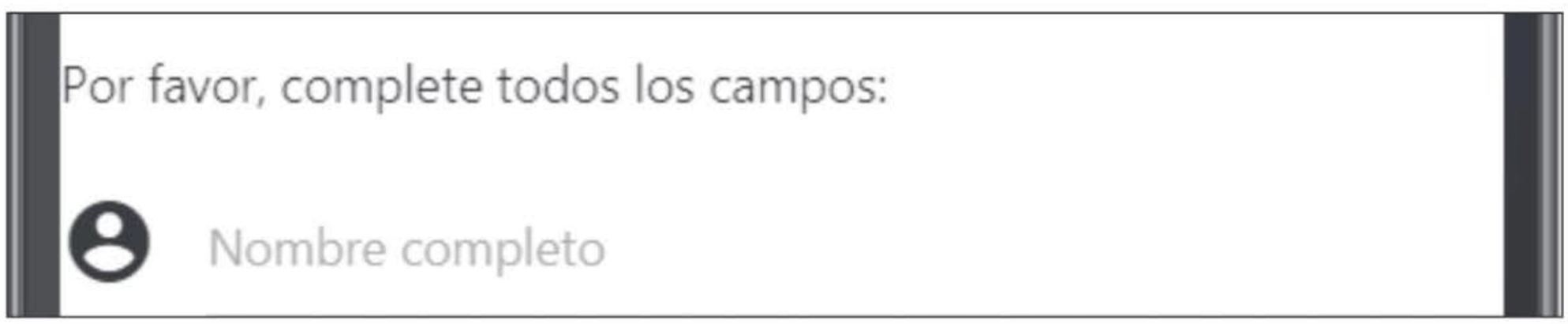


Agregar iconos a cada campo

Podemos agregar un icono al lado de cada campo, anteponiendo al metatag **<input id>**, tal como vemos en la siguiente línea de código:

```
<i class="material-icons prefix">account_circle</i>
```

El resultado es similar al de la siguiente figura:



Diseño de un formulario

Descarguemos del repositorio de archivos que acompaña a esta obra el proyecto **Formulario-Materialize-base.zip**. Descomprimimos la carpeta y creamos un nuevo proyecto en VS Code, con este contenido. Lo que haremos a continuación será agregar una serie de campos que conformen un completo formulario de datos.

Como base tomamos el código HTML del input text existente, y lo replicamos, modificando el nombre de cada campo y el tipo de dato que acepta. Los campos que crearemos son:

- *Fecha de nacimiento*
- *Email*
- *URL*
- *Comentarios*

La estructura debe quedar como se observa en la imagen.

Validación de datos

Acorde a lo descripto en la sección del **text input** que maneja Materialize, la clase **validate** se ocupa de validar el contenido ingresado en cada campo. Ahora, ¿cómo podemos verificar que esto funciona? ¡Fácil!, debemos ejecutar el proyecto e ingresar en cada campo validado un valor diferente del cual acepta. Si nuestra clase HTML de validación está bien definida, veremos que el borde inferior de cada campo se torna de color rojo.

Mensajes de confirmación o error personalizados

Materialize también ofrece una manera de notificar al usuario con mensajes de confirmación o error sobre los datos ingresados, mucho más descriptivos.

```
<span class="helper-text" data-error="Fecha ingresada errónea" data-
success="Ok"> </span>
```

Para esto, seguido a la etiqueta **<label>** de cada text input, agregamos el metatag **** con la clase **helper-text**. Esta clase contiene dos atributos, **data-error** y **data-success**, en los cuales podemos especificar el mensaje de error o de validación sobre el dato ingresado.

Botón de envío de formulario

Agregamos a continuación el botón que nos permitirá enviar el formulario, una vez completo. Para hacerlo, utilizamos el metatag **button**:

```
<div class="row center">
    <button class="waves-effect waves-teal btn-flat"
type="submit" name="action">Enviar
        <i class="material-icons right">send</i>
    </button>
</div>
```

En este código, creamos una nueva fila alineada al centro del documento HTML, y añadimos dentro de ella el metatag **<button>**; el atributo **class**

almacena clases que definen su estilo (**btn-flat** = plano) y los efectos animados que tendrá cuando se pulse sobre él. También incluye el atributo **type="submit"**, el cual se relaciona con la acción de enviar el formulario de datos, y el **name="action"**, que identifica con un nombre similar al ID dentro del formulario.

Metatag form

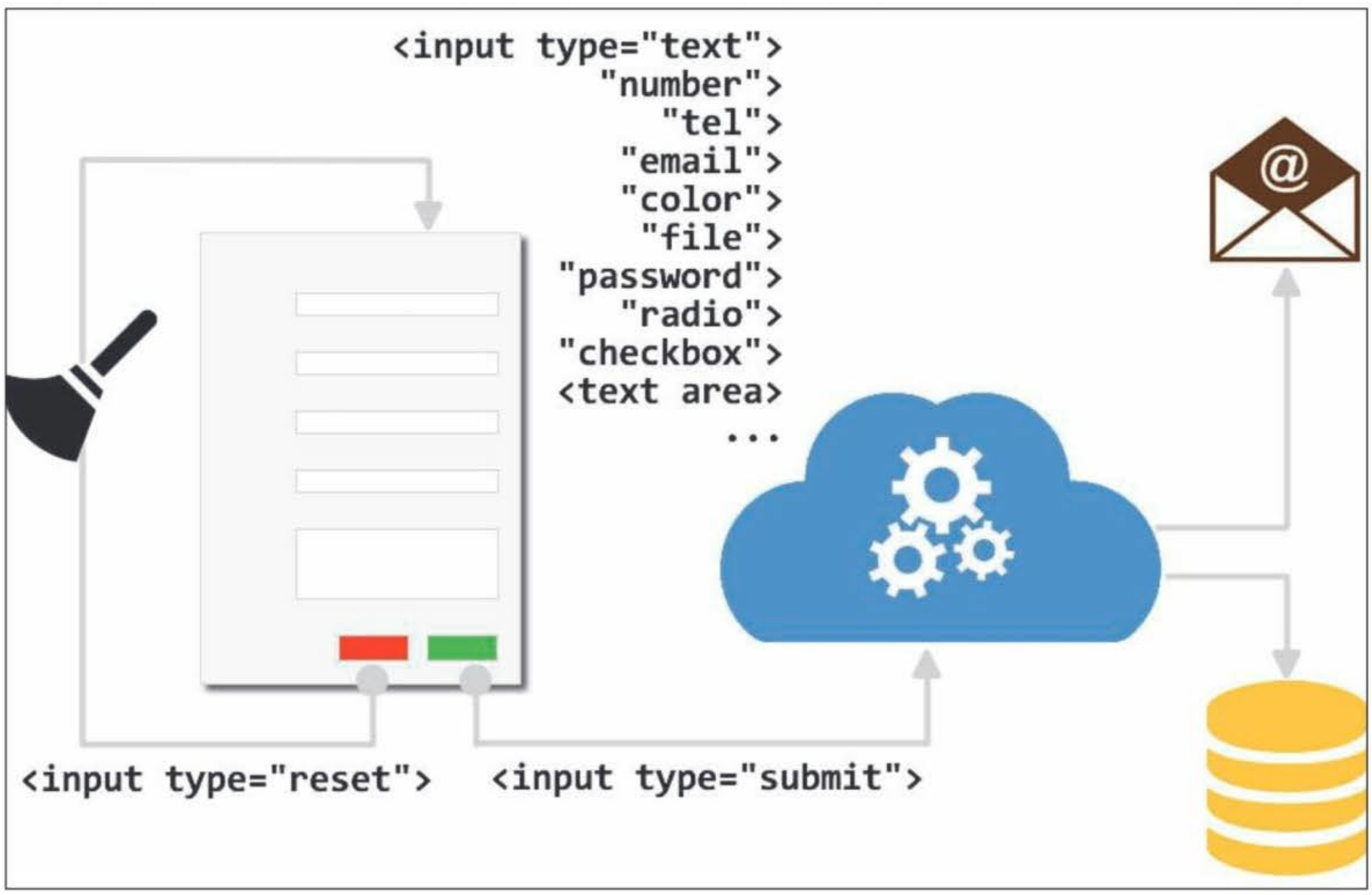
Modificamos a continuación el HTML para poder incluir el metatag **<form>**, que debe contener todos los elementos HTML del formulario. Para esto, en el código HTML, luego del metatag **<main>**, agregamos la siguiente línea:

```
<form action="acción para enviar formulario">
```

Y antes del metatag de cierre **</main>**, incluimos esta otra línea:

```
</form>
```

En la siguiente figura, se representan las acciones que se pueden realizar cuando presionamos el botón **Enviar**:

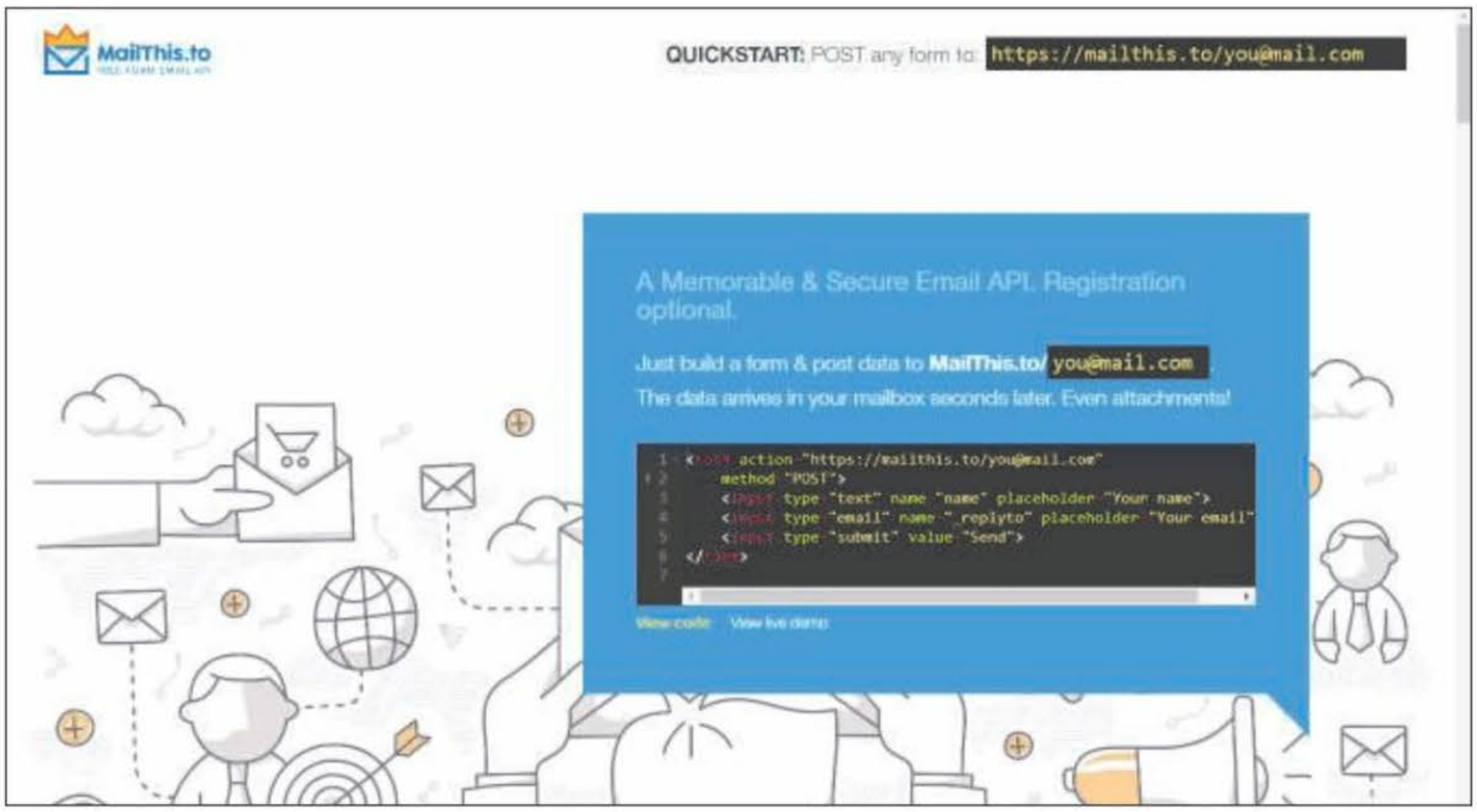


Como podemos ver, al presionar el elemento **<button type="submit">**, todos los datos cargados en el formularios son enviados hacia un destino específico. Este destino (la nube) puede tener dos rumbos: el primero, una dirección de correo electrónico; y el segundo, una posible base de datos o archivo de datos.

El resultado de esta acción siempre depende de un programa alojado en el servidor que se ocupa de leer y recopilar la información cargada en cada campo, y darle el curso correspondiente.

El servicio Mailthis.to

Esta web ofrece un servicio rápido y fácil de implementar, para recibir en una cuenta de correo electrónico los datos cargados en un formulario web.



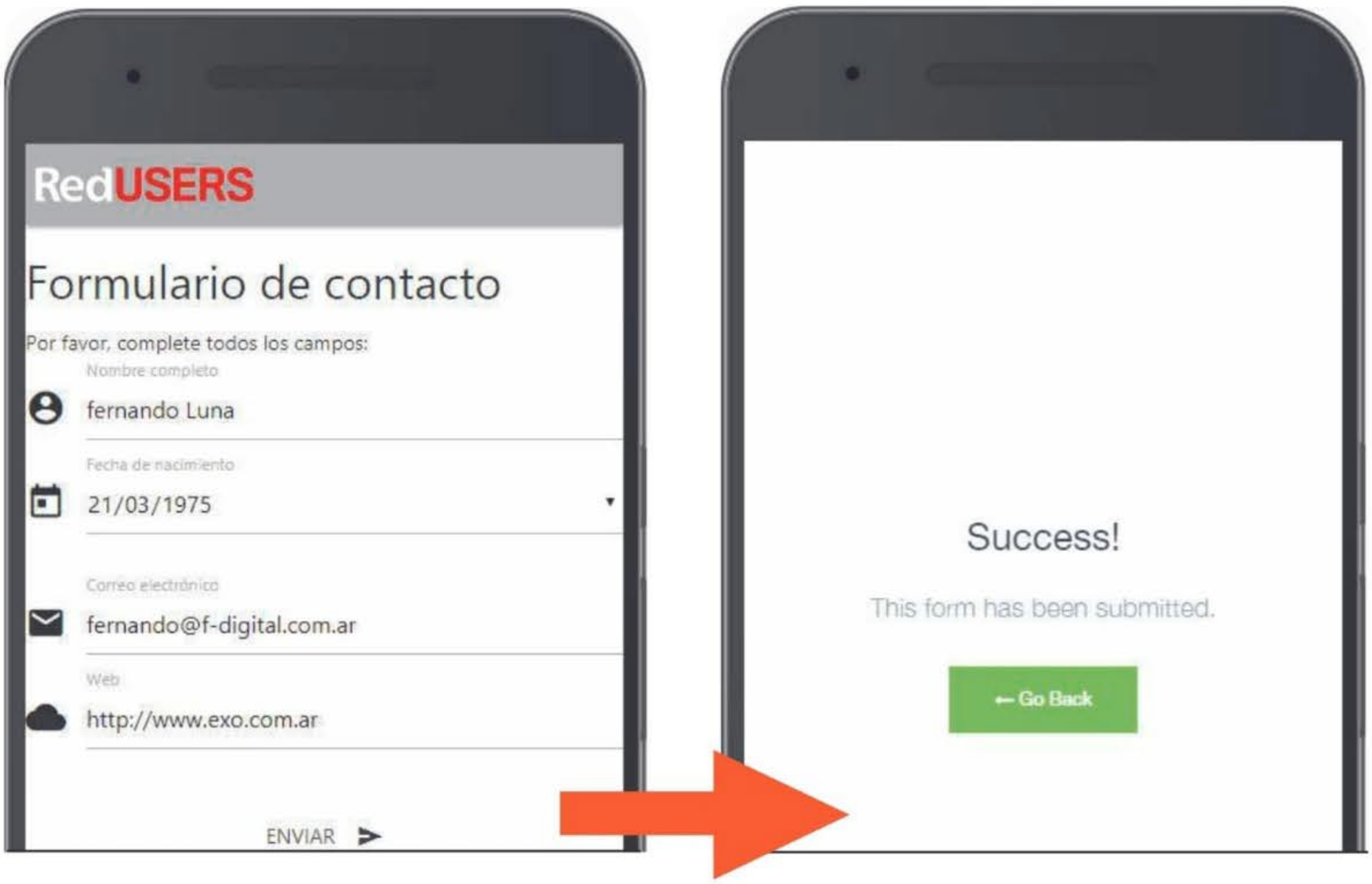
La forma de implementar este servicio es muy simple: en el atributo **action** del metatag **<form>** asignamos el siguiente valor:

```
https://mailthis.to/nombre@tuemail.com
```

Por supuesto que debemos reemplazar el mail descripto aquí por una dirección de correo electrónico válida. Luego, antes del cierre del metatag **<form>**, agregamos este otro atributo:

```
... method="POST">
```

Finalmente nos queda la prueba de fuego. Para probar el envío de los datos del formulario por correo electrónico, nos convendrá cargar el proyecto en un servidor web propio, o en una web que tengamos. Luego ejecutamos el formulario, completamos sus campos y presionamos el botón Enviar. Si todo va bien, el servicio **MailThis.to** nos mostrará un formulario de confirmación básico, con un botón que nos permitirá volver hacia atrás.



Ejercicio práctico: diseño de una web corporativa

Con los conocimientos adquiridos hasta aquí, estamos en condiciones de diseñar nuestra primera web institucional corporativa, con los elementos mínimos necesarios que esta requiere. Este diseño, por supuesto, deberá incluir y limitarse a los componentes HTML de Materialize CSS que hemos aprendido hasta ahora. Veamos a continuación las premisas a tener en cuenta:

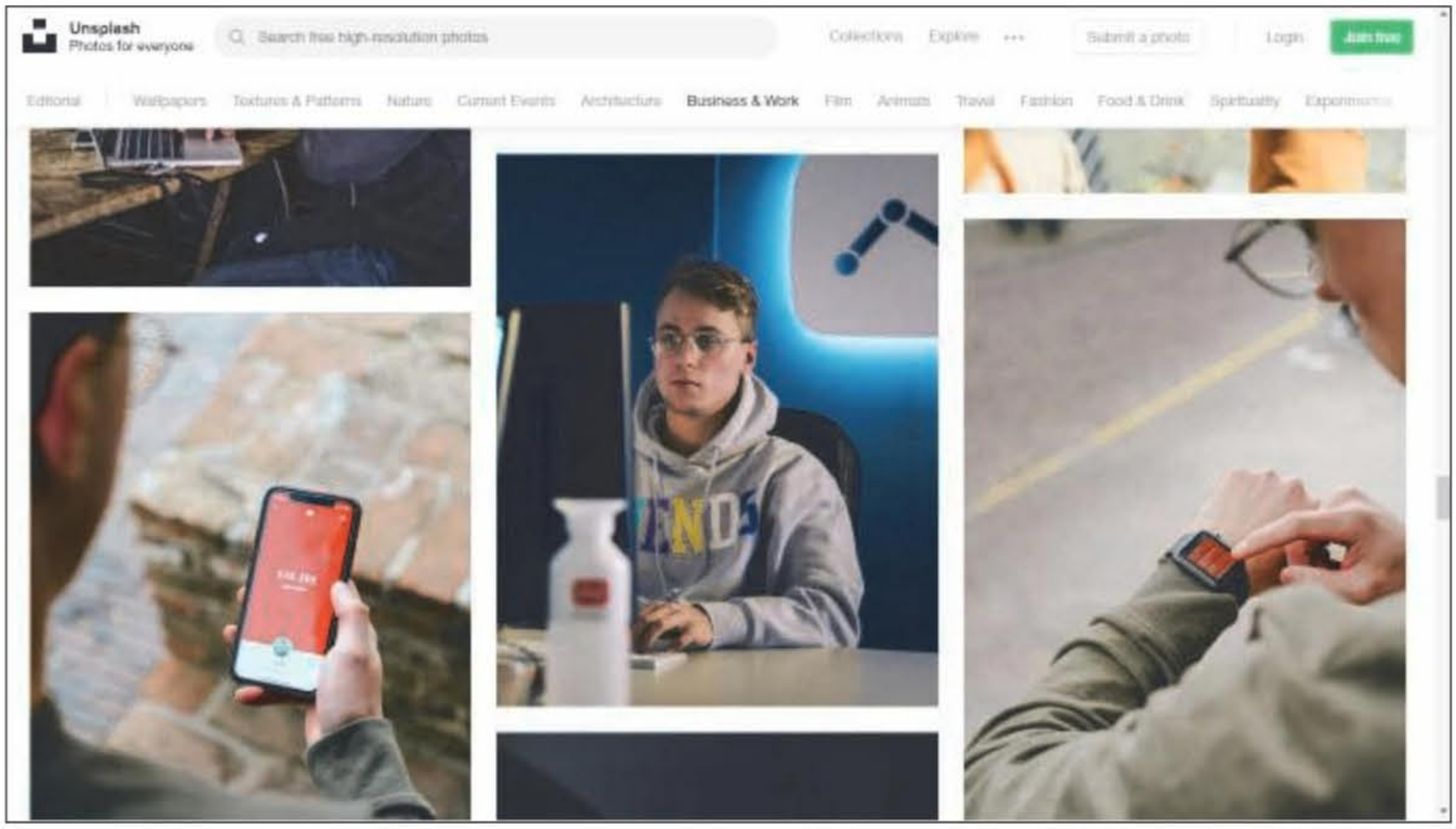
- *El sitio debe contener al menos tres secciones.*
- *Utilizar en todas ellas el elemento Navbar.*
- *Los hipervínculos a cada sección deben incluirse en el elemento Navbar.*
- *A su vez, los hipervínculos de Navbar tienen que ser diseñados como:*
  - *Iconos para móviles*
  - *Textos para tablets o desktop*
- *Usar títulos, blockquotes, párrafos e imágenes con* ***materialboxed****.*
- *Incluir formulario de contacto enlazado a* ***Mailthis*** *o similar.*

Para ver un ejemplo práctico de cómo debería lucir la web móvil que debemos realizar, es posible acceder al siguiente link: https://youtu.be/2IvrwFgV2JA. En este video, se observa de qué forma tiene que quedar estructurado el sitio web, teniendo en cuenta su barra de navegación superior (Navbar), la cual muestra en dispositivos móviles o con pantalla muy pequeña una serie de iconos que permitirán navegar en las distintas secciones.

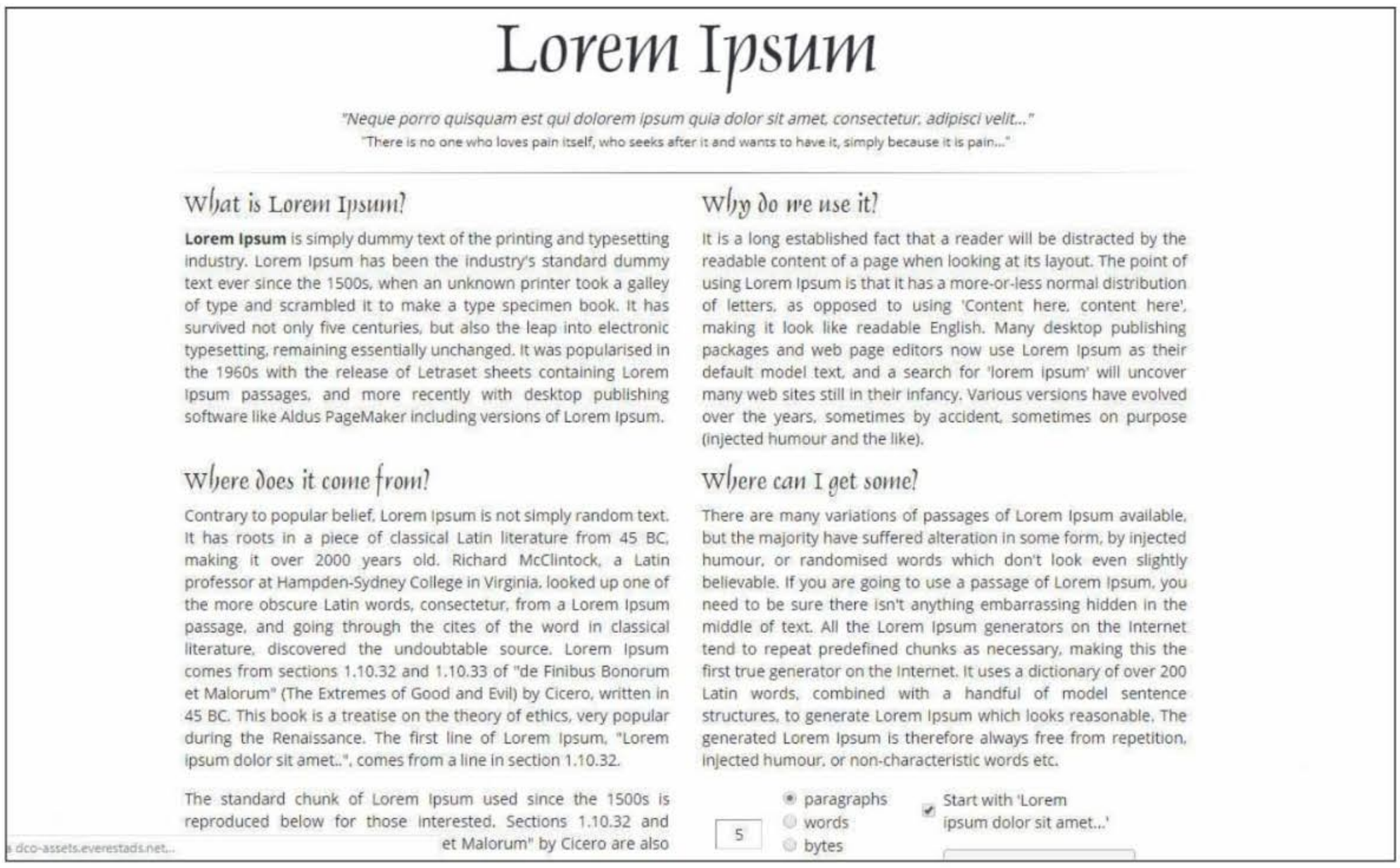
Estos iconos deben contener una imagen que permita a los usuarios ser intuitivos, e identificar sin ningún tipo de texto la sección a la que corresponde dicho hipervínculo.

Pautas complementarias para el diseño del ejercicio

Las imágenes utilizadas para complementar el contenido de esta web de ejemplo fueron extraídas del sitio web **Unsplah**: https://unsplash.com, que nos permite descargar y utilizar imágenes variadas, agrupadas por categorías. Si bien no exigen regalías ni necesidad de agregar copyright, en caso de que las utilicemos para un fin comercial, tengamos presente al menos indicar en alguna parte de nuestro sitio web quién es el autor del material.



La navegación vertical por la página debe tener uno o más títulos y, al menos, una imagen que represente el contenido de la web que queremos diseñar. En nuestro ejemplo utilizamos los párrafos de contenido **Lorem Ipsum** para armar el relleno de información, que se pueden extraer de www.lipsum.com.



Lorem Ipsum

"Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci velit..."

"There is no one who loves pain itself, who seeks after it and wants to have it, simply because it is pain..."

What is Lorem Ipsum?

Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Why do we use it?

It is a long established fact that a reader will be distracted by the readable content of a page when looking at its layout. The point of using Lorem Ipsum is that it has a more-or-less normal distribution of letters, as opposed to using 'Content here, content here', making it look like readable English. Many desktop publishing packages and web page editors now use Lorem Ipsum as their default model text, and a search for 'lorem ipsum' will uncover many web sites still in their infancy. Various versions have evolved over the years, sometimes by accident, sometimes on purpose (injected humour and the like).

Where does it come from?

Contrary to popular belief, Lorem Ipsum is not simply random text. It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old. Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia, looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage, and going through the cites of the word in classical literature, discovered the undoubtable source. Lorem Ipsum comes from sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" (The Extremes of Good and Evil) by Cicero, written in 45 BC. This book is a treatise on the theory of ethics, very popular during the Renaissance. The first line of Lorem Ipsum, "Lorem ipsum dolor sit amet..", comes from a line in section 1.10.32.

The standard chunk of Lorem Ipsum used since the 1500s is reproduced below for those interested. Sections 1.10.32 and 1.10.33 of "de Finibus Bonorum et Malorum" by Cicero are also

Where can I get some?

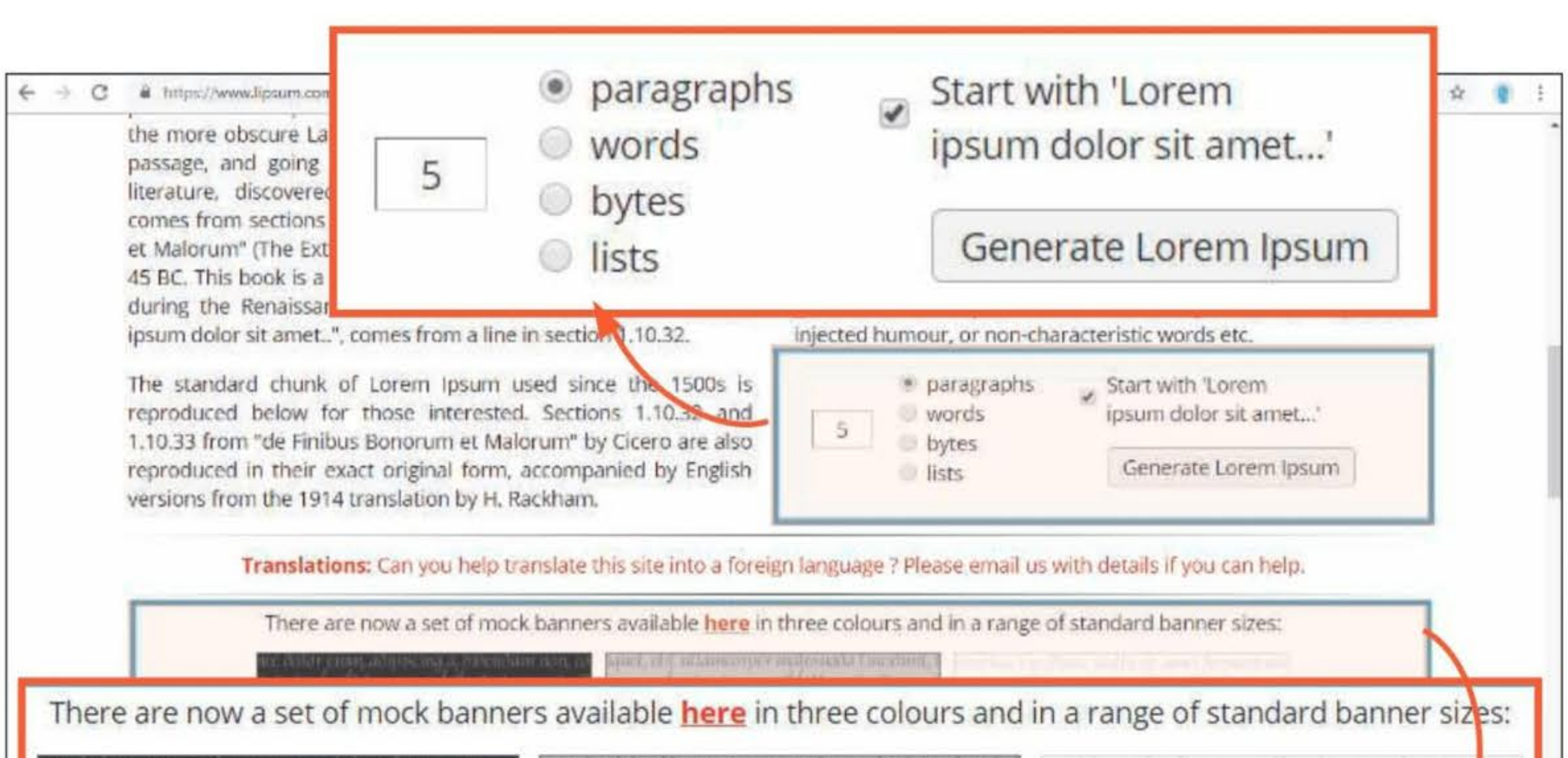
There are many variations of passages of Lorem Ipsum available, but the majority have suffered alteration in some form, by injected humour, or randomised words which don't look even slightly believable. If you are going to use a passage of Lorem Ipsum, you need to be sure there isn't anything embarrassing hidden in the middle of text. All the Lorem Ipsum generators on the Internet tend to repeat predefined chunks as necessary, making this the first true generator on the Internet. It uses a dictionary of over 200 Latin words, combined with a handful of model sentence structures, to generate Lorem Ipsum which looks reasonable. The generated Lorem Ipsum is therefore always free from repetition, injected humour, or non-characteristic words etc.

5 paragraphs words bytes

Start with 'Lorem ipsum dolor sit amet...'

dco-assets.everestads.net...

Al acceder a esta web, encontraremos una explicación general sobre cómo nació Lorem Ipsum, y de qué manera se puede aplicar en el diseño o desarrollo de soluciones de software en general, que requieran inicialmente contar con párrafos de relleno. Hacia el centro de la hoja, hay una sección que permite generar automáticamente un párrafo con determinada extensión.



Y en el encabezado de la página, figura la opción para traducir la explicación general de este mecanismo de generación automática de contenido, hacia el idioma español o cualquier otro que deseemos.

Tengamos presente que el contenido que se genera en Lorem Ipsum siempre estará escrito en latín y no, en español o cualquier otro idioma, porque es solo a los efectos informáticos.

Prueba del sitio web en dispositivos móviles

Ya sea cuando finalicemos el desarrollo del sitio o al crearlo, tengamos presente las formas de realizar el testing correspondiente desde un dispositivo móvil. Las posibilidades existentes para recurrir a un emulador móvil son:

Android ADB

Google pone a nuestra disposición **ADB (Android Debug Bridge)**, que nos permite ejecutar una serie de herramientas en la computadora para poder conectar un equipo móvil físico a la PC mediante un cable USB, y ejecutar cualquier web o aplicación móvil directamente en la pantalla de este. La información para descargar y obtener esta plataforma se encuentra en: https://developer.android.com/studio/command-line/adb?hl=es-419.

Emulador iOS

Si contamos con un equipo Apple Mac, podemos descargar e instalar XCode, el cual trae asociados los emuladores iOS. De esta forma, podremos testear cualquier desarrollo web en iOS: https://developer.apple.com/xcode. Tengamos presente que esto solo es funcional en equipos Apple; no está disponible para Windows o Linux.

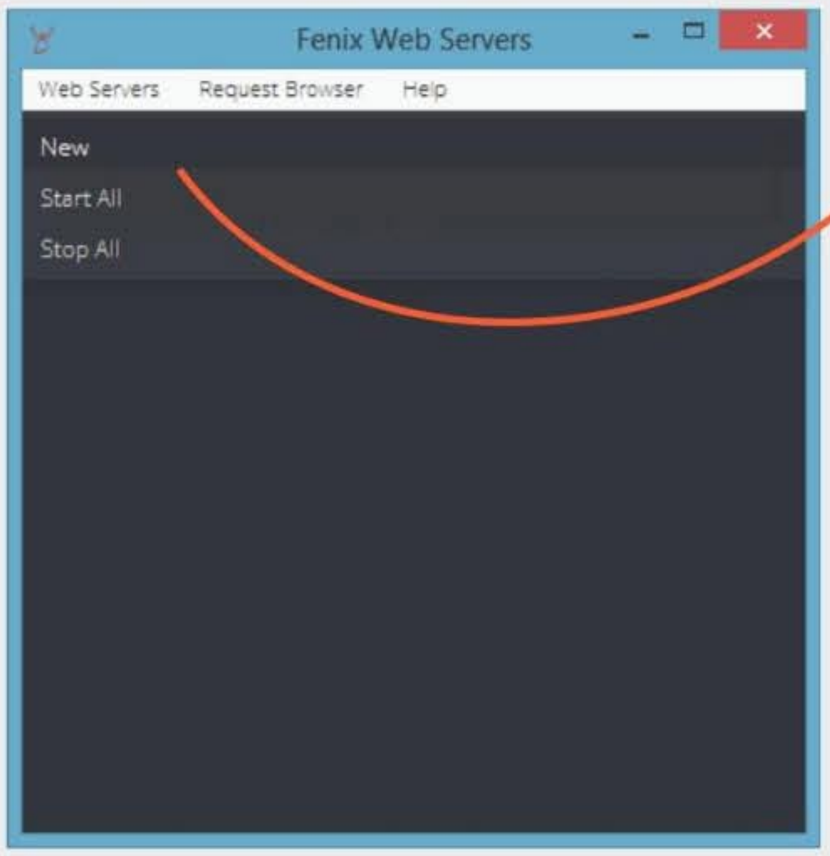
Bluestacks

La plataforma **Bluestacks** ofrece un emulador de Android versátil y fácil de instalar. Es otra alternativa válida y útil para quienes quieren practicidad al instalar y configurar un emulador Android: www.bluestacks.com/es/index.html.

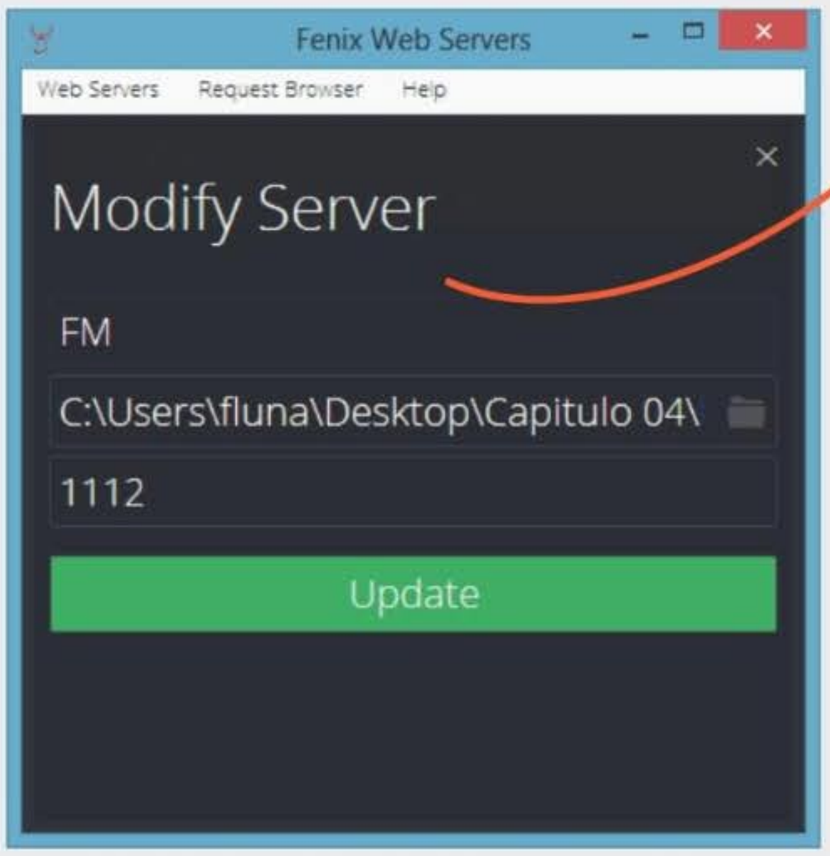
Fenix Web Server

Desde la web oficial, www.fenixwebserver.com, podemos descargar e instalar este servidor web, que nos permitirá, dentro de nuestra red WiFi doméstica, levantar un servidor web y que este sea accesible desde cualquier dispositivo conectado a esa misma red. Su instalación es simple e intuitiva, al igual que la configuración de un sitio web.

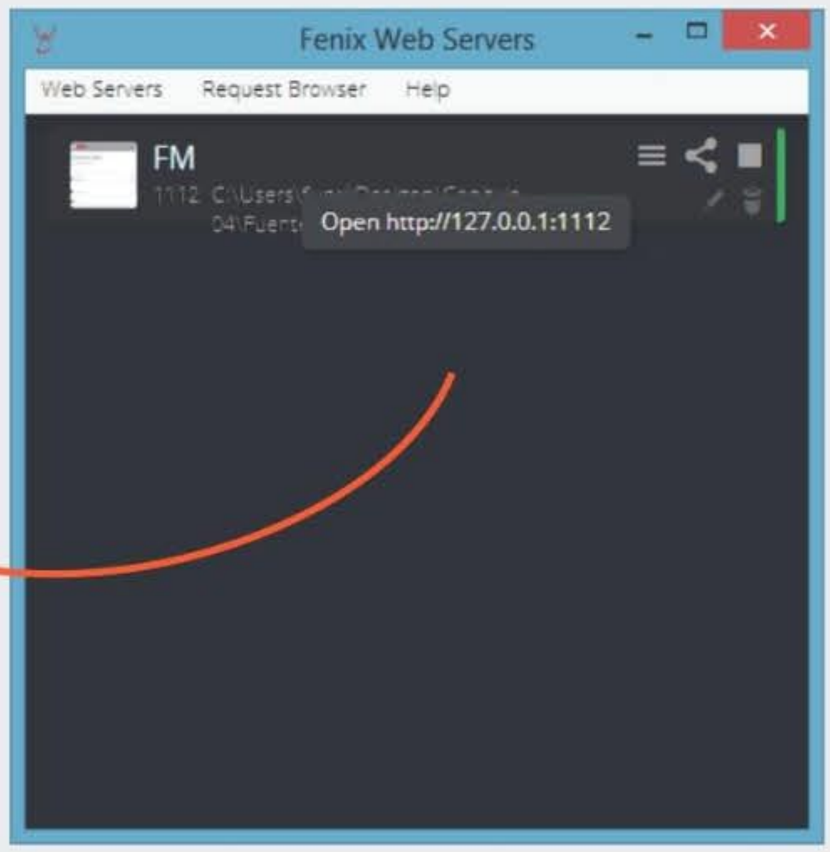
GUÍA VISUAL 2



*Una vez instalada y ejecutada la aplicación **Fenix Web Server**, desplegamos el menú **Web Servers** y seleccionamos la opción **New**.*



*En la ventana emergente cargamos un **nombre identificativo** para el servidor web, **elegimos la carpeta de archivos** de nuestro proyecto, y **seleccionamos un número de puerto** o dejamos por defecto el ofrecido.*



*Luego de **Guardar** en el paso anterior, solo nos resta poner **Play** en el botón correspondiente a nuestro web server creado. Posicionando el mouse sobre él, veremos un pop-up con la URL y el puerto que debemos ingresar desde el navegador web o dispositivo móvil (en la misma red WiFi) para poder testear nuestro desarrollo.*

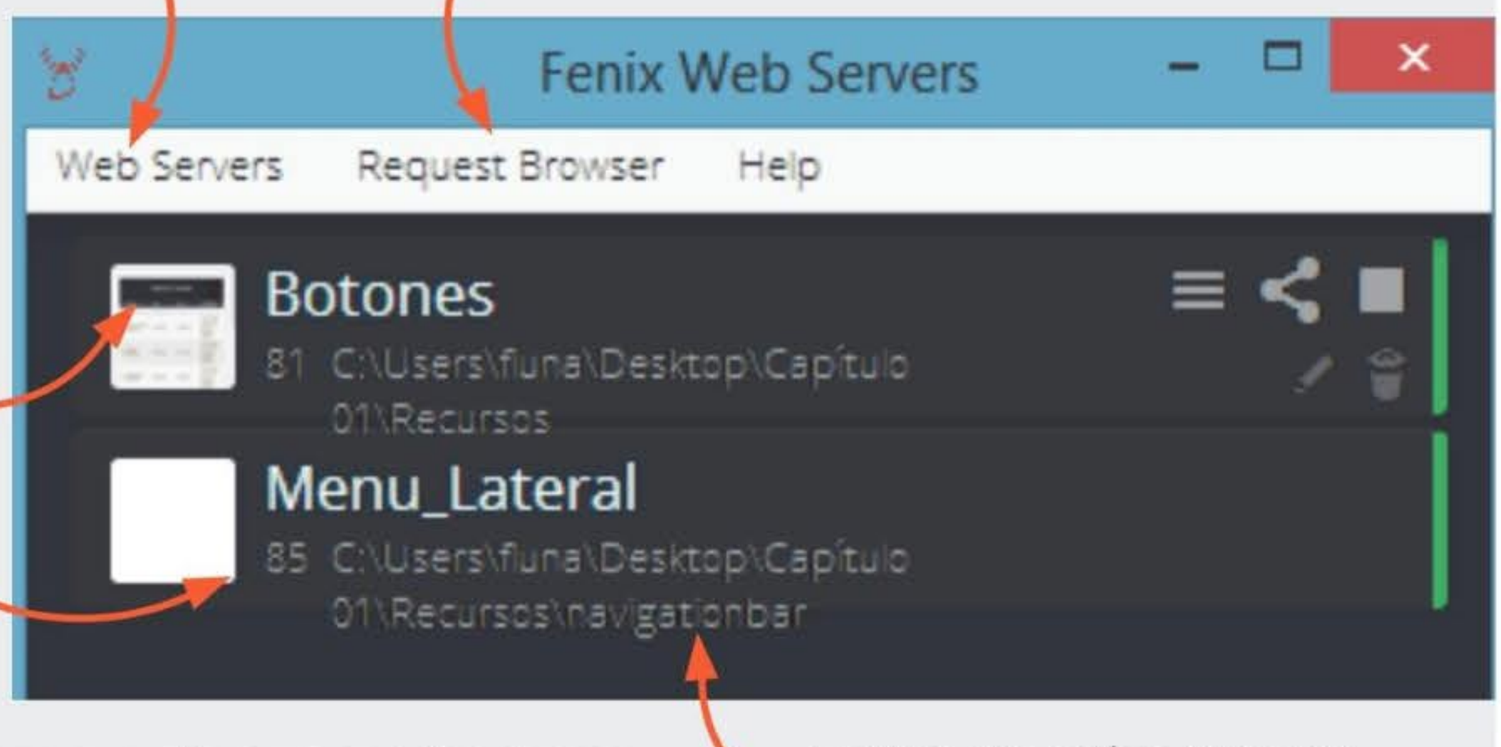
Fenix Web Server se encuentra en constante desarrollo, después de "cual" accediendo de manera frecuente a su sitio web, podremos ver las distintas características que este servidor suma en sus nuevas versiones.

GUÍA VISUAL 3

Aquí visualizamos el título indicado para un servidor web creado previamente.

Request Browser permite ejecutar el servidor web en modo público, y acceder a este de manera global. Esto genera un túnel con algún servicio web propio de los creadores de Fenix WS, pero no es necesario utilizarlo si las pruebas serán internas.

Al ejecutarlo por primera vez, el sistema toma una captura de pantalla de la página principal.

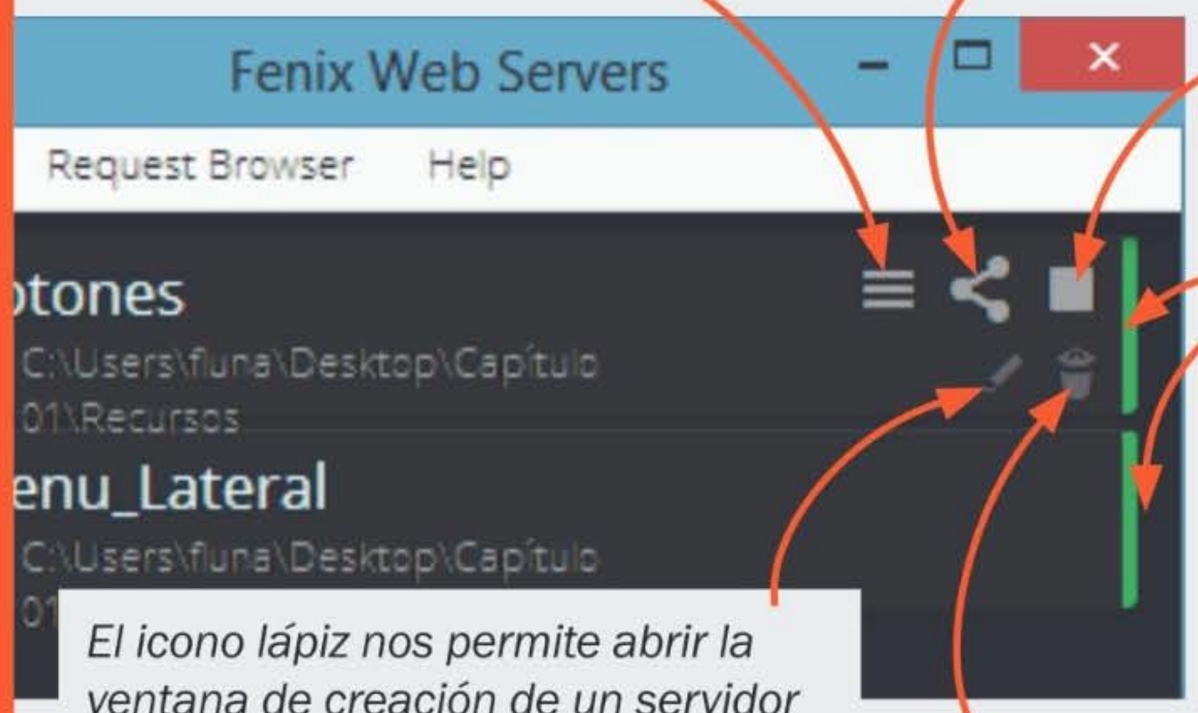


El número que allí vemos es el puerto que indicamos en la creación del mismo. Recordemos que siempre que tengamos muchos webservers, debemos crearlos con diferentes números de puertos para no causar conflictos y poder ver todos a la vez.

La ruta que allí se muestra corresponde a la carpeta del proyecto web.

Permite acceder al LOG de procesos y ejecuciones del servidor web en cuestión.

Permite compartir el servidor web en la red donde se encuentra nuestra computadora.



Ejecuta (play) o detiene (stop) el servidor web en cuestión.

El color verde indica que el servidor web está en ejecución; en color anaranjado, que está pausado; y el color rojo, que se encuentra detenido.

El icono lápiz nos permite abrir la ventana de creación de un servidor web en modo edición, para cambiar alguno de los parámetros ingresados.

Nos permite eliminar el servidor web creado, de la lista de servidores de esta aplicación.

05 Audio y Video

Desde 2005, con la popularización de YouTube y otros servicios de streaming en video y/o audio, todo contenido multimedia cobró gran importancia en el mundo web y mobile. Veamos qué nos ofrece este medio para manipular este tipo de archivos.

Este capítulo lo dedicamos enteramente a los archivos multimedia. Veremos a continuación cómo manipular archivos de audio desde HTML5, de manera independiente del framework Materialize CSS que venimos utilizando hasta aquí. Recurriremos no solo a HTML5 para el desarrollo de los ejercicios; JavaScript será nuestra contraparte importante para manipular con precisión el contenido de audio que deseemos incluir en nuestro trabajo.

El elemento <audio>

El elemento HTML **<audio>** nació junto con esta variante del lenguaje de marcado para poder, de forma ágil y práctica, manipular estos archivos multimedia en el universo web. Este elemento contiene una serie de atributos que nos permiten lograr una mayor profundidad cuando debemos referenciar un archivo del tipo MP3 o similar en un documento.

Sintaxis

La sintaxis básica de HTML5 para incluir el componente web de audio es la siguiente:

```
<audio id="reproductor"></audio>
```

Esta sintaxis nos permite visualizar un minirreproductor de audio y sus controles básicos (**Play**, **Tiempo**, **Longitud** de la pista y **Volumen**). Para agregar el contenido que deseamos reproducir al elemento **audio**, hay dos opciones:

Opción 1. Añadir el atributo **src** dentro del tag **audio**, junto con la ruta del archivo de audio por reproducir.
Opción 2. Incluir varios elementos **<source>** internos, e incorporar en ellos el atributo **src** mencionado anteriormente.

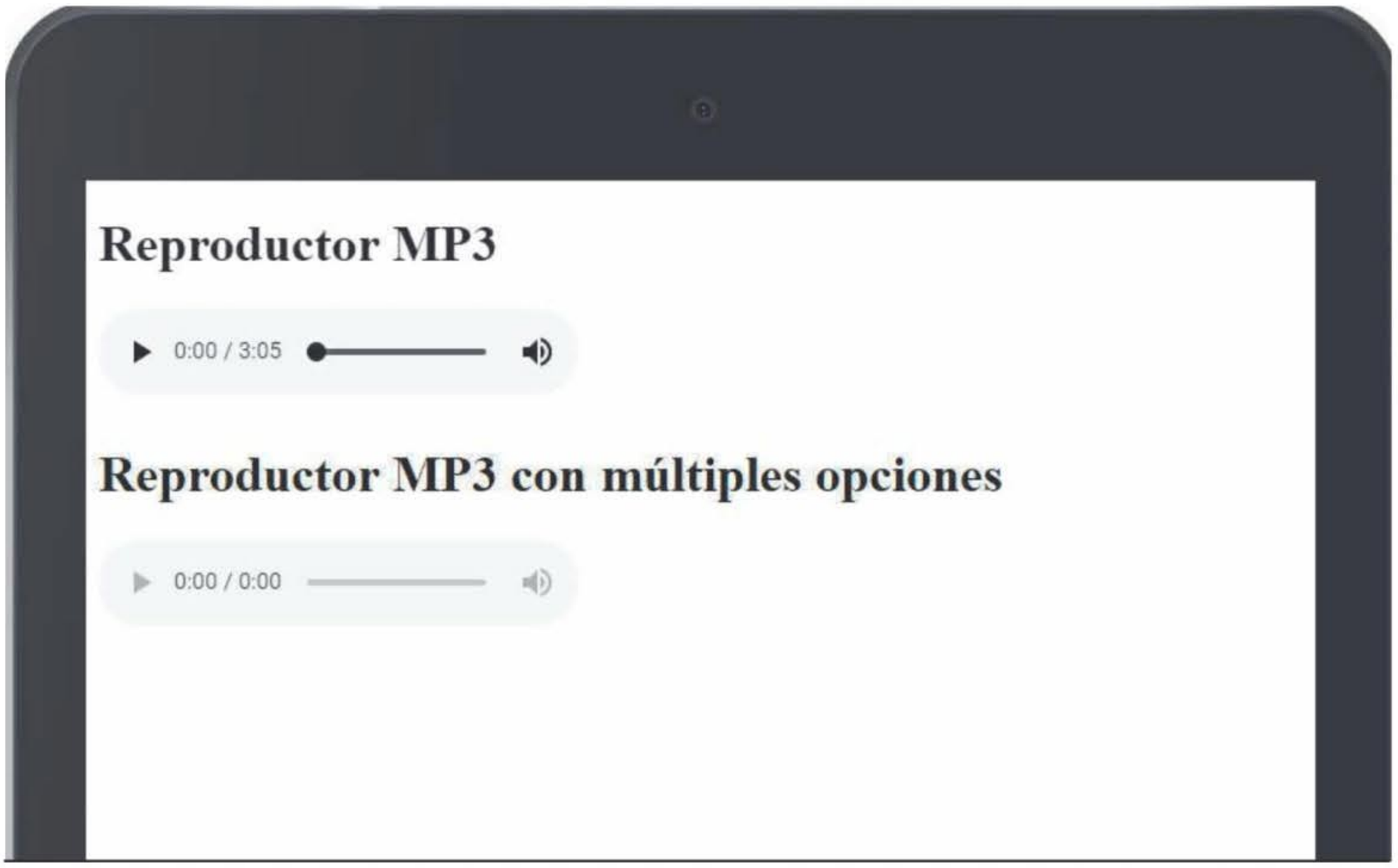
Opción 1

```
<audio controls
        src="cantando_bajo_la_ducha.mp3"
        type="audio/mpeg">
</audio>
```

Opción 2

```
<audio controls>
  <source src="cantando_bajo_la_ducha.ogg" type="audio/ogg">
  <source src="cantando_bajo_la_ducha.mp3" type="audio/mpeg">
  Su navegador no soporta el tag AUDIO.
</audio>
```

Veamos en la siguiente imagen el resultado de lo que vimos en las opciones anteriores.



Propiedades del elemento audio

Este elemento contiene una serie de **propiedades** y/o **atributos** que nos permiten manipularlo de manera más efectiva, tanto desde HTML5 como desde JavaScript. Veamos la siguiente tabla para conocer los principales:

TABLA 1

PROPIEDAD	DESCRIPCIÓN
autoplay	Permite iniciar una reproducción automática del audio, apenas este ha sido cargado o generado un nivel de búfer local óptimo.
buffered	Devuelve un rango de tiempo representando la parte del audio almacenada en el búfer.
controls	Muestra u oculta los controles multimedia básicos de HTML5.
currentSrc	Retorna la URL completa del archivo de audio cargado en el reproductor.
currentTime	Devuelve o mueve la posición de reproducción actual del archivo de audio.
duration	Devuelve el total de tiempo (en segundos) que dura el archivo de audio.
ended	Devuelve True cuando la reproducción del archivo de audio ha finalizado.
error	Devuelve el tipo de error producido ante la carga fallida de un archivo de audio.
loop	Permite configurar la reproducción continua de un archivo de audio.
muted	Permite configurar o saber si el audio del archivo se encuentra silenciado.
playbackRate	Devuelve o configura la velocidad de reproducción del archivo de audio (pitch).
played	Devuelve el rango de tiempo del archivo de audio que ya ha sido reproducido.
src	Devuelve o configura el valor del atributo src de un archivo de audio.
volume	Devuelve o configura el nivel de volumen del archivo de audio.

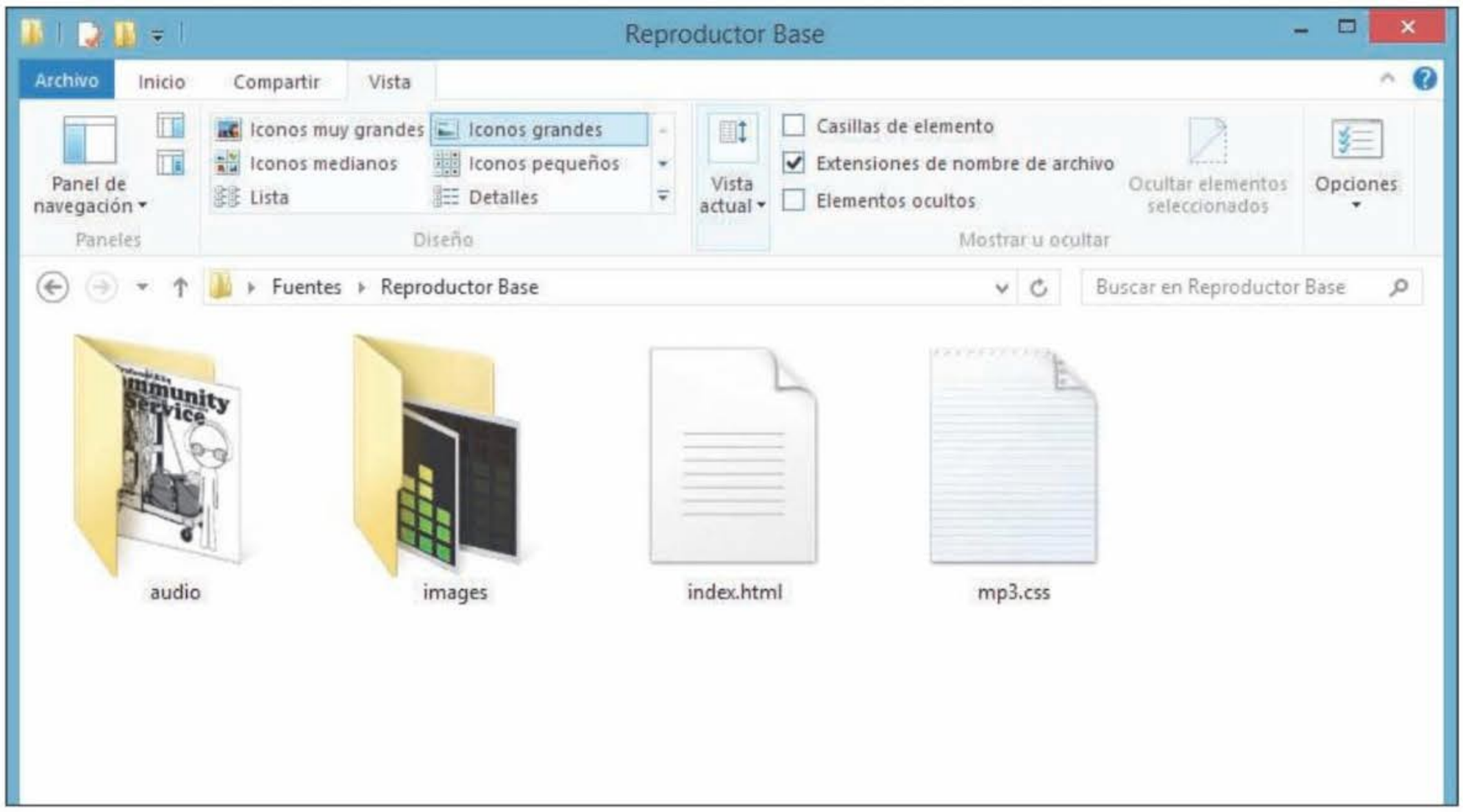
Descarguemos del repositorio de archivos de esta obra el proyecto **Audio-Ejemplo.zip**. Vamos a descomprimirlo y a ejecutar el archivo **index.html** en el navegador web o en nuestro dispositivo móvil mediante **Fenix Web Server**, creando un nuevo proyecto tal como explicamos en el capítulo anterior.

Ya podemos utilizar el componente multimedia Audio, en el documento HTML que acompaña a dicho ejemplo. En él vemos cómo reproducir un archivo MP3 estándar.

Mediante la segunda opción que representamos podremos cargar más de un archivo de audio para que sea el navegador web de la plataforma móvil, o de escritorio, el que elija el formato de archivo más adecuado a su capacidad de reproducción.

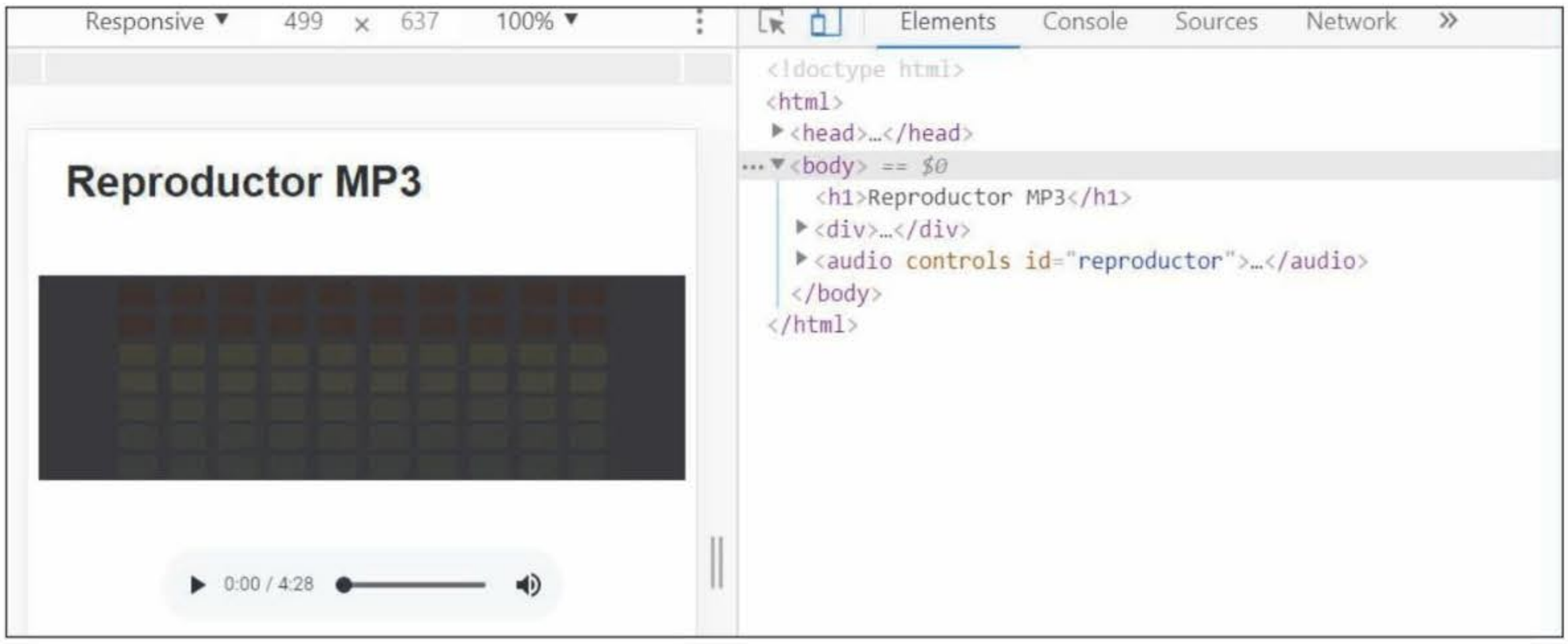
Controlar el audio desde JavaScript

A continuación, veamos cómo JavaScript puede complementarse fácilmente con nuestros proyectos, para permitirnos, en este caso, manipular desde el código de scripting los archivos de audio de nuestro desarrollo en curso. Para esto, vamos a descargar el proyecto **Audio-Ejemplo.zip** del repositorio de archivos de esta obra. Lo descomprimimos y pasamos a analizar qué contiene:



- ***Index.html:*** *es el archivo en el cual trabajaremos el manejo de reproducción de audio.*
- ***mp3.css:*** *nos permite agregar algo de contenido CSS, para mejorar la estética de nuestro reproductor MP3. Obviaremos en esta etapa el uso de Materialize CSS, para poder enfocarnos en aprender lo más importante del tag* **<audio>**.
- ***Subcarpeta audio:*** *contiene un archivo MP3 con licencia Creative Commons, que utilizaremos para desarrollar este ejercicio.*
- ***Subcarpeta images:*** *contiene dos imágenes;* ***bar-off.gif*** *corresponde a un archivo de imagen estándar que emula un vúmetro apagado, y* ***bar-on.gif*** *es un GIF animado, que utilizaremos a posteriori en este mismo ejercicio.*

Abrimos Visual Studio Code y cargamos la carpeta de este proyecto para comenzar a trabajar. Luego abrimos Google Chrome y cargamos el documento **index**.html. Mediante la tecla **F12** abrimos las **Herramientas del desarrollador** de Google Chrome; el navegador web se verá tal como muestra la siguiente imagen:

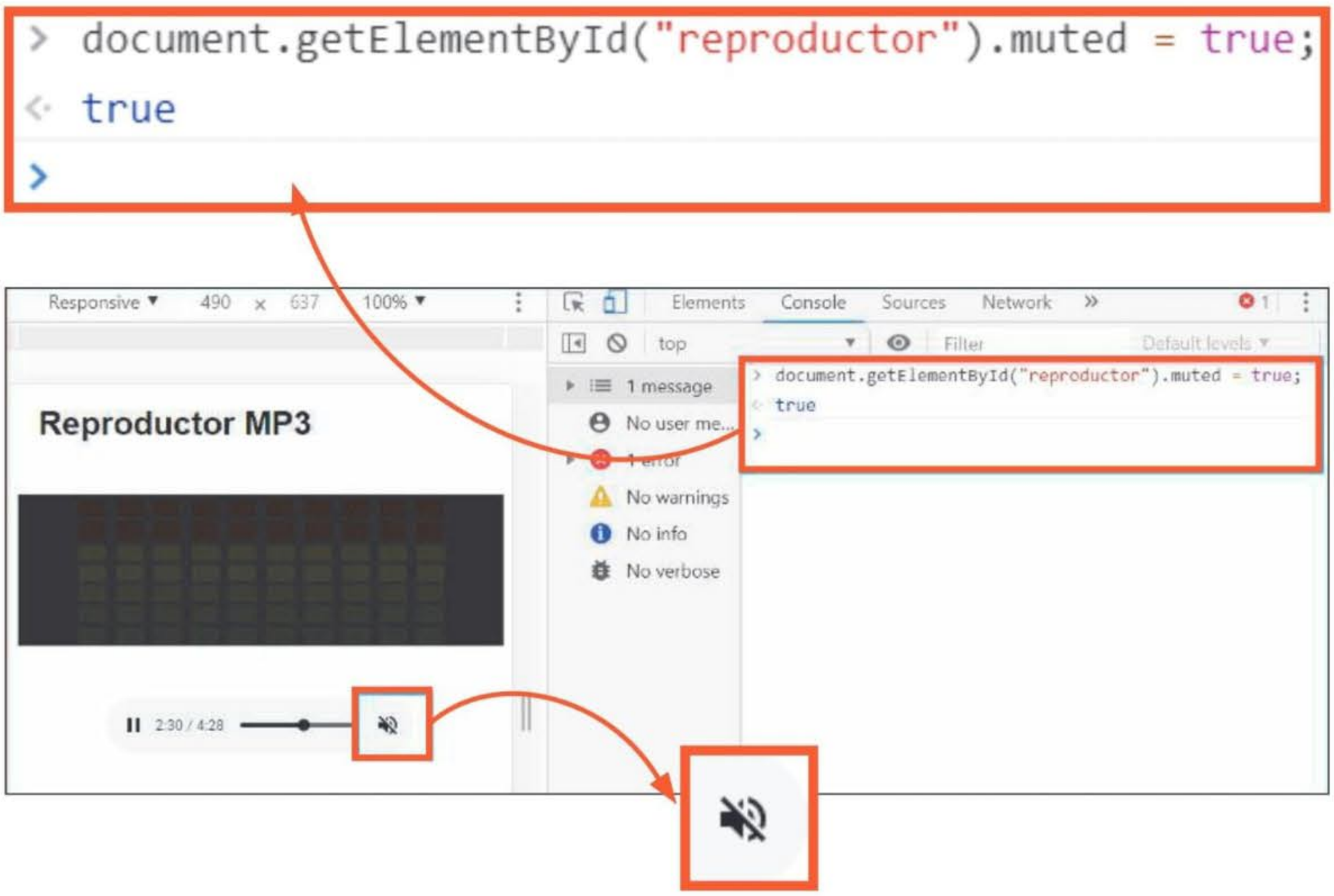


Parados sobre la ventana Herramientas del desarrollador, pulsamos en la opción **Console**, ubicada en el extremo superior central-derecho. Damos Play para comenzar a escuchar el archivo de audio y, mientras este se reproduce, escribimos en la línea de comandos de esta consola lo siguiente:

```
document.getElementById("reproductor").muted;
```

Si estamos escuchando el audio, la propiedad **muted** nos devuelve **false** como resultado en la consola del navegador. Ahora escribimos en la misma consola la siguiente línea:

```
document.getElementById("reproductor").muted = true;
```



El audio se silenciará en el momento en que ejecutamos este último comando. Vemos también que el icono **volumen** de dicho elemento HTML cambiará su gráfico por **mute**. Desbloqueamos el audio repitiendo este último comando con el agregado del parámetro **= false**, y lo ejecutamos:

```
document.getElementById("reproductor").loop = true;
```

Al finalizar el archivo de audio, este volverá a comenzar otra vez.

Entender JavaScript

Como podemos ver, JavaScript tiene total injerencia sobre este y cualquier otro componente HTML. Analicemos a continuación el código JS escrito para comprender de qué manera este lenguaje de scripting controla los componentes HTML.

GUÍA VISUAL 1

*Selecciona o hace referencia al documento o archivo HTML cargado en el navegador; **index.html** en este caso.*

*Seleccionamos, mediante **getElementById**, el componente HTML cuyo **ID** es "reproductor", indicado entre paréntesis.*

*Utilizaremos, de este componente HTML seleccionado, su propiedad **muted**.*

```
document.getElementById("reproductor").muted = true;
```

Para asignar un valor determinado a dicha propiedad, empleamos el signo igual, seguido del valor. En este caso, solo acepta valores booleanos (true o false)

index.html ×

```
<!DOCTYPE html>
<html>
    <head>
        <title>Reproducción MP3</title>
        <meta charset="utf-8">
        <script defer src="mp3.js"></script>
        <meta name="viewport" content="width=device-width, initial-scale=1.0, maximum-scale=1
        <link rel="stylesheet" type="text/css" href="mp3.css">
    </head>
    <body>
        <h1>Reproductor MP3</h1>
        <div>
            <img src="images/bar-off.gif" id="vmeter" width="80%" />
        </div>
            <audio controls id="reproductor">
                <source src="audio/Professor_Kliq_-_03_-_Bust_This_Bust_That.mp3" type="audio
            </audio>
    </body>
</html>
```

Para obtener un conocimiento más profundo sobre el lenguaje JavaScript, es recomendable recurrir a la **Guía Users N°5 – JavaScript**, donde se brinda una completa referencia a la estructura y el uso de este fascinante e indispensable lenguaje de programación.

Cómo animar el vúmetro

A continuación, pondremos en acción el vúmetro de nuestra aplicación web, el cual responderá con una animación virtualizada por el archivo **bar-on.gif**, solo en el momento en el que se reproduce la música. Dentro del archivo index.html, ubicamos la referencia al archivo **mp3.js**. Lo creamos pulsando **Ctrl + clic** sobre el archivo en sí. VS Code nos advertirá que no existe y nos ofrecerá generarlo, para lo cual presionamos el botón correspondiente. A continuación, agregamos el siguiente código en dicho archivo:

```
document.addEventListener("DOMContentLoaded", function() {
    var rep = document.getElementById("reproductor");
    var i = document.getElementById("vmeter");

})
```

Este código invoca a la función **adEventListener**, dentro de la cual hacemos referencia al evento **DOMContentLoaded**. La función en sí se ocupa de establecer un mecanismo de escucha, referenciado al documento HTML en cuestión. DOMContentLoaded es el evento que ocurre cuando la página web se ha cargado por completo. Cuando adEventListener "escucha" que ocurre dicho evento, recién entonces ejecuta el código interno de la función.

Dentro de dicha función estamos declarando dos variables: **rep** e **i**. La primera almacenará una referencia directa al componente HTML reproductor, mientras que la segunda referencia al componente HTML **vmeter**.

Función de las variables declaradas

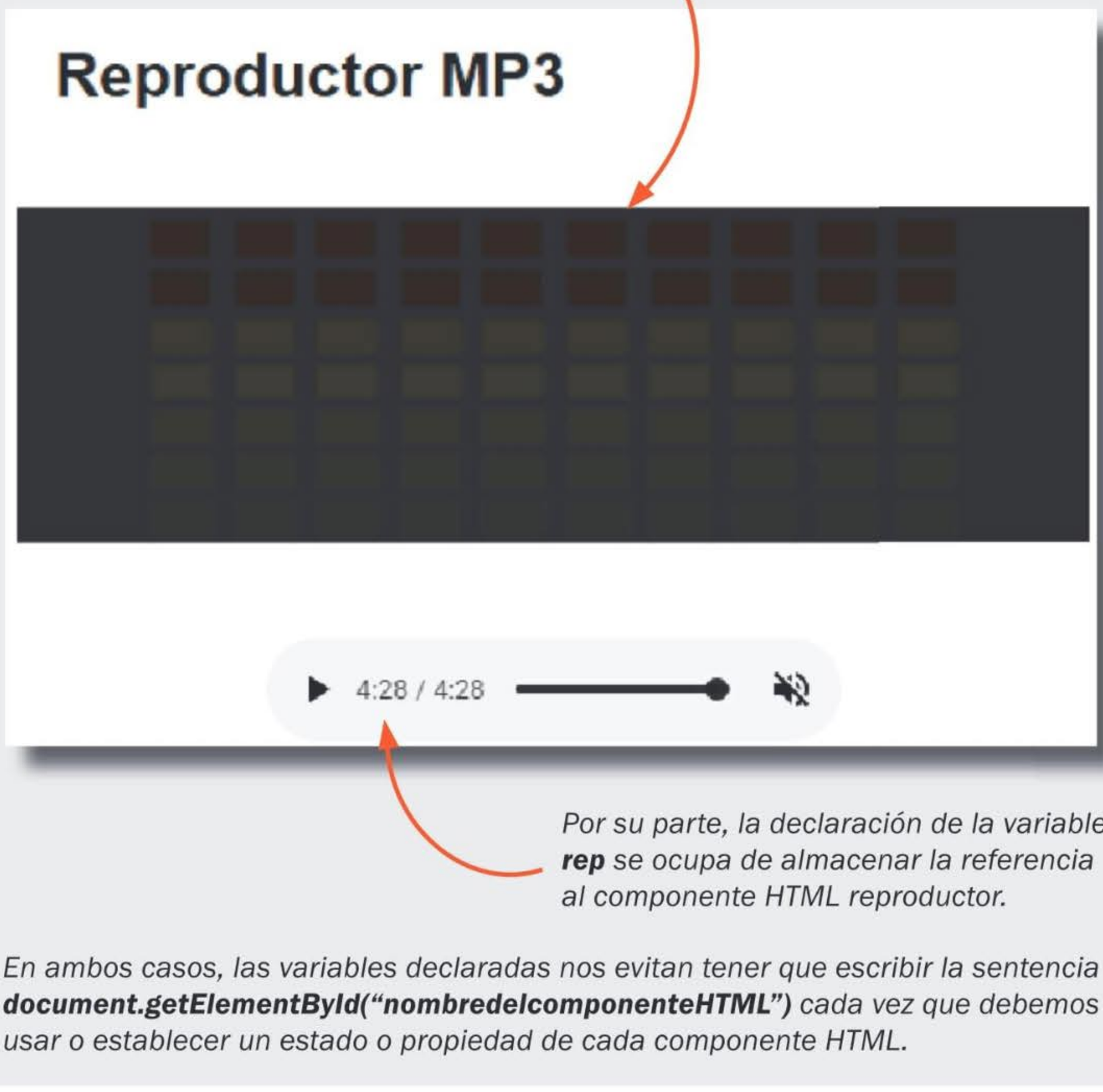
Con las variables **rep** e **i** declaradas, nos ocuparemos de detectar el cambio de estado en el evento de reproducción de audio. Para esto, a continuación de la declaración de dichas variables, agregamos el siguiente código JS:

```
rep.addEventListener("ended", function() {
    i.setAttribute("src", "images/bar-off.gif");

})
```

En este nuevo **adEventListener**, reemplazamos el objeto **document** por el objeto **rep**. Esto nos permite escuchar los eventos puntuales que genera el componente **<audio>**. El evento específico que debemos escuchar aquí es **ended**, que se produce cuando un archivo de audio finaliza su reproducción.

GUÍA VISUAL 2

*Si miramos el código de **index.html**, encontraremos un elemento **<img>** cuyo atributo **ID** es **vmeter**. Esto nos permitirá, mediante la variable **i** declarada en JavaScript, manipular la imagen de dicho elemento HTML.*



*Por su parte, la declaración de la variable **rep** se ocupa de almacenar la referencia al componente HTML reproductor.*

*En ambos casos, las variables declaradas nos evitan tener que escribir la sentencia **document.getElementById("nombredelcomponenteHTML")** cada vez que debemos usar o establecer un estado o propiedad de cada componente HTML.*

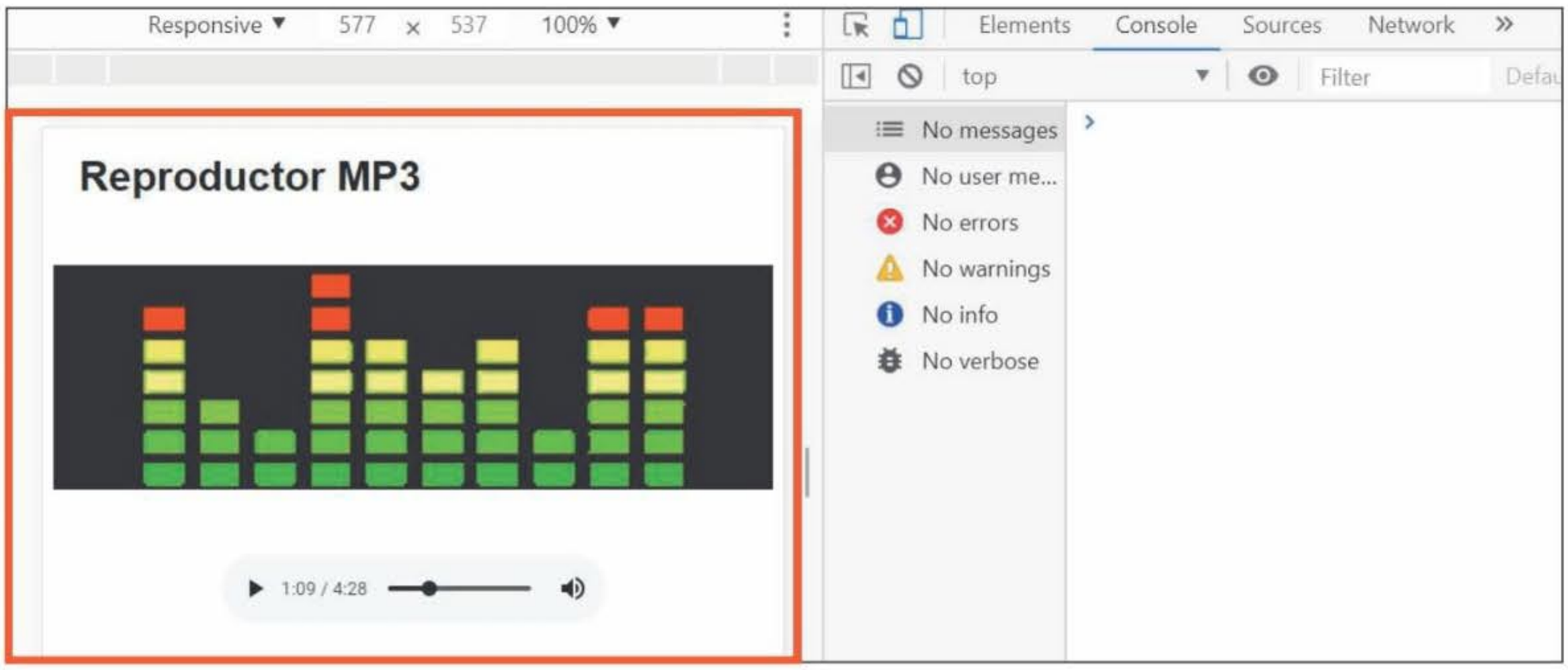
Manejo de CSS desde JS

JavaScript nos permite manipular los estilos **css** de sus componentes HTML. En el bloque **adEventListener** siguiente, vemos la línea de código **i.setAttribute(...)**. Lo que hacemos en ella es definir para el atributo **src** de la variable **i** (correspondiente a la imagen GIF), que visualice en el documento HTML la imagen **bar-off.gif**. Todo esto ocurre cuando la reproducción de un archivo de audio llega

a su fin. Ahora, haremos lo mismo pero cambiando la imagen por **bar-on.gif** cuando el archivo de audio comienza a reproducirse. Para esto, escucharemos el evento **play** del componente HTML audio.

```
rep.addEventListener("play", function() {
    i.setAttribute("src", "images/bar-on.gif");

})
```

Guardamos el contenido del archivo JS y del resto de los archivos del proyecto actual modificados, y refrescamos la página HTML cargada en Google Chrome. Si todo va bien, podremos probar la funcionalidad que añadimos hasta aquí, pulsando Play para reproducir el archivo de audio. El resultado en pantalla mientras se reproduce el archivo de audio, debe ser como la siguiente imagen:



Cuando termine y la reproducción se detenga (ejecutándose el evento **ended**), la imagen en pantalla **bar-on.gif** debe cambiar por **bar-off.gif**.

Pausar la reproducción

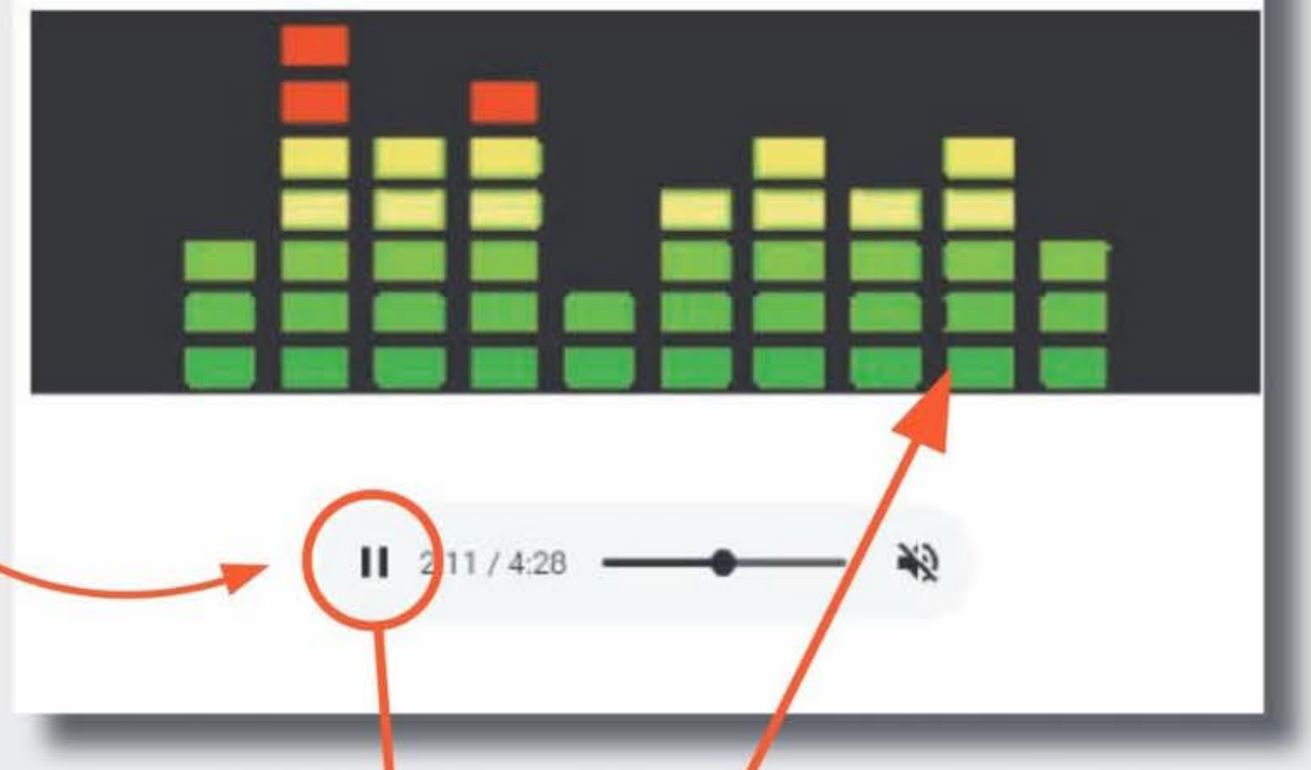
Ahora nos queda agregar la escucha del evento **pause**, para cuando ejecutemos esta acción sobre el archivo en reproducción:

```
rep.addEventListener("pause", function() {
    i.setAttribute("src", "images/bar-off.gif");

})
```

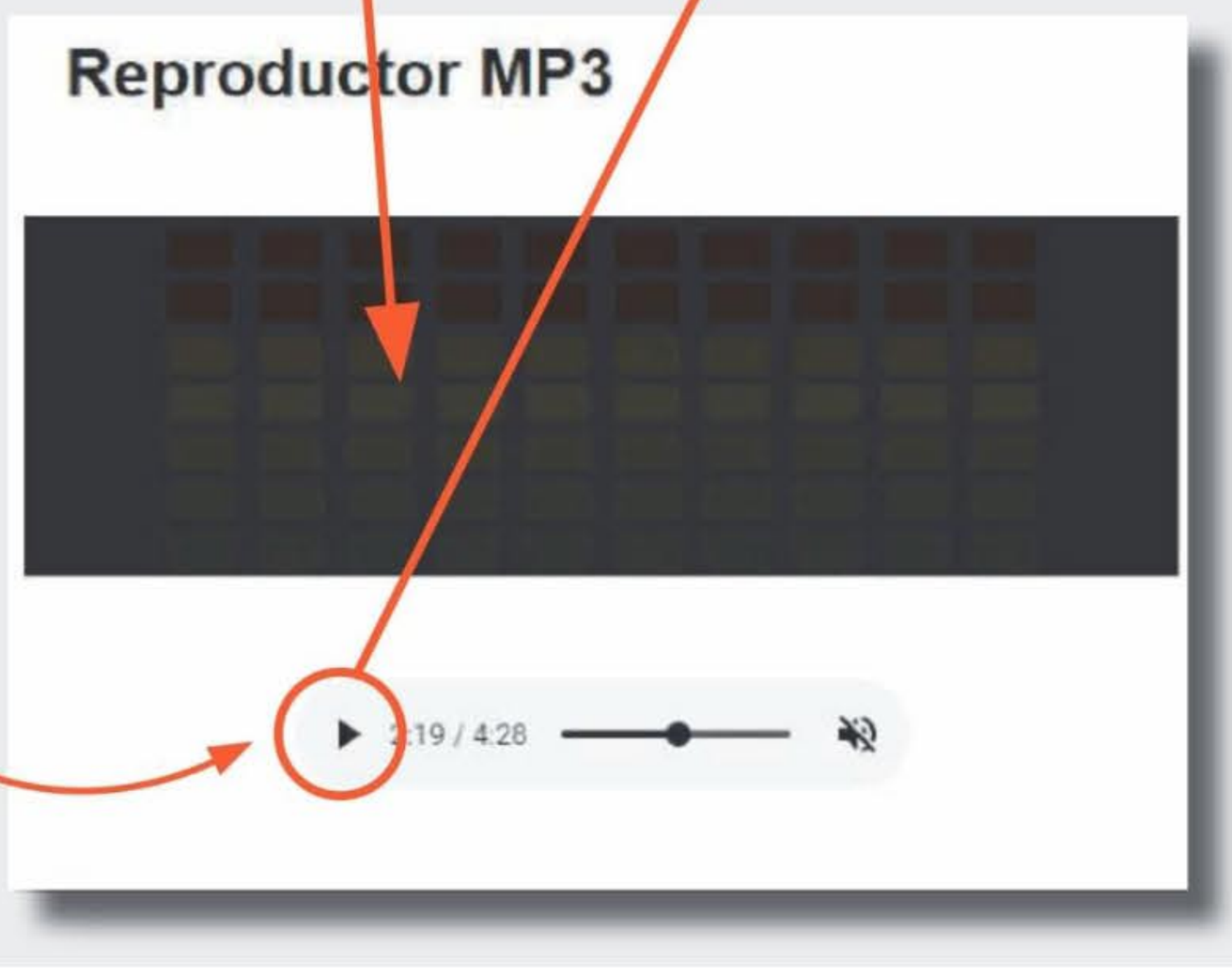
Su estructura es igual a la declarada en “ended”, pero aquí hacemos que el objeto **rep** escuche el evento **pause**, en vez del mencionado anteriormente.

GUÍA VISUAL 3

Al presionar el botón **pausa** del reproductor MP3, se ejecuta el evento “**pause**”, **adEventListener** lo detecta y cambia el atributo **src** de la imagen, por **bar-off.gif**.



Al pulsar el botón **play** del reproductor MP3, se ejecuta el evento homónimo: **adEventListener** lo detecta y cambia el atributo **src** de la imagen, por **bar-on.gif**.



Visualización según el navegador web

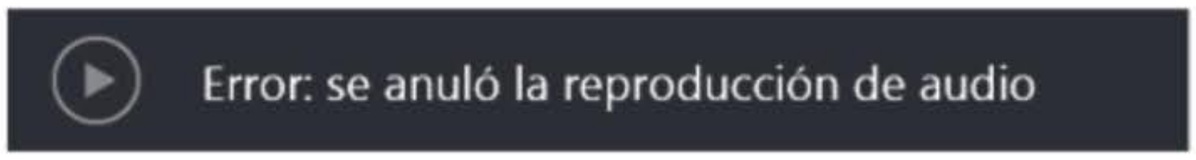
En la mayoría de los navegadores modernos, los tags **<audio>** y **<video>** pueden ejecutarse sin problemas. En la siguiente imagen, representamos de qué forma se visualizan en los navegadores web actuales: el único browser donde el tag <audio> no funciona es el viejo y ya obsoleto **Internet Explorer**.

GOOGLE CHROME

MOZILLA FIREFOX

MICROSOFT EDGE

INTERNET EXPLORER



Tengamos presente que, al momento de escribir esta obra, Microsoft Edge está distribuyendo las primeras **betas** de su nuevo web browser impulsado por el motor de Chromium. Por lo tanto, es factible que, en algún tiempo más, el look and feel del tag **<audio>** de Microsoft Edge termine asimilándose al que vemos ahora en Chrome.

Control multimedia integral desde JS

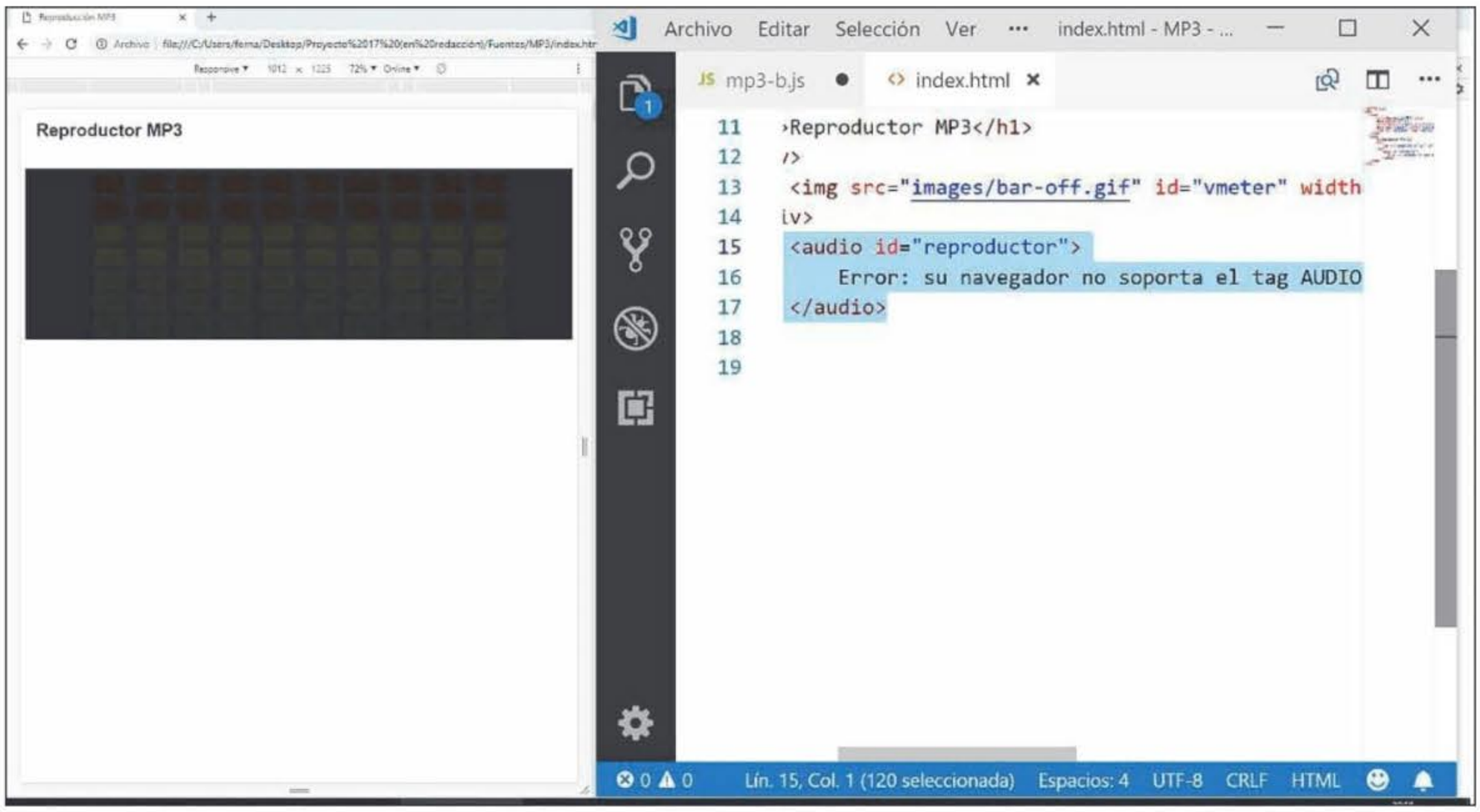
Modifiquemos nuestro último proyecto, renombrando el archivo **mp3. js** por **mp3-b.js**. Corrijamos a continuación la referencia hacia él en el documento HTML y eliminemos todo el código del archivo JS, exceptuando el **Event Listener** correspondiente a DOMContentLoaded, y la declaración de las variables rep e i. Luego, creemos una nueva variable, como muestra el siguiente código:

```
var archivoMP3 = "audio/Professor_Kliq_-_03_-_Bust_This_Bust_That.mp3";
```

Pasemos a modificar el documento HTML, eliminando en él la referencia **<source src...>** y el atributo controls del tag **<audio>**. Guardamos todo y verificamos el archivo HTML en el navegador web, que ahora no debe mostrar los controles correspondientes a dicho tag; solo el título y la imagen del vúmetro.

El tag **<audio>** sigue existiendo, aunque de forma invisible y sin un archivo de audio referenciado que podamos reproducir. A continuación, agreguemos en el documento HTML el siguiente código, justo debajo de la declaración del tag **</audio>**:

```
<button id="btn-cargar" >Cargar archivo</button>
<button id="btn-play" disabled>Reproducir</button>
<button id="btn-pause" disabled>Pausar</button>
<p id="nombrearchivo"></p>
```



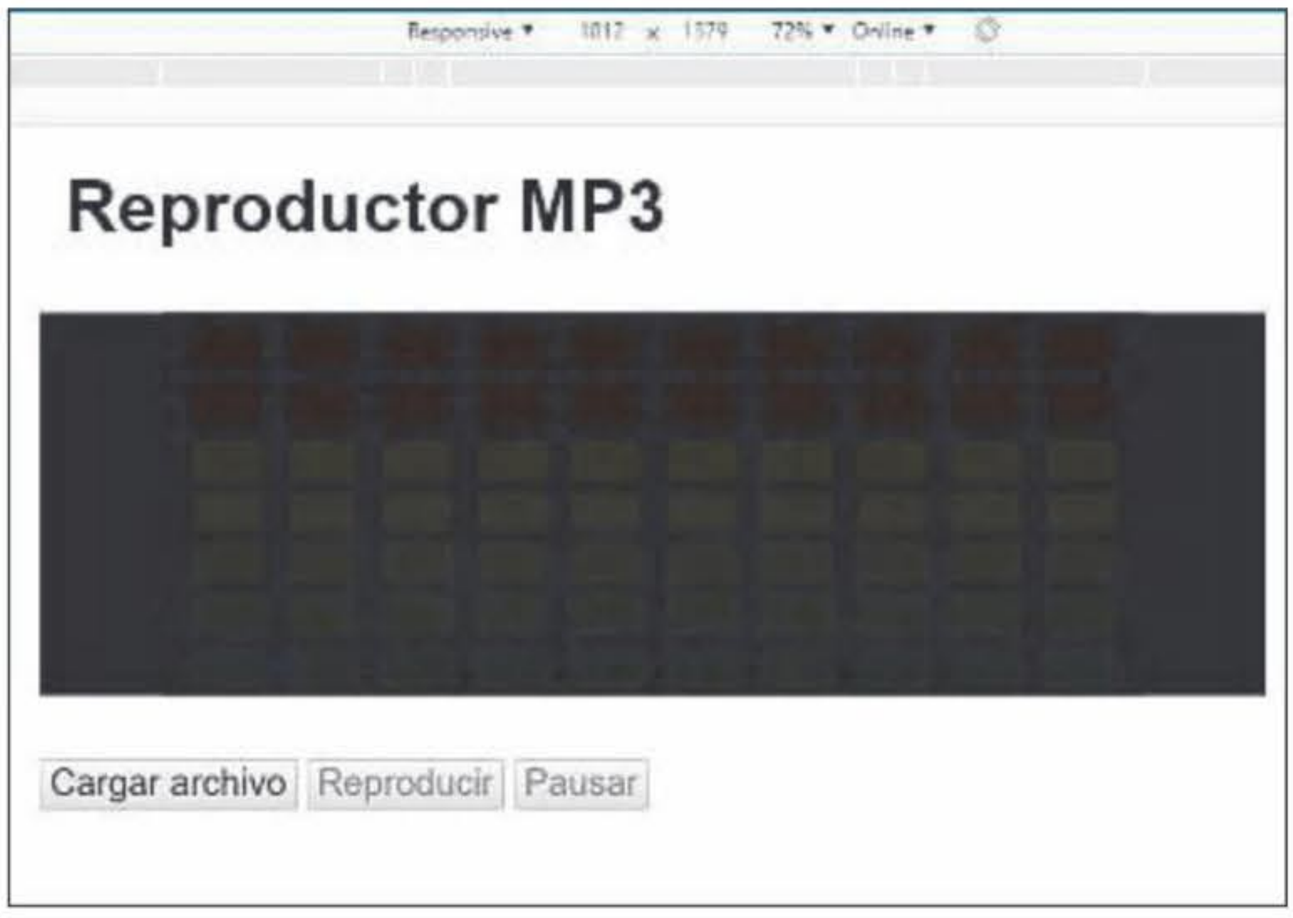
Con estos tres botones, nos ocuparemos de cargar el archivo de audio en el tag correspondiente, mostrar el nombre del archivo cargado en el tag **<p>**, y habilitar los botones **Reproducir** y **Pausar**. Añadimos el siguiente código en el archivo CSS para mejorar la visualización del texto de los botones:

```
button {
    font-size: 28px;
}
```

Y en el mismo archivo CSS, dentro de la referencia **body**, colocamos este otro código:

```
font-size: 28px;
margin: 30px;
```

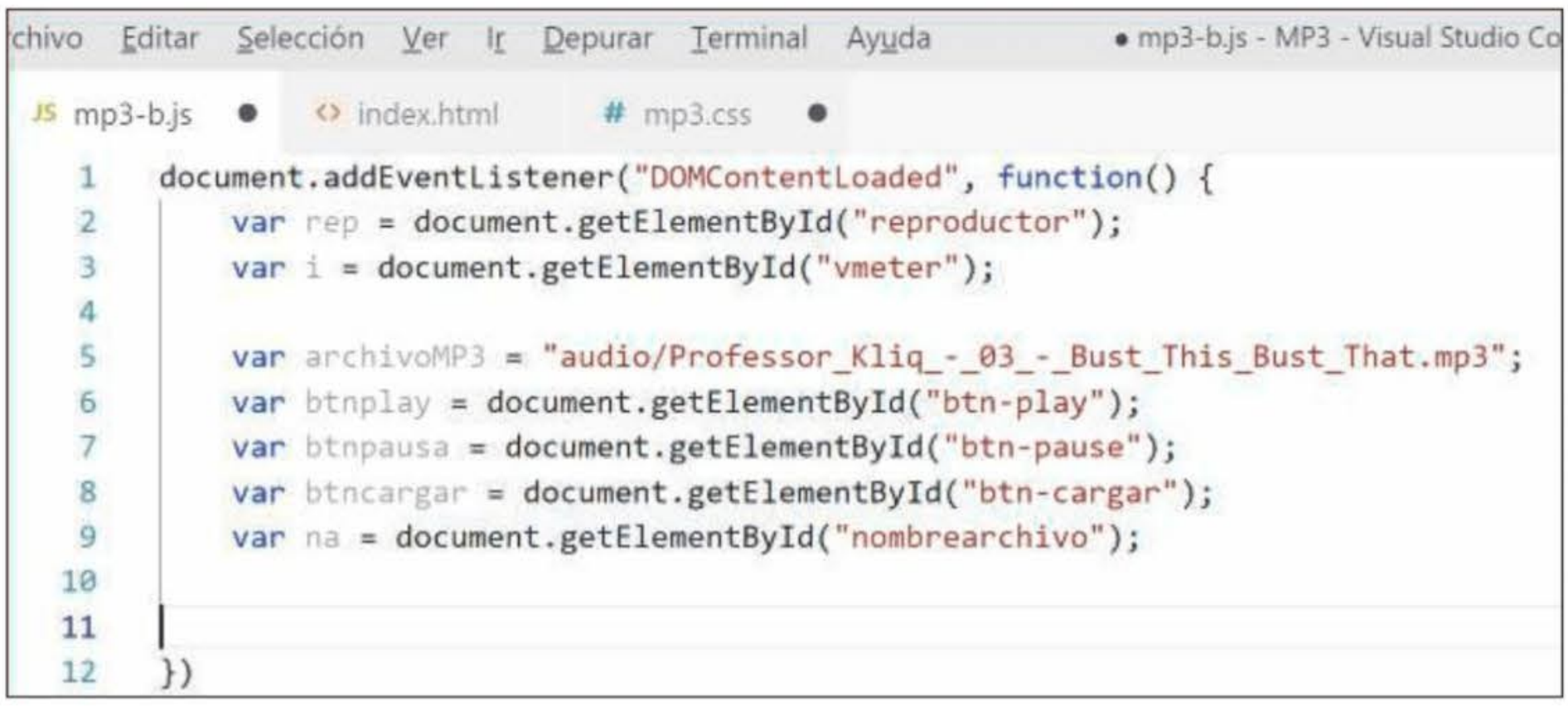
Nuestro proyecto debe lucir como se observa en la siguiente imagen:



Para darle nuevamente funcionalidad a nuestro reproductor MP3, agregamos en el archivo JS, justo después de la declaración de variables previamente realizadas, el siguiente bloque de código:

```
var archivoMP3 = "audio/Professor_Kliq_-_03_-_Bust_This_Bust_That.mp3";
var btnplay = document.getElementById("btn-play");
var btnpausa = document.getElementById("btn-pause");
var btncargar = document.getElementById("btn-cargar");
var na = document.getElementById("nombrearchivo");
```

En la variable **archivoMP3** almacenamos el **path** hacia el archivo de audio mp3. Las variables **btnplay**, **btnpausa** y **btncargar** generan la referencia hacia los botones homónimos, y la variable **na** almacena la referencia hacia el tag **<p>**. Con esto ya tenemos las variables necesarias para manipular, desde JS, los componentes HTML.



Ahora crearemos las funciones que les darán vida a nuestros botones.

Eliminar el atributo disabled

Para poder reproducir y pausar el contenido MP3, tenemos que activar los respectivos botones que, por defecto, se renderizan desactivados. Para hacerlo, creamos la siguiente función JS:

```
function estadoBotones() {
    btnplay.removeAttribute("disabled");
    btnpausa.removeAttribute("disabled");
}
```

Los tags **<button>** son renderizados por el motor HTML con el atributo **disabled**. Por esto, necesitamos llamar al método JS **removeAttribute()**, para quitarlo de los dos botones, pasándole el parámetro en cuestión que deseamos eliminar. Este atributo se encuentra en cada objeto JS del tipo **button**, y hará que estos se habiliten.

Cargar el archivo de audio

Declaramos a continuación la función que permite cargar el archivo de audio por reproducir. Este va indicado como parámetro a la función:

```
function cargarArchivo(file) {
    if (file != null) {
        rep.setAttribute("src", file);
        if (rep.src != "") {
            na.innerHTML = rep.src;
            estadoBotones();
        }
    }
}
```

Si el parámetro **file** de dicha función no está vacío, entonces configuramos el atributo **src** del objeto **rep**, con el nombre del archivo que queremos reproducir. Luego, solo por seguridad, validamos que el atributo **src** no llegue vacío. Si todo va bien, agregamos el nombre del archivo al tag **<p>**. Finalmente, invocamos el cambio de botones, llamando a la función **estaboBotones()**. Ahora, para darle vida al botón de carga de archivo, nos queda invocar la función **cargarArchivo()**:

```
btncargar.addEventListener("click", function() {
    cargarArchivo(archivoMP3);
})
```

Tal como vemos en la siguiente imagen, nuestra función se ejecutó correctamente cargando el archivo MP3 en el elemento **<audio>**, visualizando el path completo en el elemento **<p>**, y habilitando los botones de reproducción y pausa. Esta última función simplemente ejecutará la carga del archivo cuando pulsemos el botón en cuestión. Si bien nuestra aplicación aún no es totalmente funcional, al menos ya podemos validar la carga del archivo de audio y hacer que se visualice el **path** en el elemento **<p>** correspondiente.

Activar los botones de comando

Solo nos queda codificar estos últimos botones mencionados. Para hacerlo, agregamos al archivo JS el siguiente código:

```
btnplay.addEventListener("click", function() {
    rep.play();
    i.setAttribute("src", "images/bar-on.gif");
})
btnpausa.addEventListener("click", function() {
    rep.pause();
    i.setAttribute("src", "images/bar-off.gif");
})
rep.addEventListener("ended", function() {
    i.setAttribute("src", "images/bar-off.gif");
}
```

Con este último bloque de código activamos los eventos **adEventListener** que identifican cuándo presionamos **Play** y **Pausa**, y cuándo finaliza la reproducción de un archivo de audio. Cada uno de estos eventos dispara el código asociado para el comportamiento necesario de esta aplicación web multimedia. Ahora ya estamos en condiciones de ejecutar otra vez el proyecto y evaluar su funcionamiento:

Ejercicio práctico: potenciar el reproductor MP3

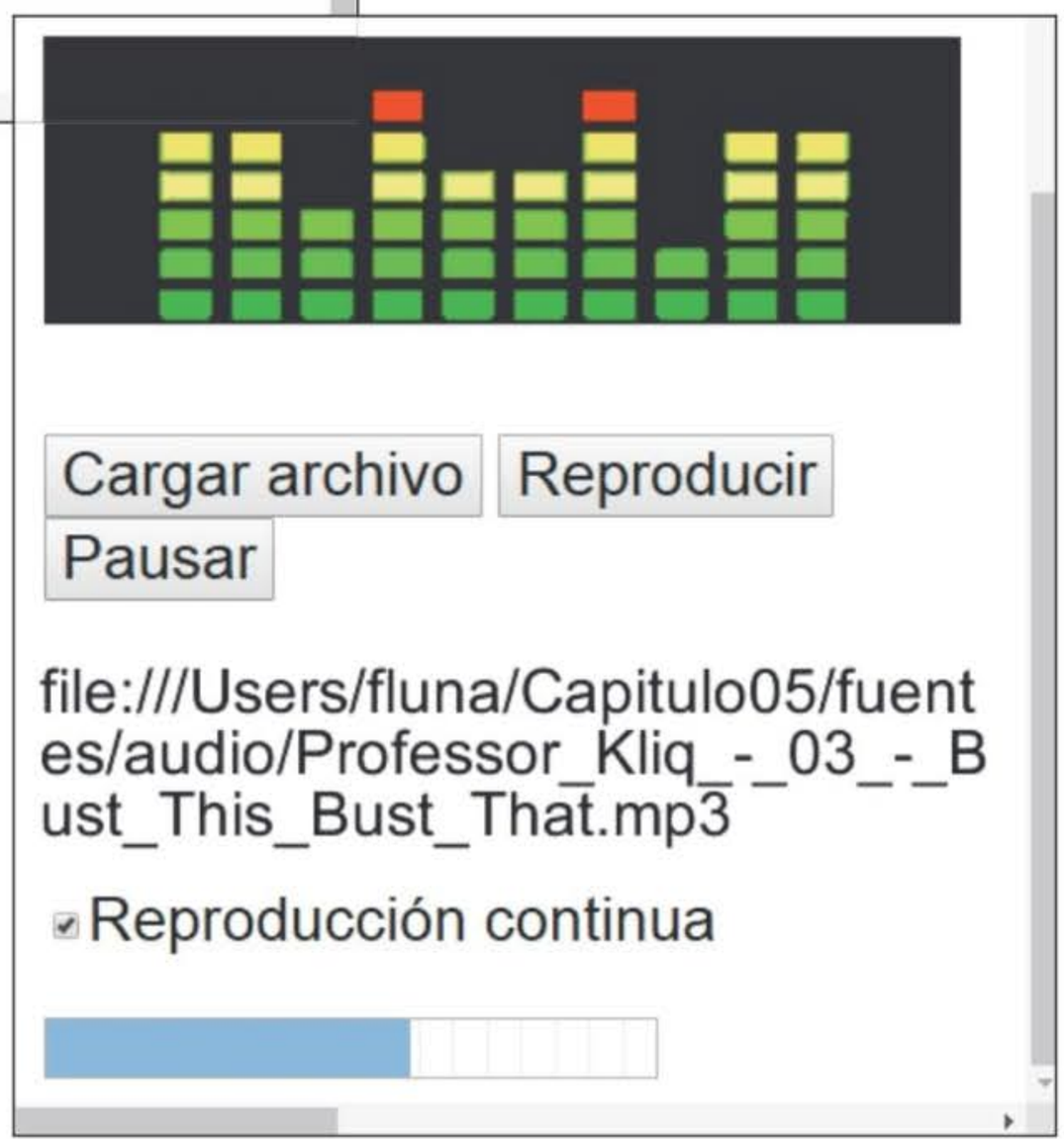
A continuación mejoraremos más la funcionalidad de nuestro reproductor MP3, agregándole los siguientes elementos:

- **Reproducción continua**

<input type="checkbox"> permite definir si el archivo de música se reproducirá de forma continua, mediante el atributo HTML **loop**, o una sola vez. **Loop** se activa cuando marcamos dicho componente.

- **Progreso de la reproducción**

A través del componente **<progress>**, sumaremos a la funcionalidad de nuestro reproductor MP3, la posibilidad de ver el progreso de reproducción de la pista de audio.



• **Volumen de audio**

Finalmente, aprovecharemos el componente **<input type="range">** para incorporar un control de volumen del archivo MP3.

Como una guía extra para este ejercicio, agregamos los siguientes datos respecto a cada componente que se debe utilizar en el desarrollo:

Range tiene los atributos **value**, **min** y **max**, los cuales deben utilizarse para identificar el nivel de volumen. Estos mismos atributos forman parte también del componente **Progress**, y se usan de manera similar que con Range. Finalmente, el atributo booleano **value** del componente **Checkbox** puede relacionarse directamente con el atributo **loop** del elemento **<audio>**.

05

El elemento <video>

Con algunas particularidades adicionales con respecto al elemento <audio>, <video> nos permite crear en pantalla la interfaz necesaria para controlar cualquier video que haya en una página web. Veamos a continuación cómo utilizar este tag en nuestros desarrollos.

El tag **<video>** engloba dentro de HTML el acceso al elemento en cuestión:

```
<video src="media/mivideoHD.mp4" controls>
    Lo sentimos: tu navegador no soporta el elemento video.
    :(
</video>
```

Como podemos ver en el código anterior, su uso es casi similar al que propone el tag **<audio>**. A través del atributo **src** especificamos la ruta hacia el video que deseamos visualizar en el documento HTML. Dicha ruta puede ser relativa a nuestro sitio, o una ruta completa hacia un video alojado en una web externa: www.otrositio.com/media/unvideoremoto.mp4.

Mensaje de no compatibilidad

Si bien en la actualidad es raro que alguien utilice un navegador web que no soporte este tag, siempre nos conviene agregar una leyenda dentro de los metatags **<video>** y **</video>**, que alerte al usuario sobre el problema mediante un mensaje claro.

Atributo controls

Para visualizar los controles por defecto que nos permiten manipular el video, utilizamos el atributo **controls**, dentro del primer tag. Obviarlo implicará no tener manera visual de ejecutar la reproducción.

Atributo autoplay

Si agregamos dentro del tag inicial el atributo **autoplay**, apenas se haya cargado el documento HTML y renderizado un porcentaje del video en cuestión, este comenzará a reproducirse sin intervención del usuario.

Atributo loop

También contamos con la posibilidad de que el video se reproduzca de forma constante, aprovechando para esto el atributo **loop**. Este, al igual que el resto de los atributos hasta aquí mencionados, se especifica dentro de la etiqueta **<video>** inicial.

```
<video src="media/mivideoHD.mp4" controls autoplay loop>
    <p>Tu navegador no implementa el elemento video</p>
</video>
```

Como podemos apreciar en la imagen anterior, la visualización del componente **<video>** difiere en cuanto a su estética cuando lo desplegamos en un navegador web con motor **WebKit** (**Apple Safari**) o cuando lo hacemos en un navegador web basado en **Blink** (**Chromium / Chrome**). De todos modos, estos cambios no afectan en nada a los atributos que posee el metatag **video**, que vimos y seguiremos viendo a lo largo de este capítulo.

Manos a la obra

Descargamos del repositorio de archivos de esta obra, el llamado **video-base.zip**. Lo descomprimimos y abrimos dicha carpeta como un proyecto nuevo de Visual Studio Code. Allí encontraremos dos documentos HTML: **index.html** e **indexb.html**. Ejecutamos el primero de ellos en el navegador web o en nuestro dispositivo móvil. Veremos un título y que el elemento video contiene un archivo multimedia referenciado. Si damos play, el video comienza a reproducirse.

Ahora cargamos en el navegador el segundo archivo HTML. Si lo visualizamos en un dispositivo móvil, es probable que no inicie automáticamente, pero si lo hacemos en un navegador web de escritorio, debe reproducirse sin nuestra intervención. La reproducción automática sucede gracias al atributo **autoplay** que tiene agregado el elemento **<video>**:

```
<video id="video" src="media/RU-video-intro.mp4" controls  autoplay>
```

Ancho del video

Si no especificamos un ancho determinado para el video, este podrá cargarse distribuyendo su ancho de acuerdo con el tamaño base del archivo. Si es fullHD (1920x1080), es probable que, al no controlarlo, la previsualización se salga de pantalla. Para evitarlo, agregamos el atributo **width** dentro del elemento **video**:

```
Width="80%"
```

Con esto, ya controlamos de manera segura que el ancho del video no sea mayor que el ancho de la pantalla del dispositivo. Refrescamos la página web cargada en el navegador para ir viendo los cambios aplicados.

Poster

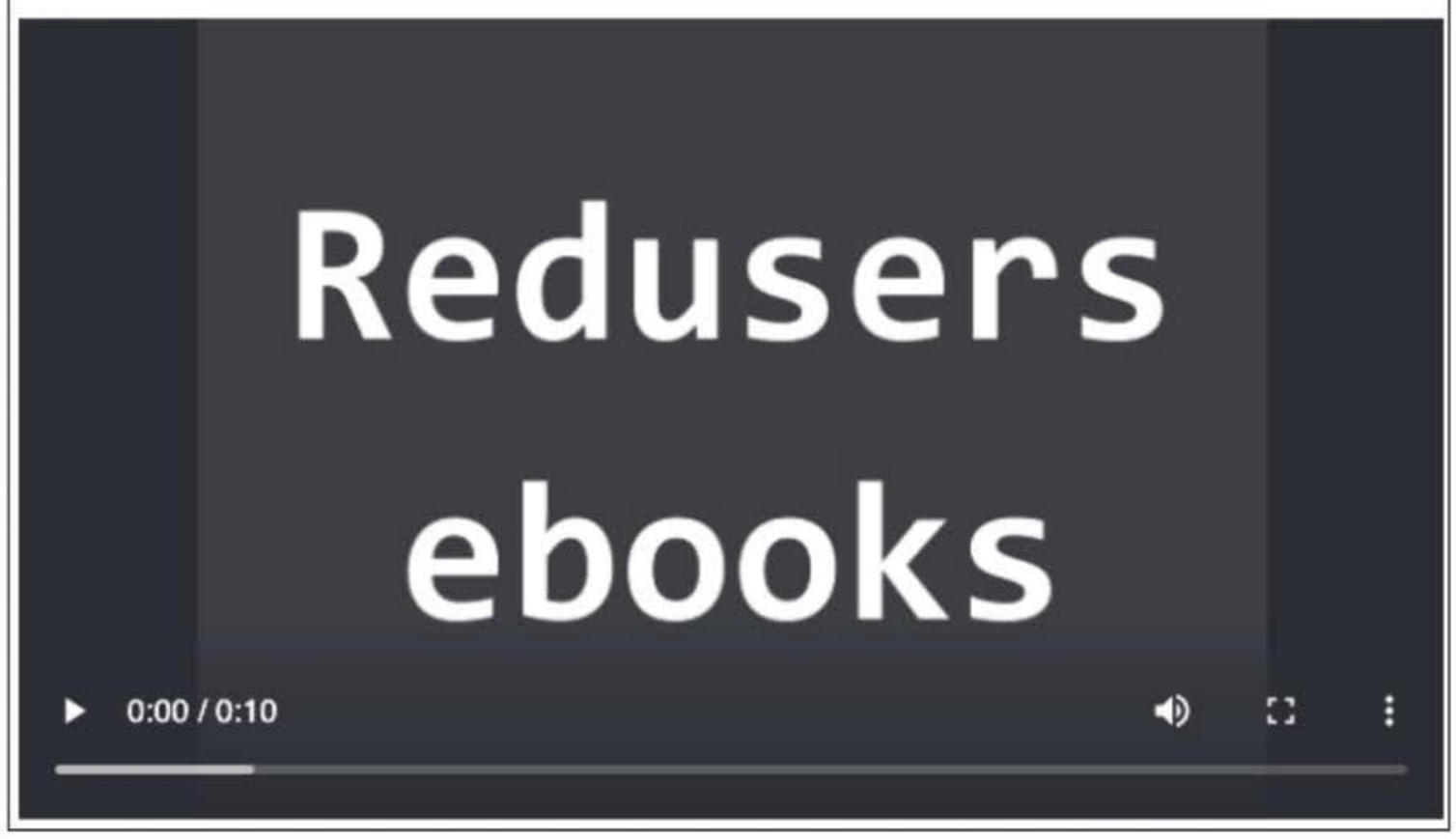
En los videos que cargamos en cualquier página, solemos ver un preview o una portada que ilustra el contenido de lo que veremos. El elemento **<video>** nos permite seleccionar una imagen para este fin, a través del atributo **poster**:

```
<video id="video" src="media/RU-video-intro.mp4" controls  width="80%"
poster="media/poster.png">
```

En el proyecto descargado, encontraremos imágenes que ofician de poster dentro de la subcarpeta **media**. Cargamos en nuestro proyecto de ejemplo el atributo **poster** referenciando a la imagen llamada **video-poster.png**.

Al visualizar el resultado del código anterior, vemos que la imagen seleccionada se corta en ancho respecto al ancho que tiene el video. Y como es probable que esto nos suceda frecuentemente, veamos a continuación una alternativa que nos permite subsanar esta diferencia de ancho entre la imagen del poster y del video en sí.

Video estándar



CSS background-color

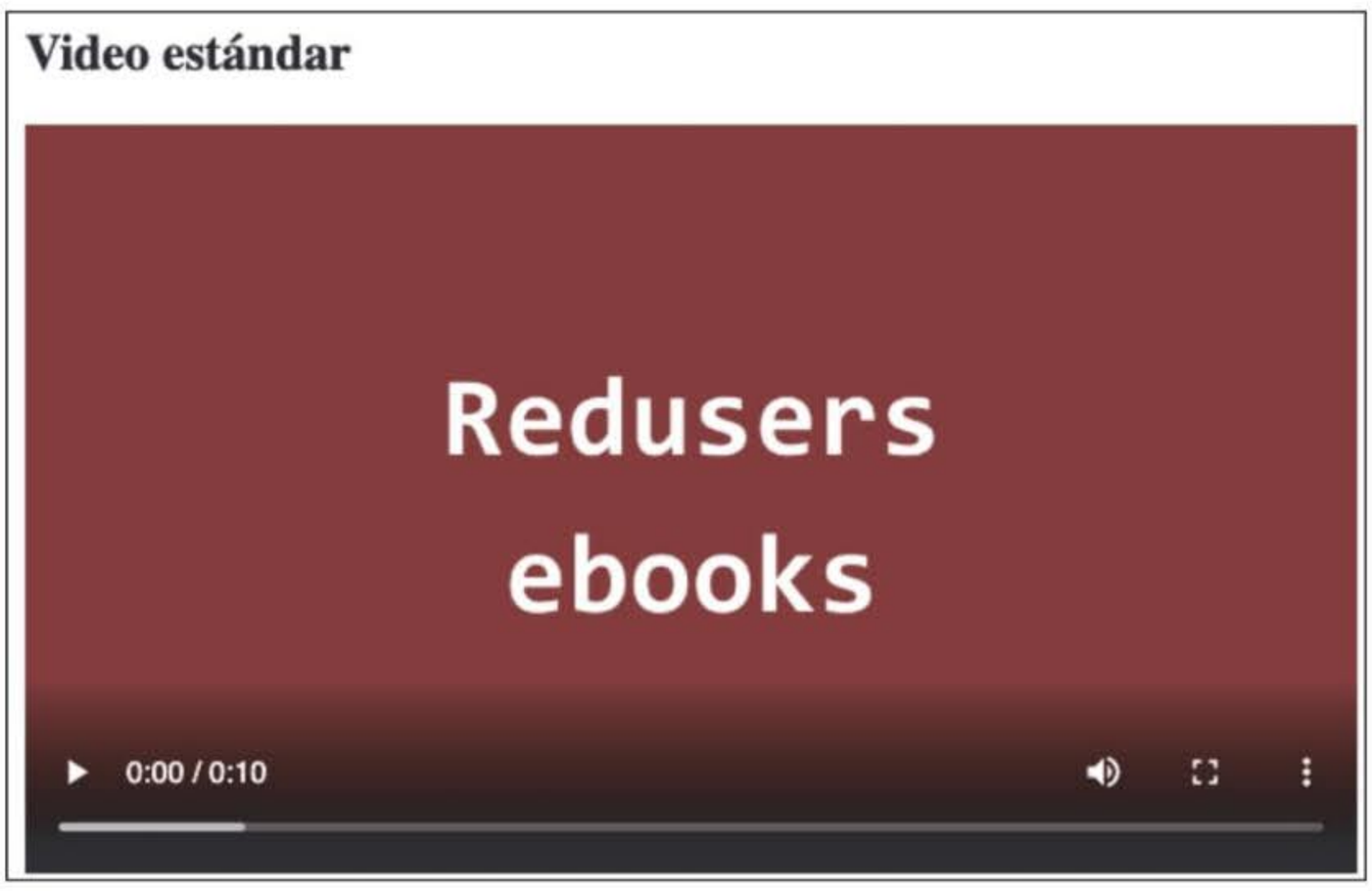
Lo primero que haremos será agregarle el atributo **style** al elemento **video**. Dentro de dicho atributo, especificamos el color de fondo del video, utilizando la propiedad CSS **background-color**. Le asignamos el **rgb(150, 66, 66)**, tal como vemos en la siguiente línea de código:

```
style="background-color: rgb(150, 66, 66);"
```

Luego modificamos el color de fondo de la imagen eliminando el negro por defecto y dejándolo transparente. Esto puede hacerse utilizando un archivo del tipo PNG o GIF, únicos formatos estándar que permiten establecer un fondo transparente. En la propiedad **poster** de nuestro video, cambiamos el nombre de archivo video-poster.png por **video-poster-tr.png**. El resultado debe ser similar a la siguiente imagen:

Ahora nuestro video se verá de color bordó. El archivo **video-base-terminado.zip** contiene todos los cambios aplicados hasta aquí.

Siempre tengamos presente que este atributo aplicado desde una hoja de estilo CSS sobre el elemento **video** no funciona. Solo lo hace si lo aplicamos mediante el atributo **style** mencionado.



Múltiples formatos de video

Al igual que el elemento **<audio>**, **<video>** soporta la configuración de múltiples formatos de archivo. Para esto, utilizaremos **<source>**, como lo hicimos anteriormente. Modificamos el proyecto **video-base-terminado.zip**. En la subcarpeta de este proyecto hay dos videos iguales, con diferente formato:

* RU-video-intro.mp4
* RU-video-intro.ogg

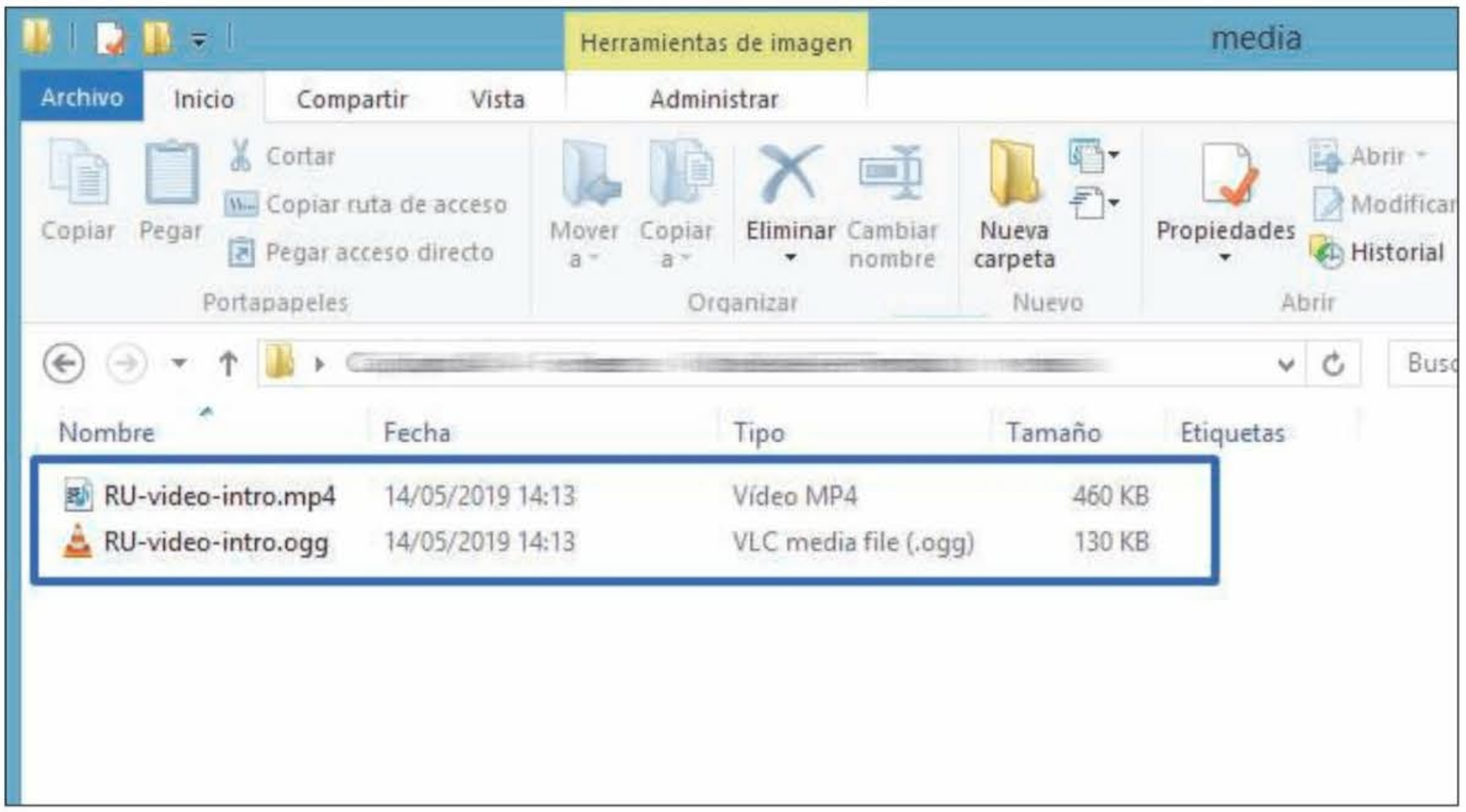
Cargamos la referencia a ambos videos con el metatag correspondiente, para permitir que sea el navegador web quien elija el video más óptimo. Cambiamos entonces el código de html, para que quede como el siguiente:

```
<video ...
<source src="media/RU-video-intro.mp4" type="video/mp4" />
<source src="media/RU-video-intro.ogg" type="video/ogg" />
...
</video>
```

Veamos en la imagen cómo debe quedar estructurado nuestro código:

```
<!DOCTYPE html>
<html>
    <head>
        <meta charset="utf-8">
        <title>Elemento Video</title>
        <meta name="viewport" content="width=device-width, initial-scale=1.0">
        <link rel="stylesheet" type="text/css" href = "css/style.css" />
    </head>
    <body>
        <h2>Video estándar</h2>
        <video id="video" controls  width="80%" style="background-color: rgb(150, 66, 66);'
        poster="media/video-poster-tr.png">
            <source src="media/RU-video-intro.mp4" type="video/mp4" />
            <source src="media/RU-video-intro.ogg" type="video/ogg" />
            <p>Su navegador web no soporta reproducción de video.</p>
            <p><strong>:(</strong></p>
        </video>
</html>
```

Pero, **¿cuál es la diferencia entre ambos videos?** Principalmente, lo que distingue a un formato del otro es el tipo de **códec** utilizado para comprimir los fotogramas. Esto no solo le da una extensión diferente al archivo, sino que también mejora el tiempo de carga del video en los navegadores web y no estropea la calidad de imagen. En la siguiente imagen se muestra la diferencia de lo que ocupa en disco el mismo video en dos formatos distintos (MP4 y OGG).



Formatos de video soportados según el navegador

Existen tres formatos de video estándar soportados por los navegadores web actuales, aunque no todos los browsers los soportan todos. Se trata de los dos vistos hasta ahora (MP4, OGG) y, además, **WebM**. En la siguiente tabla se muestra un resumen de los navegadores web más populares y el soporte que cada uno les da (o no) a estos formatos.

TABLA 2

NAVEGADOR	MP4	WebM	Ogg
Chrome	Sí	Sí	Sí
Firefox	Sí	Sí	Sí
Safari	Sí	No	No
Opera	Sí	Sí	Sí
Edge	Sí	No	Sí

Media Types

Como vimos en el bloque de código anterior, el tipo de video está determinado por el atributo **mediaType**. Cuando declaramos el tag **<source src...>**, debemos también incluir el mediaType en cuestión, para facilitarle al navegador web la tarea de identificar el video que más le conviene cargar. La siguiente tabla muestra los media type de cada formato:

TABLA 3

FORMATO DE ARCHIVO	MEDIA TYPE
MP4	mediaType="video/mp4"
WebM	mediaType="video/webm"
Ogg	mediaType="video/ogg"

Propiedades y atributos del elemento video

Si bien este elemento comparte muchos atributos y propiedades con **<audio>**, dada la complejidad que tiene un video, pueden existir o no el soporte de cada uno de estos atributos o propiedades que componen el elemento. Veamos en la siguiente tabla un resumen de estos, destacando el tipo de navegador según el tipo de plataforma (móvil o web).

Como esta referencia puede cambiar constantemente, los invitamos a consultar, llegado el caso, el link oficial de **Mozilla Developer Network**: https://mzl.la/2w1QzGR.

	Chrome	Edge	Firefox	Internet Explorer	Opera	Safari	Android webview	Chrome for Android	Edge Mobile	Firefox for Android	Opera for Android	Safari on iOS	Samsung Internet
video	3	Yes	3.5	9	10.5	3.1	Yes	Yes	Yes	4	Yes	Yes	Yes
autoplay	3	Yes	3.5	9	10.5	3.1	Yes	Yes	Yes	4	Yes	10 *	Yes
buffered	?	Yes	4	?	Yes	?	?	?	Yes	4	Yes	?	?
controls	3	Yes	3.5	9	10.5	3.1	Yes	Yes	Yes	4	Yes	Yes	Yes
crossorigin	?	Yes	12	?	?	?	?	?	Yes	14	?	?	?
height	3	Yes	3.5	9	10.5	3.1	Yes	Yes	Yes	4	Yes	Yes	Yes
intrinsicsize	71	?	?	No	58	No	71	71	?	?	50	No	?
loop	3	Yes	11	9	10.5	3.1	Yes	Yes	Yes	14	Yes	6	Yes
muted	30	Yes	11	10	Yes	5	Yes	Yes	Yes	14	Yes	?	Yes
played	?	Yes	15	?	Yes	?	?	?	Yes	15	Yes	?	?
poster	3	Yes	3.6	9	10.5	3.1	Yes	Yes	Yes	4	Yes	Yes	Yes
preload	3 *	Yes	4	9	Yes *	3.1	Yes *	Yes *	Yes	4	Yes *	Yes	Yes
src	3	Yes	3.5	9	10.5	3.1	Yes	Yes	Yes	4	Yes	Yes	Yes
width	3	Yes	3.5	9	10.5	3.1	Yes	Yes	Yes	4	Yes	Yes	Yes

Full support — No support

Compatibility unknown — Experimental. Expect behavior to change in the future.

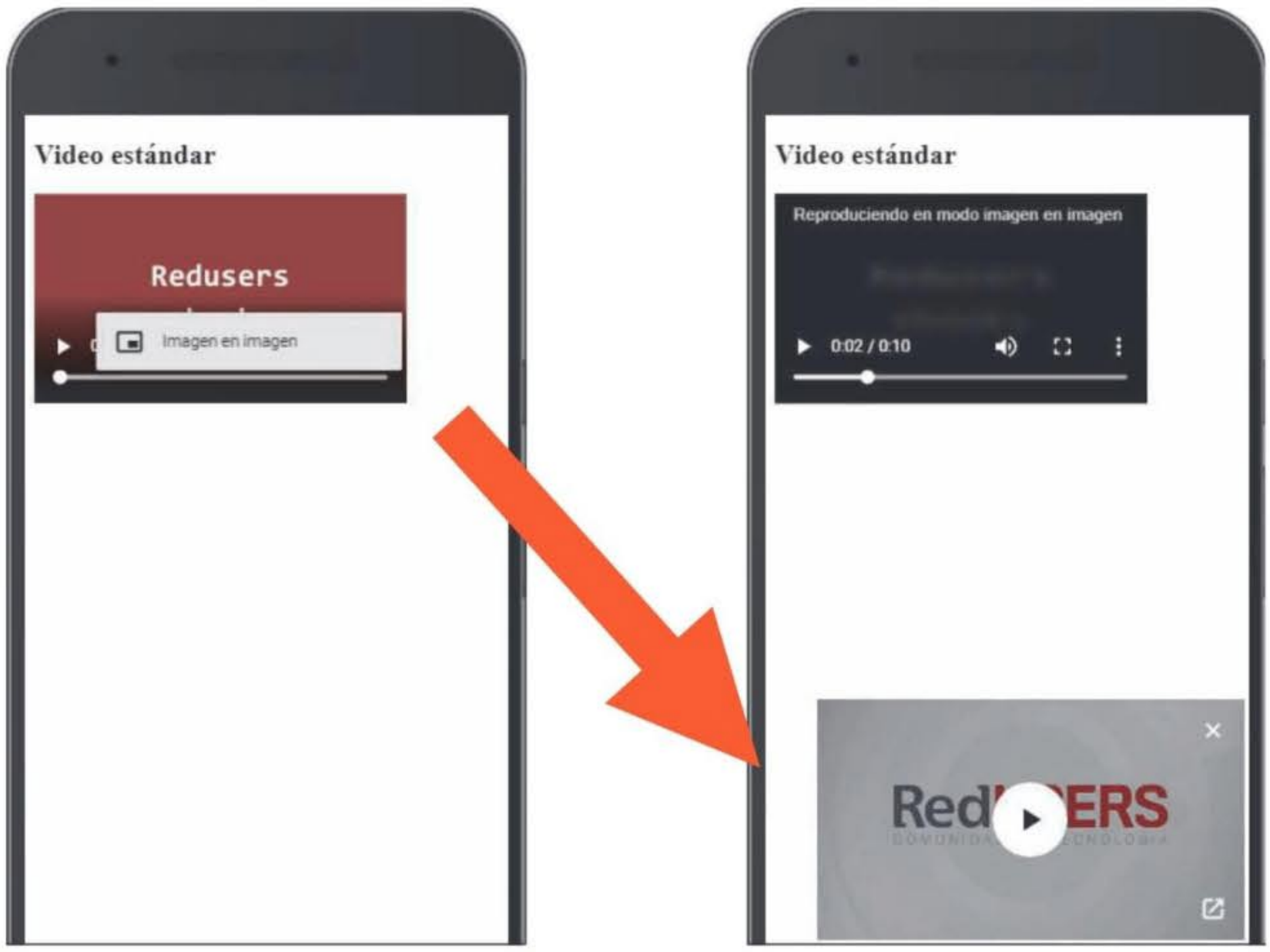
Picture in Picture

Esta característica es muy nueva y está presente solo en algunos navegadores web. Chrome, tanto en su versión de escritorio como en su versión mobile, la tiene implementada en su motor web. Por lo tanto, en cualquier video que compartamos a través de una web y que se visualice en Chrome, el usuario cuenta con la posibilidad de desplegar la característica **Picture in Picture** en su computadora o dispositivo móvil.

En la próxima figura, podemos ver que el reproductor de video HTML despliega, a través de un menú ubicado en el extremo inferior derecho, la opción de activar **PiP**. Al pulsar en ella, se desprende un marco del video y se aloja por defecto en el extremo inferior de la pantalla del móvil.

Este quedará suspendido por encima de cualquier ventana que abramos hasta tanto lo cerremos, e independientemente de si está reproduciendo

contenido o no. Por su parte, Firefox no cuenta con el soporte para PiP, al menos hasta el momento de escribir esta publicación.



Como podemos observar en la siguiente imagen, los controles de video de Firefox difieren de los mostrados, por ejemplo, en Chrome, y no incluyen la opción PiP. Seguramente, en versiones futuras de todos los navegadores web, esta opción se vuelva una característica por defecto del reproductor que despliega el componente HTML video.

Picture in Picture y JavaScript

Para conocer cómo manejar Picture in Picture desde JavaScript, es posible acceder a http://bit.ly/2W55Fdg (en inglés), un tutorial creado por el mismo Google. En él se incluyen todos los trucos y formas necesarias para sacarle provecho a este sensacional modo de visualización de videos.

Video tracks

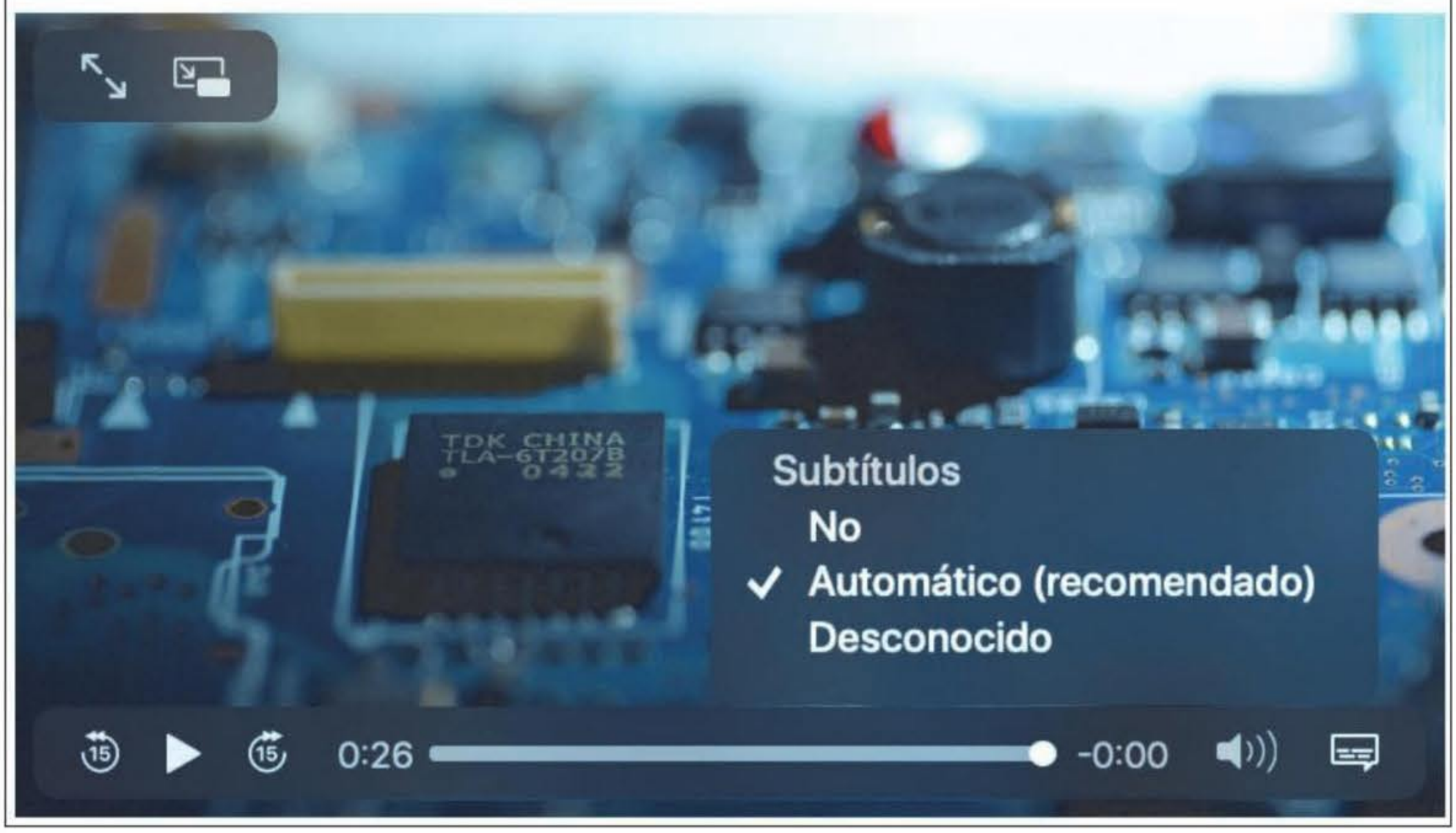
Como todo sistema de video actual, este elemento incorpora también la capacidad de subtitular los videos que necesitemos, aprovechando para hacerlo la etiqueta **<track>**. Esta se agrega dentro de los tags **<video>** y **</video>** y nos permite, entre otras cosas, traducir el video en curso, subtitular, agregar descripciones en pantalla y hasta brindarle al usuario la selección del subtítulo en múltiples idiomas.

Su uso es muy simple; veamos la estructura de este tag a continuación:

```
<video id="video" src="media/mivideo.mp4" poster="poster.png" controls
width="70%">
    <track default src="media/subtitulos.vtt">
</video>
```

Como podemos notar en el bloque de código anterior, la etiqueta **<track>** contiene una serie de parámetros internos que permiten definir la forma en la cual se agregarán los subtítulos al video. Mediante el atributo **src**, indicamos el archivo que contiene el subtitulado o las leyendas que deseamos visualizar. Por otro lado, el atributo **default** permite indicar que dicho archivo específico debe cargarse por defecto junto al video.

Video con tracks



Estructura de los archivos VTT

Los archivos con extensión VTT son archivos planos de texto que permiten, en su interior, definir los textos o leyendas que deben verse junto al video. Su estructura es muy sencilla:

```
WEBVTT - subtitles
1
00:01.000 --> 00:04.000
Hola, soy un subtítulo para un video.
...
```

El archivo se inicia con la sentencia **WEBVTT**. El guión y texto siguiente son opcionales, y sirven para tener una referencia de qué es lo que compone el archivo. Luego comienzan los espacios de subtitulado. Para agregar un subtítulo, primero debemos definir la estructura de tiempo en la cual se visualizará el párrafo. Después establecemos el texto en sí que deseamos mostrar en pantalla.

Para ver todo de forma cómoda, conviene siempre agregar una numeración inicial consecutiva, y luego especificar el período de tiempo y texto por mostrar. Repetimos este formato a lo largo del video, todas las veces que lo necesitemos.

La temporización que indica en qué momento de tiempo debe aparecer el subtítulo se estructura de la siguiente forma: **HH:MM:SS.CCC** (horas, minutos, segundos, centésimas). Como vemos, podemos especificar una precisión más que aceptable. Si el video dura algunos segundos, podemos obviar el parámetro de tiempo de horas y/o minutos.

El período de tiempo durante el que se mostrará el subtítulo se especifica mediante la sentencia **-->**; y antes y después de ella debemos poner el tiempo en cuestión. El párrafo que se mostrará puede estar separado del temporizador mediante un simple espacio de teclado, aunque siempre es mejor dar un Enter y escribir debajo.

Notas adicionales en el archivo

En caso de que necesitemos agregar alguna nota dentro del archivo, a modo de referencia, podemos escribirla anteponiendo a ella la sentencia **NOTE**. De esta forma, el procesador de subtítulos obviará todo el texto que siga luego de dicha sentencia.

Esto es ideal para comentar dentro del archivo, pero debemos tener cuidado de no ser muy extensos, ya que el archivo de subtítulos puede volverse ilegible para el programador o quien necesite acceder a él para modificarlo.

index.html | track_ejemplo.vtt ●

media ▸ track_ejemplo.vtt

```
1    WEBVTT - subtitles
2
3    1
4    00:01.000 --> 00:04.000
5    Así se visualiza un track para video.
6
7    2
8    00:05.000 --> 00:09.000
9    La estructura de archivo es simple.
10   Y puede tener múltiples renglones.
11
12   3
13   00:11.000 --> 00:15.000
14   Ya puedes implementarlo en tus desarrollos móviles.
15
16   4
17   00:18.000 --> 00:24.000
18   Y hasta modificar a gusto con CSS la estética de estos captions
19
20   NOTE Esto no saldrá impreso en pantalla
21
```

Visualización de subtítulos

Algunos motores de navegación permiten activar o desactivar los subtítulos. En la próxima imagen, vemos que Apple Safari nos da esa opción a través del menú de interacción con el video en curso. Por defecto, este navegador web tiene la visualización de títulos de manera automática.

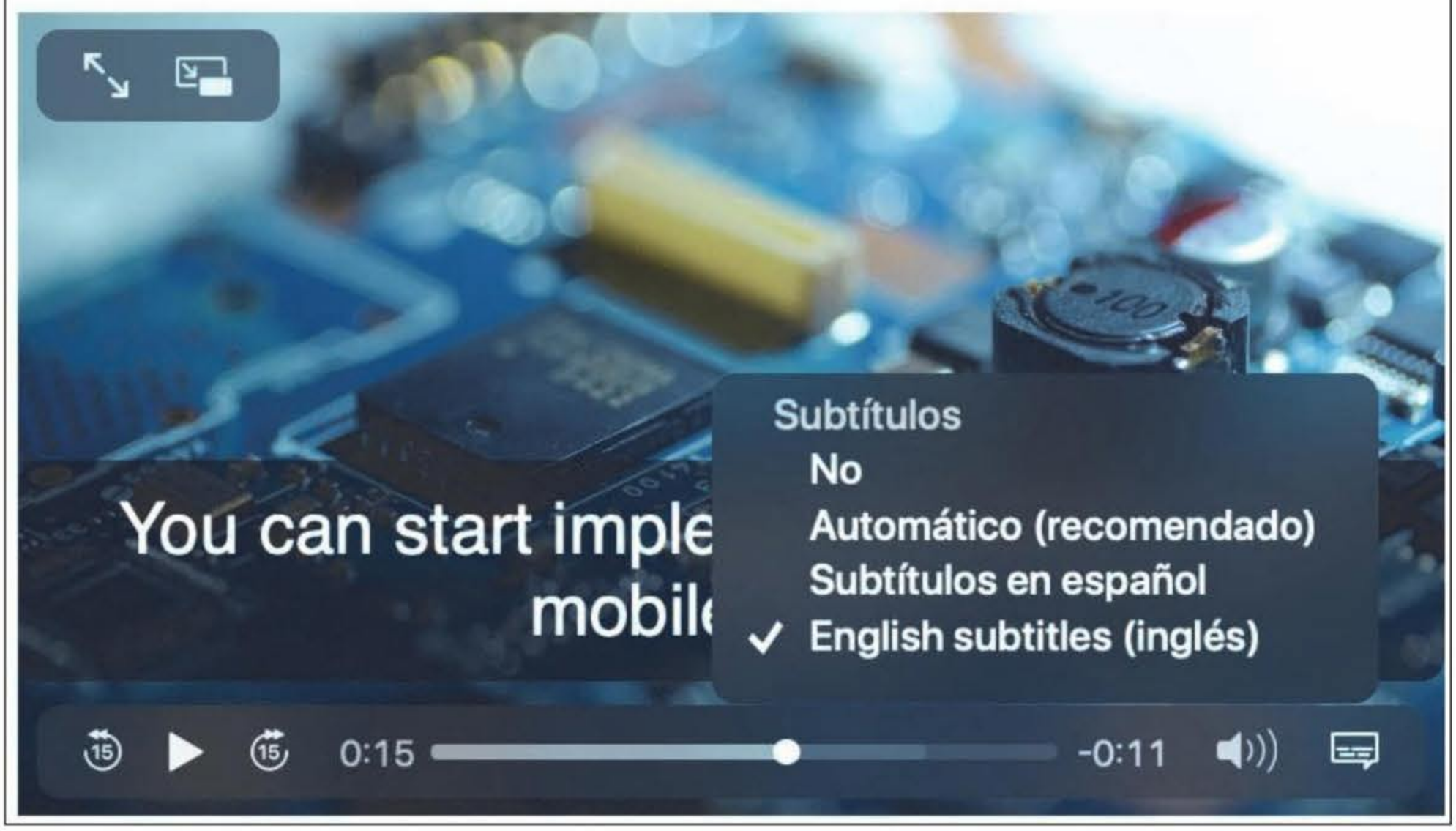
Múltiples idiomas

Esta misma etiqueta nos permite establecer el subtitulado de un video en múltiples idiomas. Para aprovechar esta capacidad, simplemente debemos crear el tag **<track>** una vez por cada idioma que queramos ofrecerle al usuario. Tengamos presente que, en este caso, uno de ellos podrá llevar el atributo default, para que se cargue de manera automática junto al video.

Por cada definición de un nuevo track, agregamos dos atributos adicionales: **label** y **srclang**. El primero nos permite establecer una descripción amigable para el archivo de subtítulos, y el segundo hace referencia al código ISO internacional de dicho idioma:

Video con tracks

Video con tracks



```
<video src="mivideo.mp4">
  <track label="English subtitles" src="subtitles_en.vtt" srclang="en"
default></track>
  <track label="Deutsche Untertitel" src="subtitles_de.vtt"
srclang="de"></track>
</video>
```

Prerrequisitos para tener en cuenta

El uso de subtítulos en los videos requiere que el desarrollo y la prueba correspondientes se hagan bajo un web server. Ninguna prueba funcionará ejecutando directamente el documento HTML desde nuestra computadora. Por lo tanto, recomendamos iniciar Fenix webserver y configurar un website para llevar a cabo todas las pruebas necesarias.

Material de referencia

En el repositorio de archivos que acompaña a esta obra, se incluyen dos ZIP que contienen los proyectos web tratados en estas últimas páginas. El archivo Video tracks.zip tiene un ejemplo funcional de cómo se comporta el archivo de subtítulos, y el archivo **Video multiples idiomas.zip** trae otro ejemplo funcional con tracks en español e inglés para el mismo video.

Ejercicio práctico: video multi-idioma

Tomando como premisa el archivo **video tracks.zip**, es posible expandir las posibilidades de múltiples idiomas de dicho video, agregando al menos tres o cuatro opciones diferentes, como: **inglés**, **italiano**, **portugués**, etcétera. Utilicen **Google Translate** para traducir cada párrafo del archivo VTT a dichos idiomas, y generen un nuevo archivo por cada uno. Finalmente, agréguenlos al video de dicha página utilizando el tag correspondiente.

El sistema de subtitulado no es solamente para agregar una traducción a otros idiomas. También podemos tener presente agregar subtítulos a los videos en nuestro propio idioma, para incluir así a personas con incapacidad auditiva, promoviendo de esta forma la inclusión y ampliando a su vez el público de nuestros videos.

06 Integración de redes sociales

Muchos años han pasado desde que el universo “social” comenzó a ser parte de los sitios web, mostrando actividad de este ecosistema y dándonos beneficios para los sistemas de comentarios. Veamos qué nos ofrecen estas redes a través de sus herramientas para developers.

Seguramente hemos navegado de forma reiterada a través de diferentes sitios web que muestran en sus páginas distintos sistemas de actividad social, como un sistema de comentarios al pie de una publicación, el total de seguidores sociales de un sitio o las últimas publicaciones de este en la red Twitter. A través de este capítulo, recorreremos las diversas herramientas para desarrolladores que las redes sociales ponen a nuestra disposición, y veremos de qué modo podemos integrarlas en nuestros sitios web.

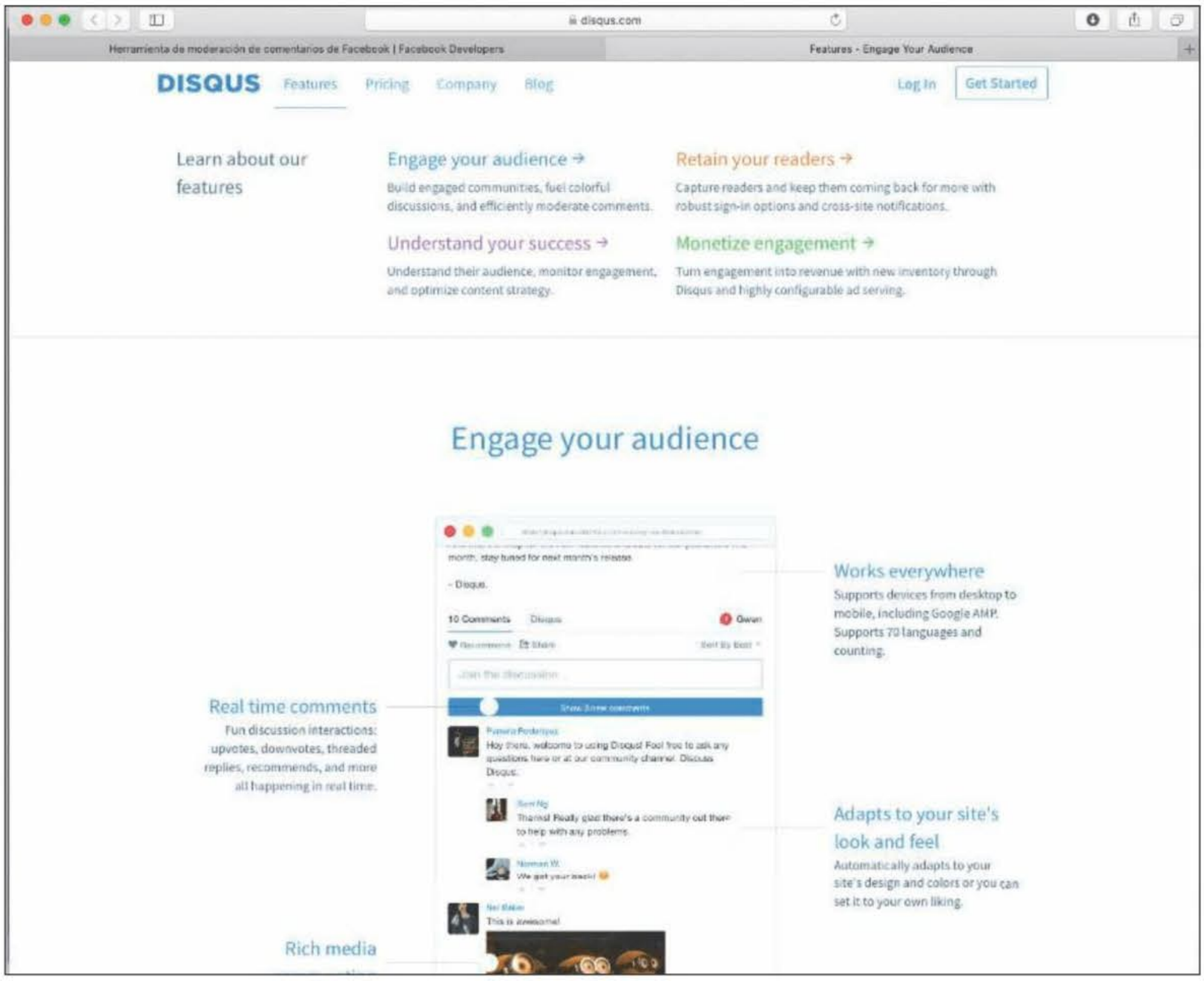
Ventajas de la integración de plugins sociales

Una de las principales ventajas de utilizar un plugin social es la posibilidad de evitar comentarios anónimos y spam. Dentro del universo de blogs basados en **WordPress**, muchos de ellos utilizan el sistema de comentarios estándar que ofrece esta herramienta, lo cual se torna una desventaja, ya que, por defecto, permite aceptar comentarios anónimos.

El uso de otras redes sociales, como Facebook, no solo tiene la opción de evitar el anonimato y el spam, sino que también se puede aprovechar el tráfico proveniente de esta red social si tenemos una página de fans o corporativa, dado que cualquier comentario que dejen en una de nuestras publicaciones se verá reflejado en nuestra página social y en la del usuario que escribió, y así se expandirá mucho más la visibilidad de nuestro sitio.

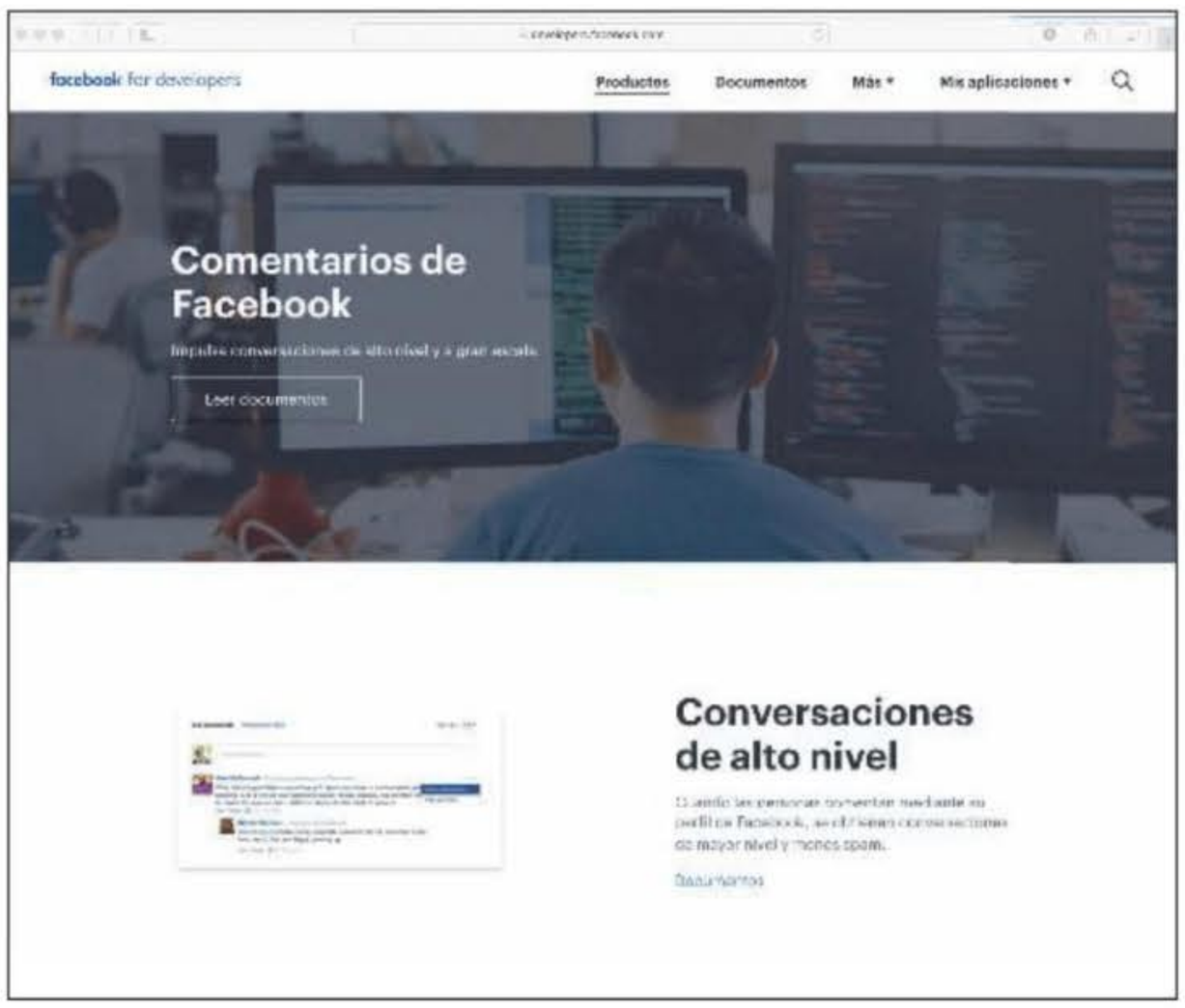
Sistema de comentarios de Facebook

Una web que sube noticias de forma periódica a través de un blog o de su propio sistema de publicaciones suele poner a disposición de los lectores algún sistema de comentarios. Si bien existen decenas de opciones en Internet, por ejemplo, para integrar los sistemas de comentarios de **Disqus**, debemos reconocer que



Facebook, más allá de algunas controversias que ha generado estos últimos años, sigue siendo la más popular y la que mayor visibilidad puede darle a nuestro sitio web, blog de noticias o sitio corporativo.

Como es de esperar, las redes sociales cuentan con una sección dedicada a desarrolladores. Facebook, en particular, posee un sitio web destinado a programadores web y mobile, con un sinfín de opciones y soluciones para integrar en casi cualquier plataforma: https://developers.facebook.com.



Requisitos para utilizar Facebook Developer

Lo primero que debemos hacer es ingresar en developers.facebook.com con la cuenta de usuario personal o corporativa asociada a esta red social. Luego ubicamos la sección **Docs** y, dentro de ella, la sección **Plugins/Comments**.

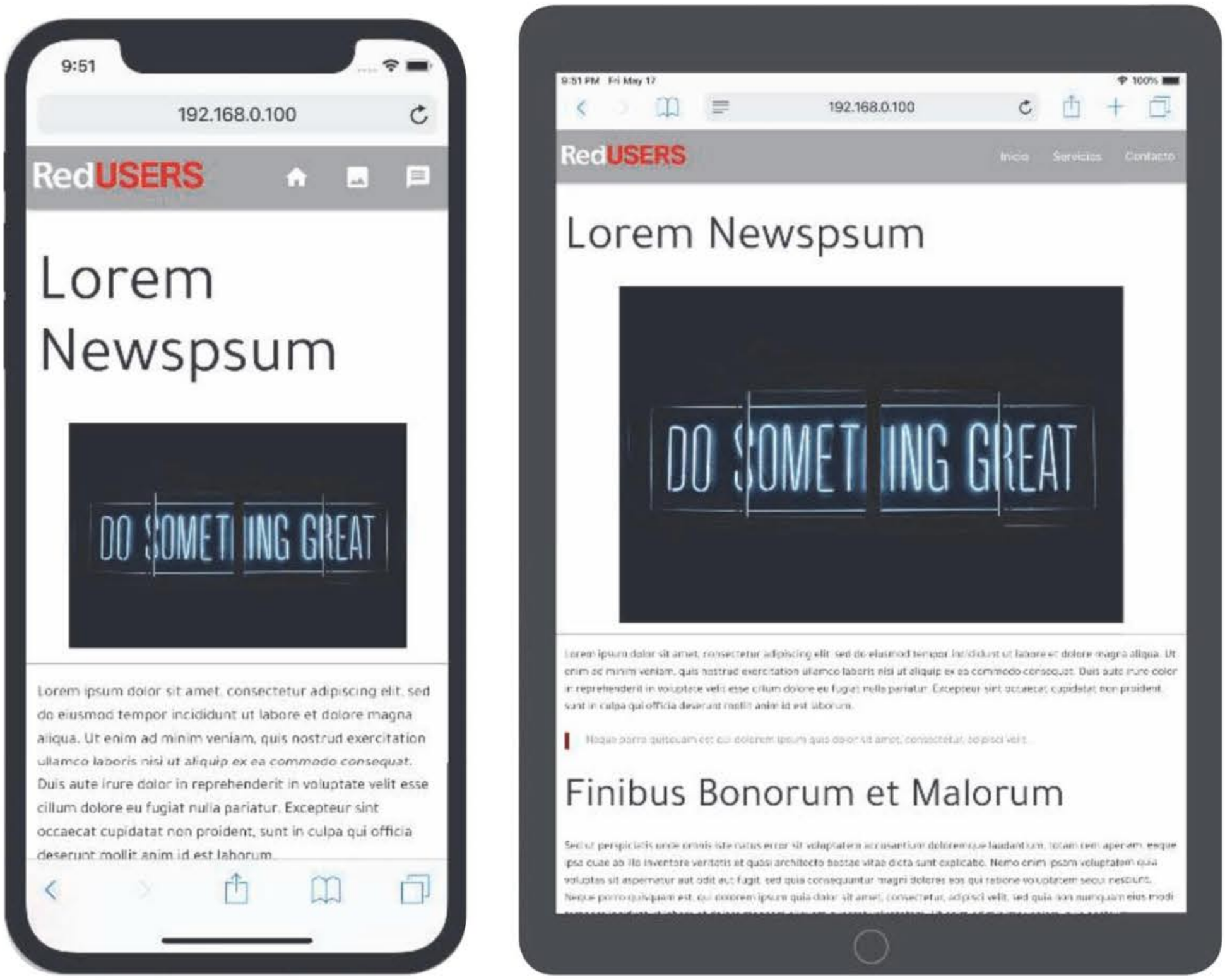
En este apartado encontraremos una serie de botones con los cuales vamos a interactuar a continuación.

TABLA 1

OPCIÓN	DESCRIPCIÓN
Comments plugin configurator	Nos da acceso a una herramienta visual que nos permite completar datos fácilmente.
Moderation	Brinda acceso al sistema de moderación, para establecer reglas sobre lo que se escribe (esta opción necesita generar primero un Facebook App ID).
Settings	Nos lleva a una tabla con pautas de personalización del plugin. Podemos elegir colores, número de posts, orden cronológico, uso o no de optimización móvil, entre otras.
Adjust Language	Permite cambiar el lenguaje en el cual Facebook mostrará los botones y etiquetas de la sección Comentarios.

Con esta rápida revisión de las herramientas que utilizaremos a continuación, pongamos entonces manos a la obra en nuestro primer proyecto, descargando del repositorio de archivos el ZIP denominado **social-base.zip**. Creamos un nuevo proyecto en Visual Studio Code, agregando la carpeta contenida en este ZIP.

Editaremos a continuación el archivo index.html y lo ejecutaremos (utilizando Fenix Web Server) en un navegador web o dispositivo móvil. Para hacerlo, necesitamos un web server que nos permita visualizar los comentarios de Facebook, porque estos no se verán si ejecutamos el archivo index.html directamente desde el disco local de la computadora.



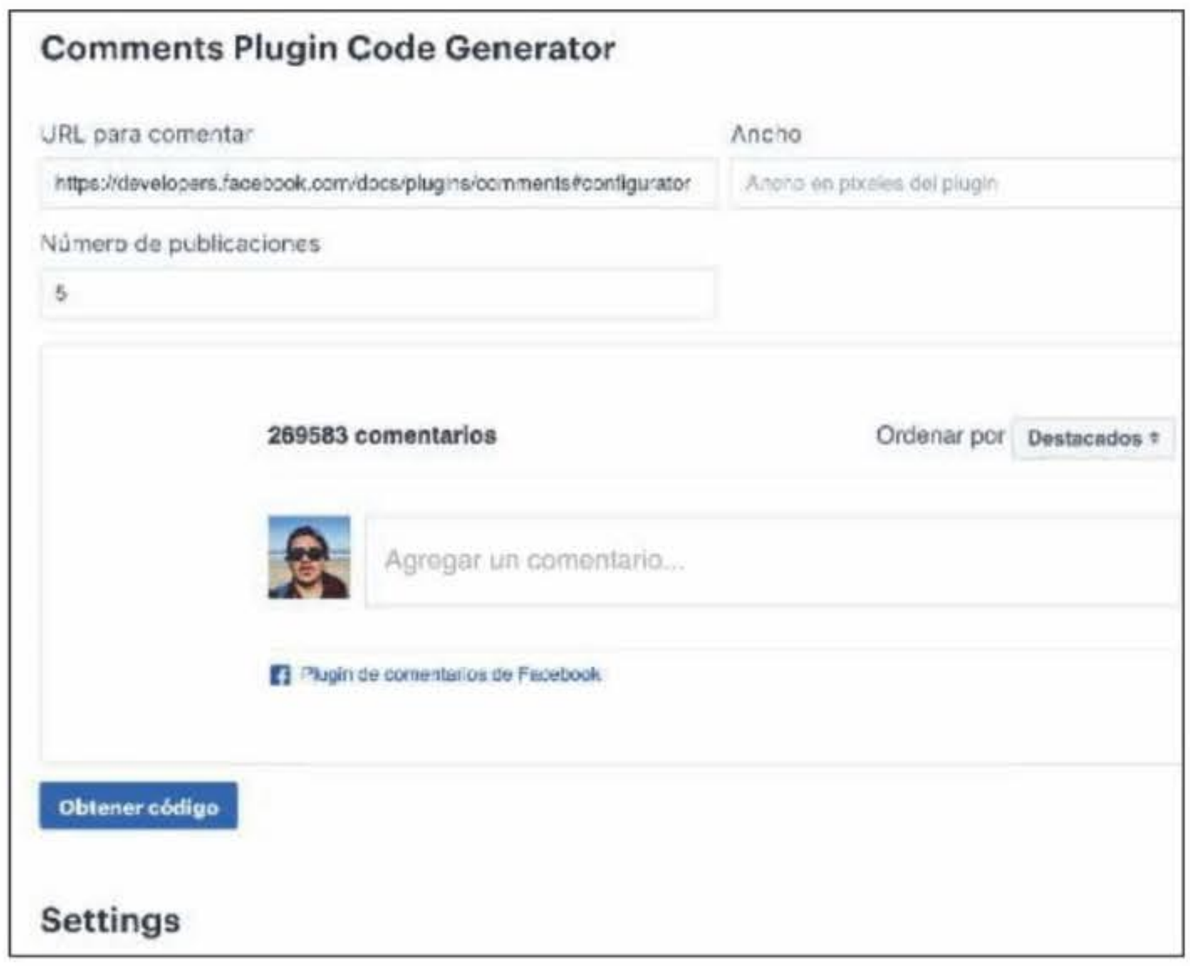
Como podemos ver en la imagen anterior, el archivo index.html tiene un look and feel similar a uno de los ejercicios realizados anteriormente. Este documento HTML representa una noticia, en cuyo pie de página, agregaremos el plugin de comentarios de Facebook.

Volvamos al código fuente HTML. No vamos a modificarel código existente, solo incorporaremos dos bloques de código dentro del tag **<body>**. Estos bloques, usualmente llamados **snippets**, se ocuparán de conectar la herramienta de comentarios de Facebook y visualizar dicho sistema para que cualquier lector interactúe. Uno de estos bloques de código será insertado justo debajo del elemento **<navbar>**, y el otro lo agregaremos al final del HTML, justo antes del cierre **</body>**.

Configurar nuestro plugin social

Volvamos entonces a la web de desarrolladores de Facebook. Ubicamos el apartado **Comments Plugin** y pulsamos en el botón **Comments Plugin Configurator**. Nos llevará a la sección específica, donde encontraremos un sistema de configuración básico para generar luego el código del sistema de comentarios.

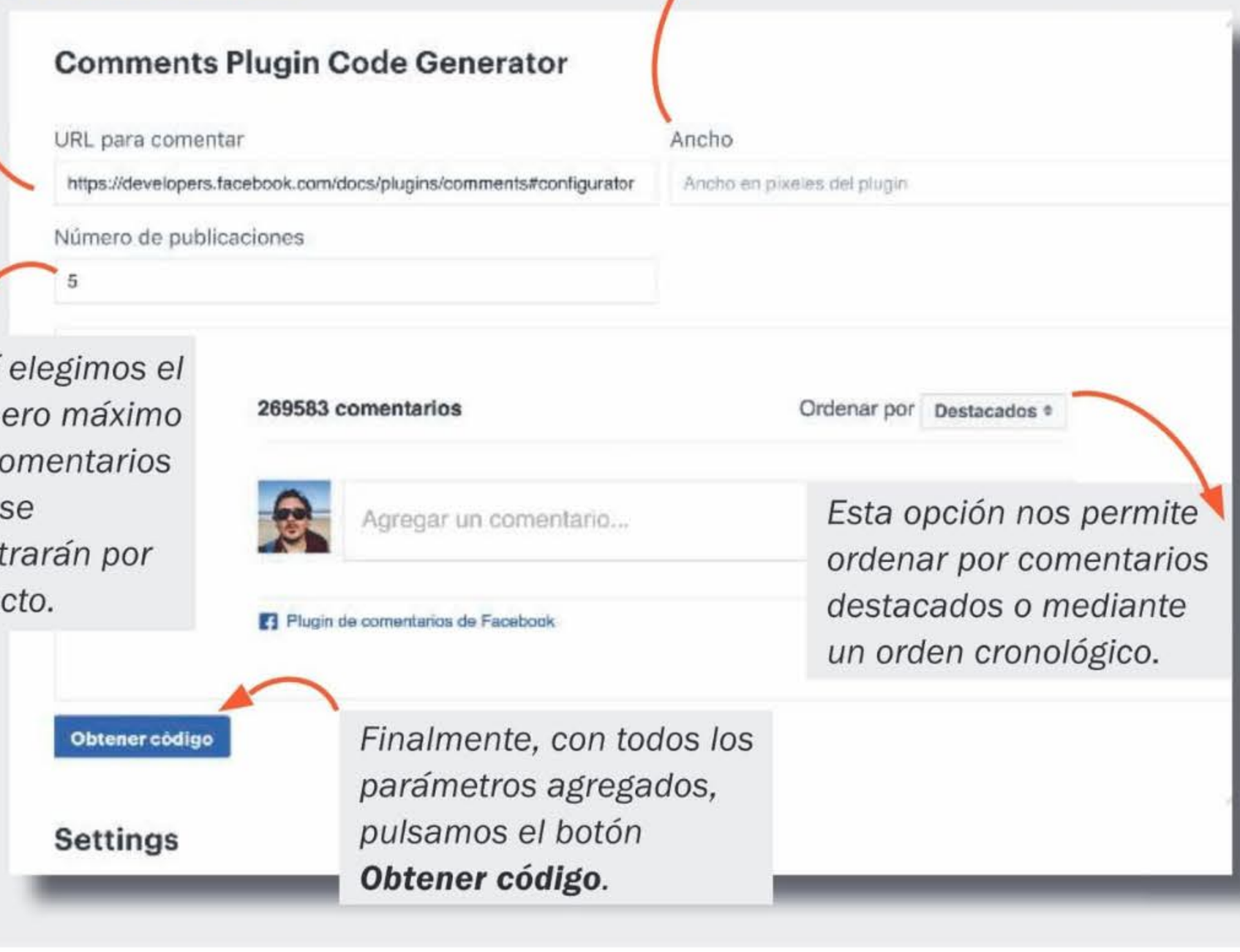
Analicemos este a través de la siguiente Guía Visual, para entender qué datos precisamos ingresar.



GUÍA VISUAL 1

En este campo agregamos la URL de nuestro sitio; por ejemplo: www.misitioweb.com/blog. Para este ejemplo en particular, debemos incorporar la URL que nos devuelve Fenix Web Server, que puede ser similar a la siguiente: http://192.168.0.100:81.

Agregamos un ancho específico en píxeles que tendrá el apartado de comentarios. Pensando que esto está orientado a móviles, pongamos, por ejemplo: ***400****.*

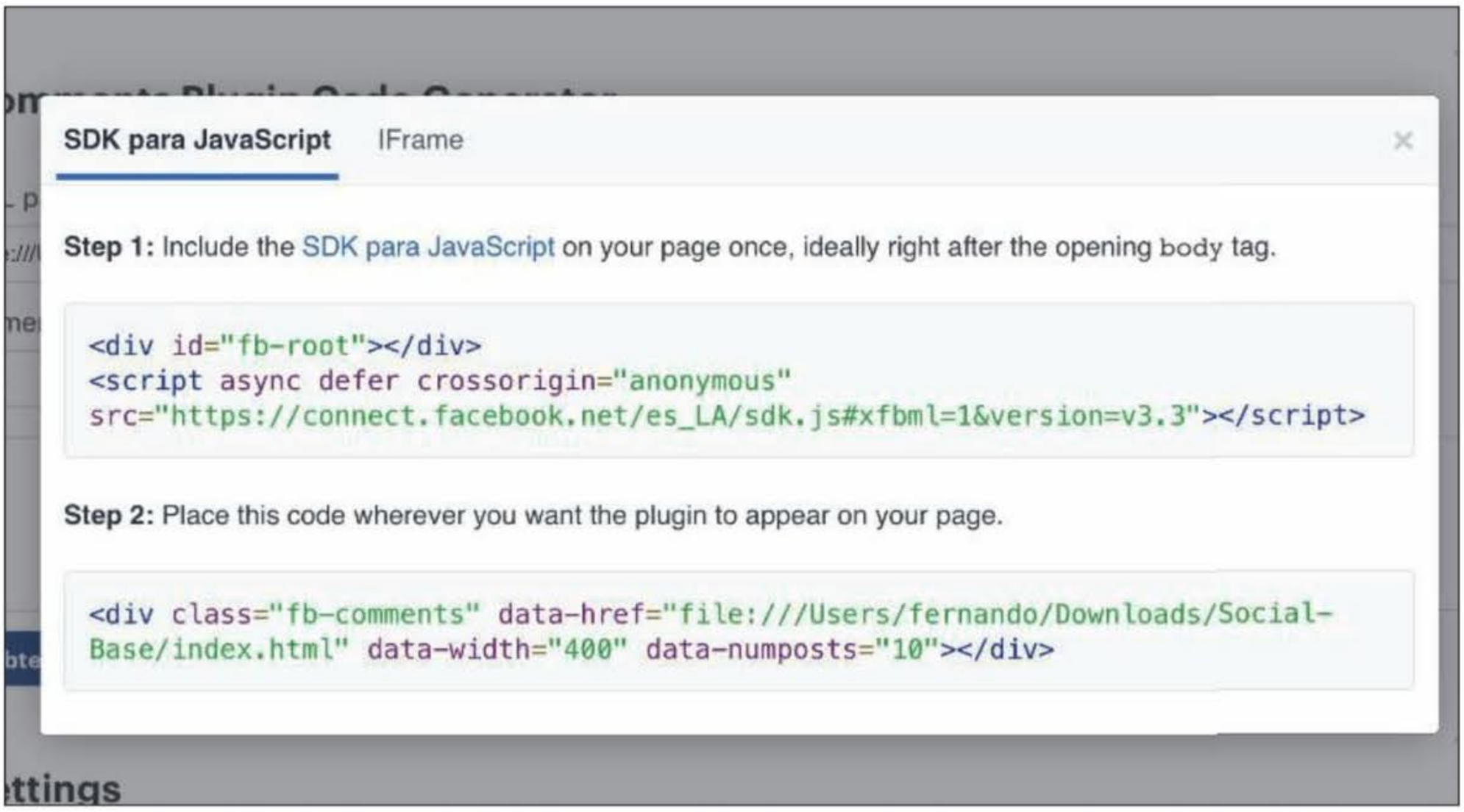


Aquí elegimos el número máximo de comentarios que se mostrarán por defecto.

Esta opción nos permite ordenar por comentarios destacados o mediante un orden cronológico.

Finalmente, con todos los parámetros agregados, pulsamos el botón ***Obtener código****.*

La configuración explicada en esta infografía nos dará como resultado dos bloques de código, tal como lo mencionamos anteriormente. Estos bloques de código serán visualizados por Facebook Developers a través de una ventana emergente, como vemos en la siguiente imagen:



Tal como lo indican los pasos detallados en la infografía, lo primero que debemos hacer, es seleccionar el bloque de código superior y pegarlo en algún lugar cercano al tag **<body>**, que corresponde al tag de apertura. Nosotros agregaremos dicho código en el documento HTML, justo después del cierre **</navbar>**.

```
...
    <div id="fb-root"></div>
    <script async defer crossorigin="anonymous" src="https://connect.
facebook.net/es_LA/sdk.js#xfbml=1&version=v3.3">
    </script>
<main>
```

Este primer bloque de código se encarga de llamar a un archivo JavaScript, propio de Facebook, que se ocupa de inicializar las herramientas de esta red social para tener disponible el elemento Comentarios dentro de nuestro sitio web. Nos queda agregar el bloque de código correspondiente a los comentarios en sí. Debemos añadirlo al finalizar el artículo de este documento HTML, por lo que lo hacemos después de la etiqueta **</main>** y antes de la etiqueta **<footer>**:

```
</main>
<hr/>
<div class="fb-comments" data-href="http://localhost:81/" data-
width="400" data-numposts="10"></div>
<footer class="…
```

Como podemos apreciar, este segundo bloque de código se compone de un ** ** con una serie de atributos. En ellos configuramos los parámetros correspondientes que le darán vida y estética al sistema de comentarios de Facebook integrado a nuestro sitio web. Veamos a continuación, un resumen de ellos y su funcionalidad.

TABLA 2

ATRIBUTO	DESCRIPCIÓN
data-colorscheme	Esquema de colores usado para personalizar los comentarios; puede ser **light** o **dark**.
data-href	URL absoluta donde impactarán los comentarios que se realicen. Si bien nosotros especificamos la URL base de nuestro sitio de ejemplo, para blogs o sitios web con noticias dinámicas, este atributo deberá completarse de forma dinámica.
data-mobile	Valor booleano (true – false) que notifica al plugin si deberá adaptarse a un formato óptimo para dispositivos móviles.
data-numposts	Permite variar el número de comentarios por defecto que se muestran.
data-width	Soporta un valor numérico en píxeles, como también un valor numérico en porcentaje. Por defecto, ante la carga de la página en un dispositivo móvil, su valor cambia automáticamente al 100%.

Personalizar el lenguaje

En la siguiente Guía Visual veamos cómo personalizar el idioma de los comentarios, de acuerdo con nuestra necesidad.

Como podemos observar en el gráfico, el llamado a la API de Facebook incluye una URL, donde encontramos el lenguaje que debe utilizar la sección de comentarios. Para cambiarlo, simplemente reemplazamos esta porción de la URL por el código **ISO** correspondiente al lenguaje que deseamos: **en_US** / **fr_FR** / **it_IT** / **es_ES** / **en_CA**, entre otros tantos.

GUÍA VISUAL 2

Ejecución en modo asincrónico: funciona independientemente del resto de la webapp.

***defer**: se ejecuta cuando haya finalizado la carga de todo el documento HTML.*

No necesitamos credenciales de Facebook para utilizar este SDK.

```
<script async defer crossorigin="anonymous"
src="https://connect.facebook.net/es_LA/sdk.js#xfbml=1&version=3.3">
```

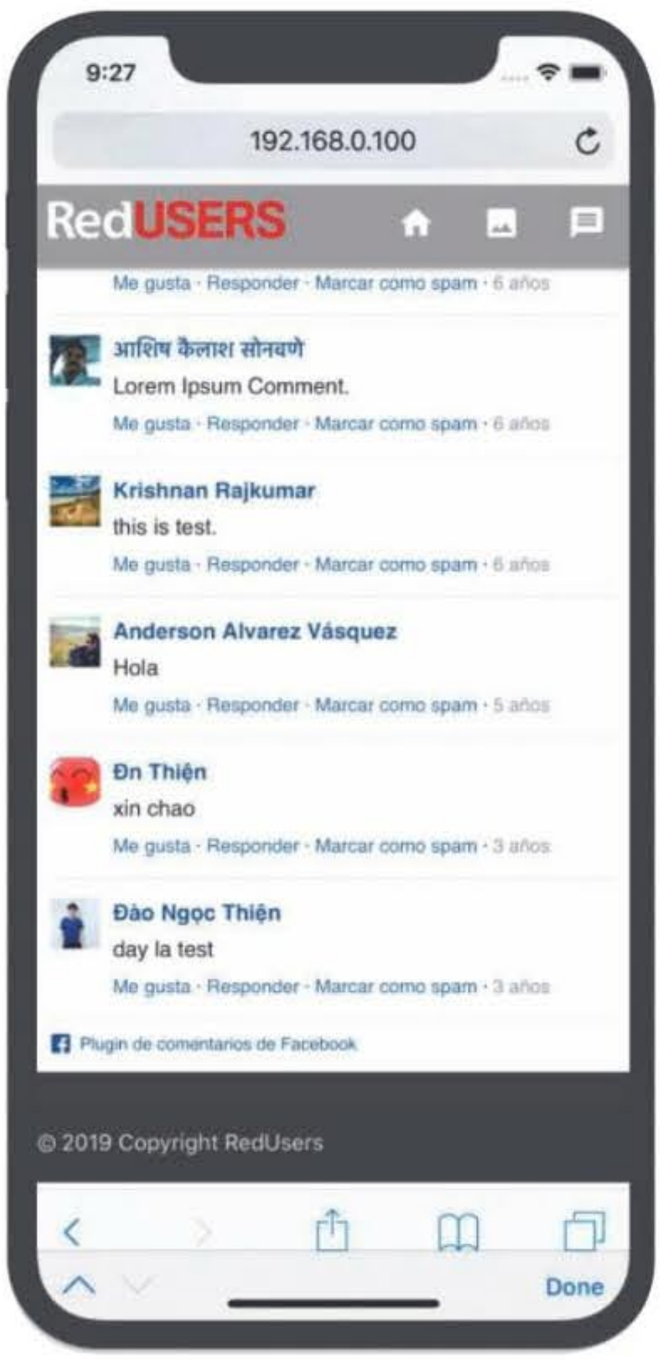
es_LA corresponde al código ISO: Español-Latinoamérica.

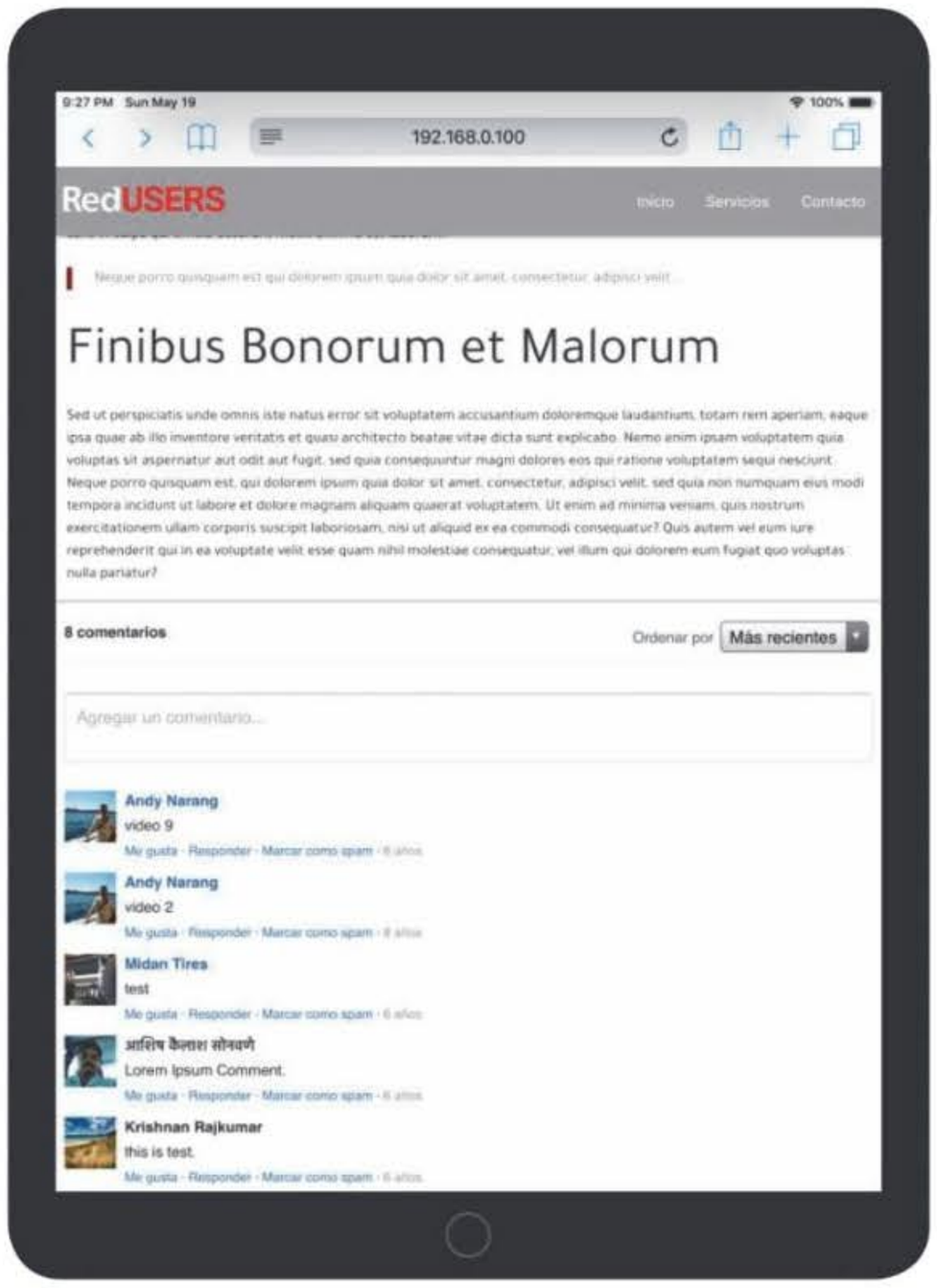
Número de la versión de la API de Facebook que estamos utilizando.

Resultado final de comentarios

Estamos en condiciones de probar el sistema de comentarios funcional. Guardamos el proyecto y ejecutamos el web server para levantar el sitio web desde un dispositivo móvil o emulador. El resultado deberá ser como el siguiente:

Ya debemos observar el sistema de comentarios en el sitio web. Si aparecen





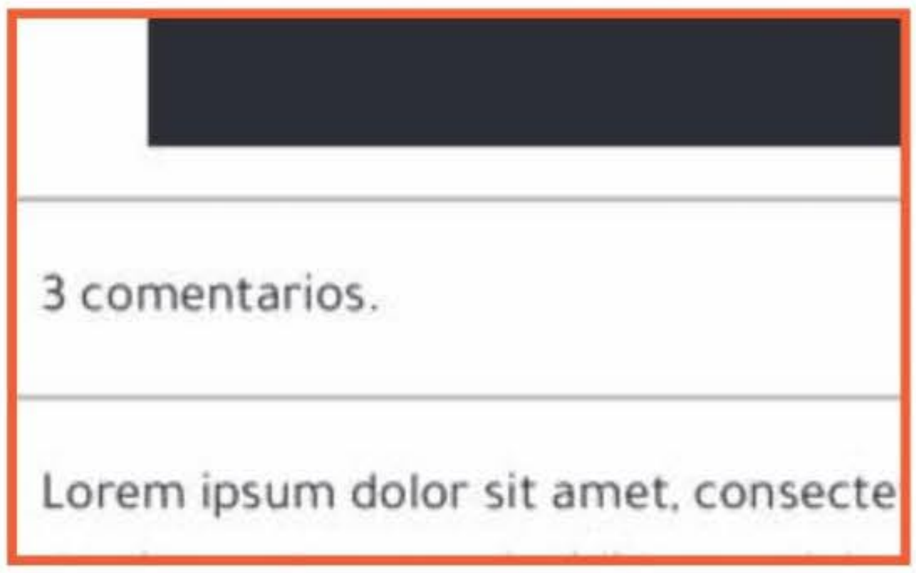
comentarios genéricos, como los que vemos en la imagen anterior, es probable que hayan sido realizados por otros developers que utilizaron nuestra misma URL de prueba. En este caso usamos http://localhost:81/ para visualizar el sitio web con los comentarios asociados.

Total de comentarios al inicio de la publicación

Con una simple línea de código, Facebook permite visualizar un totalizador de comentarios sobre la publicación. El código es el siguiente:

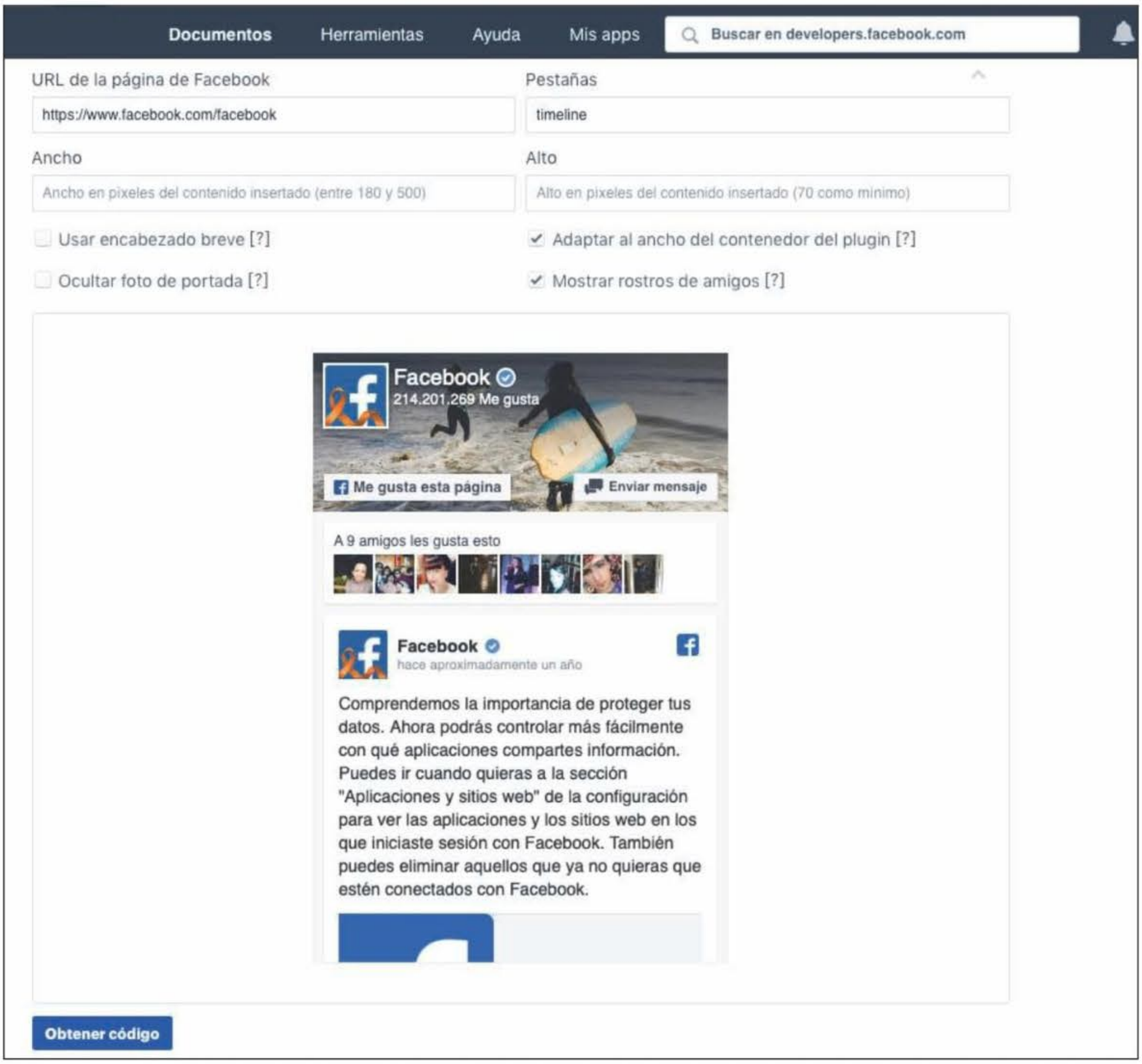
```
<span class="fb-comments-count" data-href="https://localhost:81/"></
span> comentarios.
```

Como vemos, la etiqueta **** invoca a una clase que obtiene el total de comentarios del sitio web indicado en el atributo **data-href**. Finalmente, muestra en pantalla un número, que debemos acompañar por una o más palabras que le den el contexto correcto. Presentamos el resultado en la siguiente imagen:



Plugin de páginas

El plugin de páginas de Facebook permite insertar y promocionar cualquier página pública que esté creada en esta red social. A través de él podremos hacer que visitantes de este sitio web puedan poner **Me gusta** directamente en la página de Facebook sin tener que buscarla manualmente. Para implementar este plugin de manera sencilla y práctica, accedemos a la URL: https://developers.facebook.com/docs/plugins/page-plugin, allí encontraremos un Configurador con opciones.



Este configurador permite generar un banner publicitario de nuestro sitio web en Facebook, que puede ser vertical u horizontal, y tiene una serie de parámetros adicionales para configurar fácilmente. Además de los parámetros estándar especificados también para la sección de comentarios, este plugin adiciona atributos importantes, como los siguientes:

TABLA 3

ATRIBUTO	DESCRIPCIÓN
data-height	La altura del plugin, tomando como valor mínimo 70 px.
data-tabs	Permite mostrar pestañas dentro del banner, como **timeline**, **events** o **messages**. Estas se agregan de forma independiente, separadas por comas.
data-hide-cover	Oculta o muestra la foto de portada del encabezado.
show-facepile	Muestra la foto de perfil de los **amigos** que indicaron que les gusta la página.
data-small-header	Usa el encabezado pequeño si especificamos **true** como valor de este atributo.

Al presionar el botón **Obtener código**, obtendremos un bloque HTML similar al siguiente:

```
<div class="fb-page"
  data-href="https://www.facebook.com/redusers" data-width="340" data-
hide-cover="false" data-show-facepile="false">
</div>
```

Insertamos este bloque de código en el documento HTML que venimos trabajando, debajo de la imagen que acompaña a la publicación ficticia, reemplazando el bloque anterior dedicado al totalizador de comentarios. Nuestro documento HTML, junto con el nuevo bloque de código, debe quedar como se muestra en la imagen de la página siguiente.

Finalmente, guardamos el proyecto y lo ejecutamos en el navegador web, dispositivo móvil o emulador, utilizando por supuesto Fenix Web Server. Tenemos que visualizar el plugin de nuestra página insertado en este documento HTML, sin ningún problema.

Visualizar icono de amigos

Otra alternativa para mejorar la visualización del plugin es agregarle la opción de que, quien entra en la página, pueda ver cuáles de sus amigos le han dado Me gusta previamente. Para lograrlo, no debemos alterar el código anteriormente ingresado, sino solo cambiar el

index.html ●

index.html ▸ html ▸ body ▸ main

```
25          <div id="fb-root"></div>
26          <script async defer crossorigin="anonymous" src="https://connect.facebook.net/es_LA/sdk.js#xfbml=1&
            version=v3.3">
27          </script>
28          <main>
29              <h1>Lorem Newspsum</h1>
30              <div class="imagen">
31                  <img src="images/imagen-indexhtml.jpg" id="imagen" width="80%">
32              </div>
33              <hr/>
34              <div class="center">
35                  <div class="fb-page" data-href="https://www.facebook.com/redusers" data-width="500"
36                  data-hide-cover="false" data-show-facepile="true"></div>
37              </div>
38              <hr/>
39              <p>Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut
                labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris
                nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit
                esse cillum dolore eu fugiat nulla pariatur. Excepteur sint occaecat cupidatat non proident, sunt
                in culpa qui officia deserunt mollit anim id est laborum.</p>
40              <blockquote>Neque porro quisquam est qui dolorem ipsum quia dolor sit amet, consectetur, adipisci
                velit...</blockquote>
41              <h2>Finibus Bonorum et Malorum</h2>
```

valor del atributo **facepile=”false”** por **true**. Modifiquemos entonces el código y refresquemos la página en nuestro dispositivo móvil, navegador o emulador, para ver los resultados.

GUÍA VISUAL 3

Cuando ejecutamos la página en modo de prueba, cuando estamos deslogueados de Facebook o cuando ningún amigo le ha dado previamente ***Me gusta*** *a dicho sitio, veremos el plugin de esta forma.*

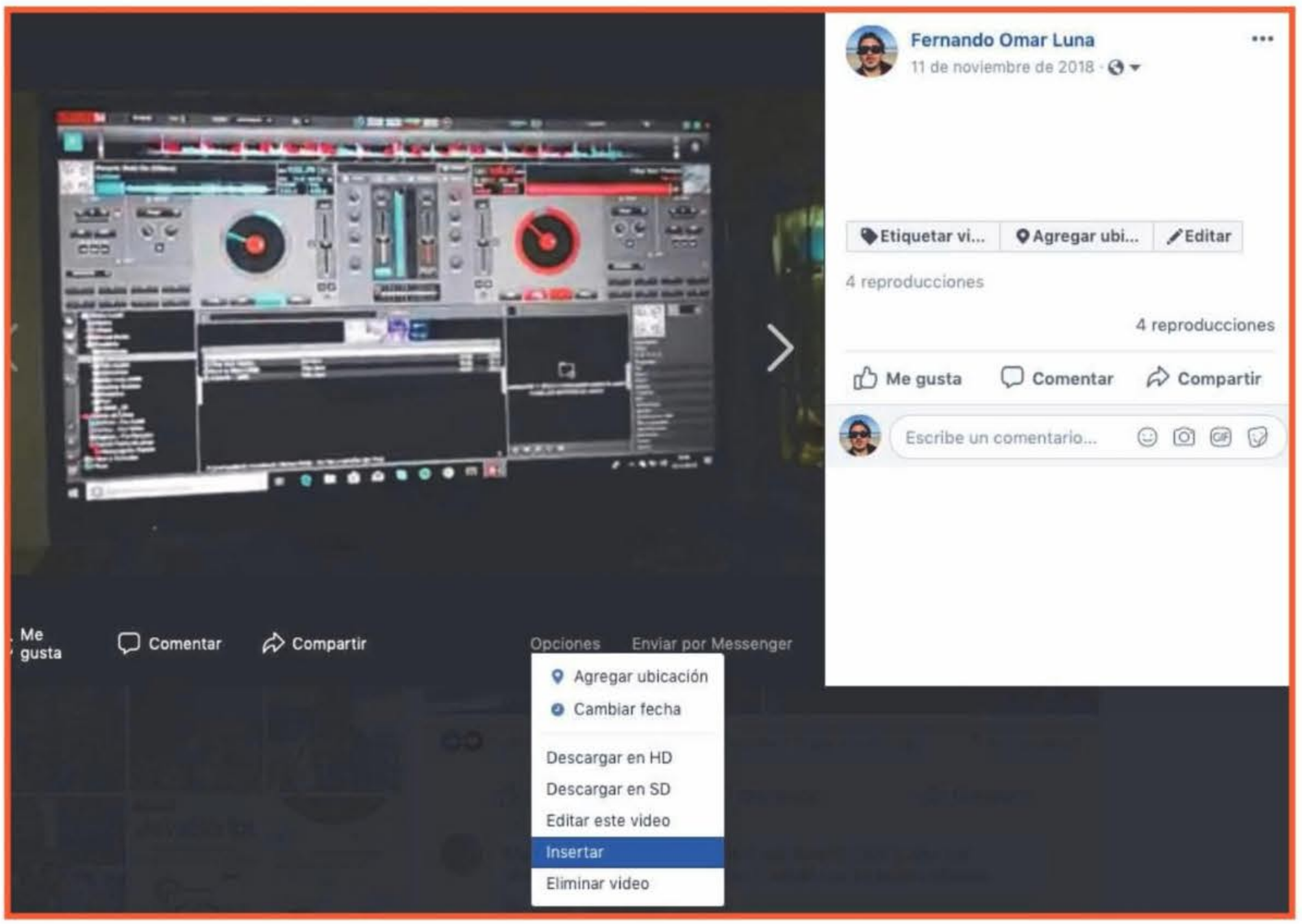
Si uno o más amigos nuestros han dado clic en ***Me gusta*** *previamente a la página en cuestión, veremos el plugin con este otro estilo (siempre y cuando estemos logueados en Facebook)..*

Cambiar el idioma

Si aplicamos el cambio de idioma en el bloque de código que iniciailiza la API de Facebook, tal como vimos en el ejercicio anterior dedicado a los comentarios, también se generará un impacto en el plugin de páginas; tengamos precaución al hacerlo. Facebook priorizará el idioma establecido en el plugin por sobre el elegido en la página de los clientes.

Compartir videos de Facebook

Si el sitio requiere mucho contenido multimedia, podemos sacar partido de nuestra página de Facebook, subiendo los videos a ella y luego insertándolos por medio de **Insertar Video** (Embed Video). Esto dará un doble beneficio: primero, ahorramos espacio de almacenamiento en nuestro hosting web; segundo, desde el video insertado en la web daremos a conocer nuestro sitio en Facebook. Veamos un ejemplo a continuación:



En el sitio web de Facebook, ubicamos un video nuestro o de alguna página pública que lo tenga compartido de forma Global. Entramos a la publicación del video y pulsamos ***Opciones****; se desplegará un menú del cual debemos seleccionar* ***Insertar*** *o* ***Embed Video****.*

01

Se abrirá una ventana emergente, que muestra un código iniciado por el tag ***<iframe>****. Este crea un marco dentro de la página web para mostrar contenido adicional o remoto. Marcamos* ***Incluir publicación completa*** *y, luego, copiamos todo el bloque de código. En el documento HTML de ejemplo, creamos un* *** *** *con la clase* ***center****, y pegamos el bloque de código copiado después de la etiqueta* ***<h2>*** *(subtítulo). Guardamos el documento HTML y cargamos la página en un navegador web o emulador.*

02

Si todo fue bien, veremos el video de Facebook insertado dentro de nuestro documento HTML. Este mostrará los controles propios del reproductor de video de la plataforma y, en nuestro perfil social, podremos sumar más reproducciones desde un sitio web externo.

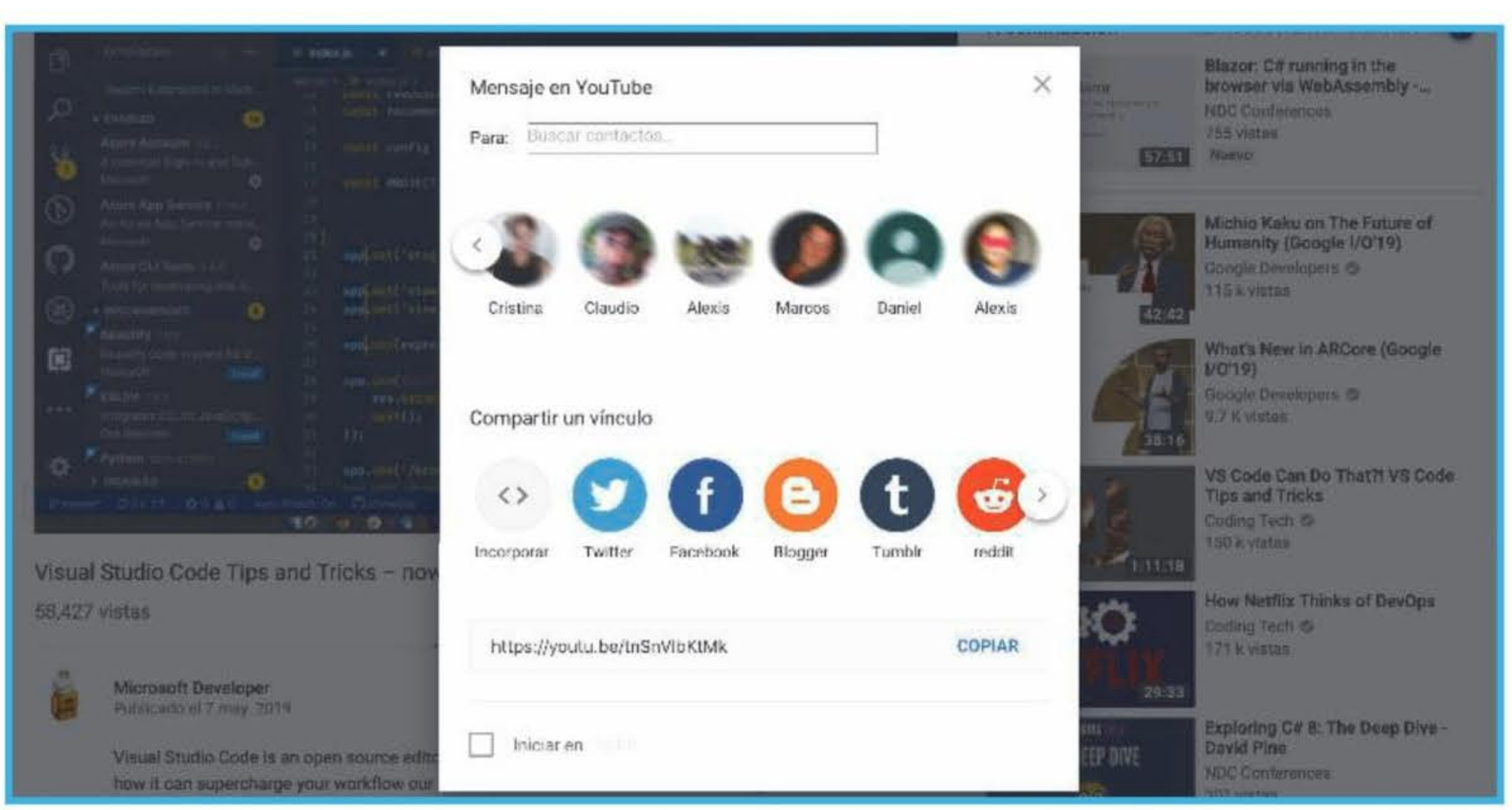
03

Siempre que insertemos videos en sitios web que desarrollamos, tengamos la precaución de mantenerlos compartidos en modo Público. Si cambiamos este modo, puede que nuestra web muestre un mensaje de error como el de esta imagen.

04

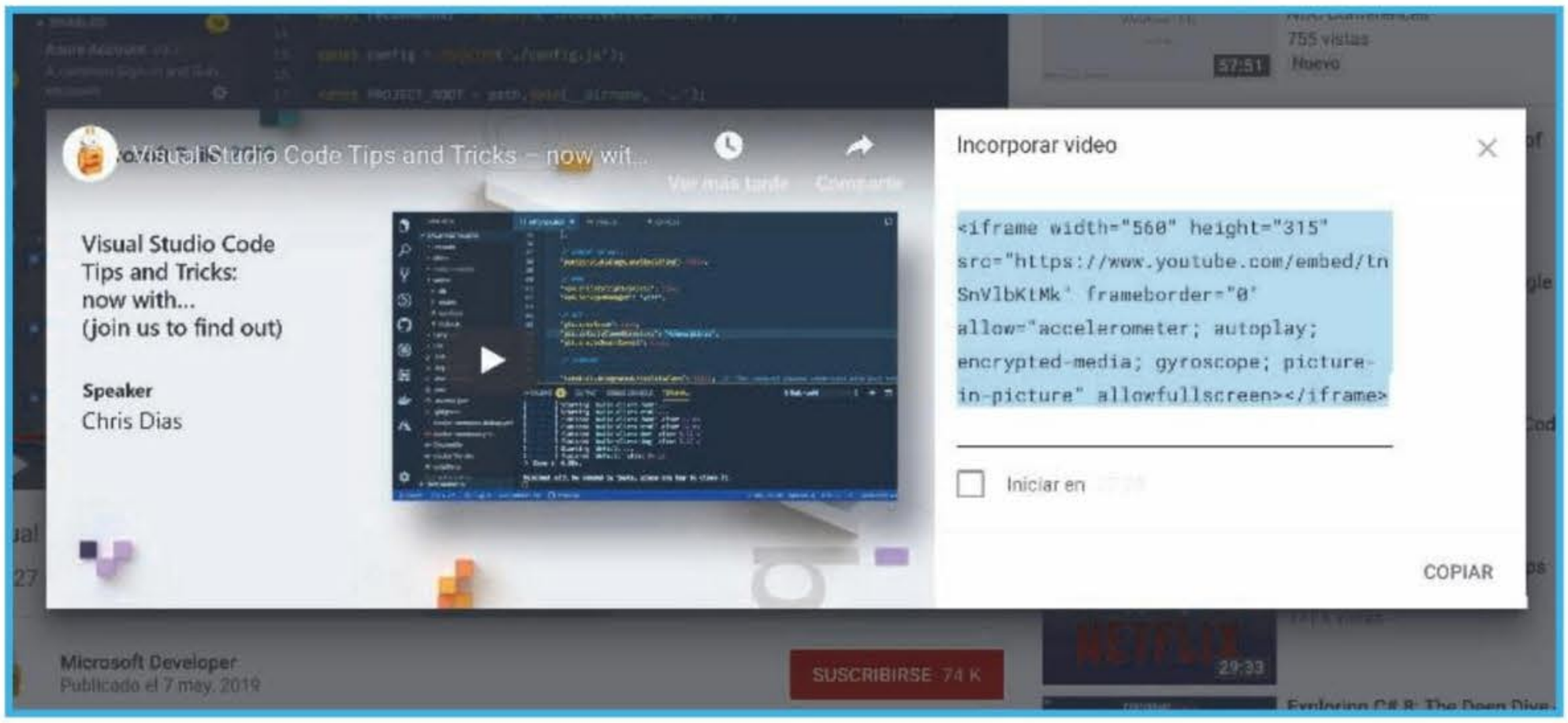
YouTube: cómo embeber videos

Integrar un video de **YouTube** en nuestra web es muy fácil. Lo primero que debemos hacer, es acceder al portal de streaming, ubicar el video que deseamos y pulsar Play para comenzar a reproducirlo.



*Ya con el video en reproducción, presionamos sobre el icono **Compartir** (representado por una flecha hacia la derecha), ubicado usualmente al pie del video, sobre el extremo derecho. Se abrirá una ventana emergente con varias opciones. Pulsamos sobre el icono **< >**, en el apartado **Compartir un vínculo**.*

01



A continuación, se abrirá una nueva ventana emergente que contiene el video en cuestión y, sobre el extremo derecho, un bloque de código HTML, iniciado también con el tag ***<iframe>****. Hacemos clic sobre este bloque de código y luego presionamos la combinación de teclas* ***Ctrl+A*** *para seleccionar todo el código, y* ***Ctrl+C*** *para copiarlo.*

02

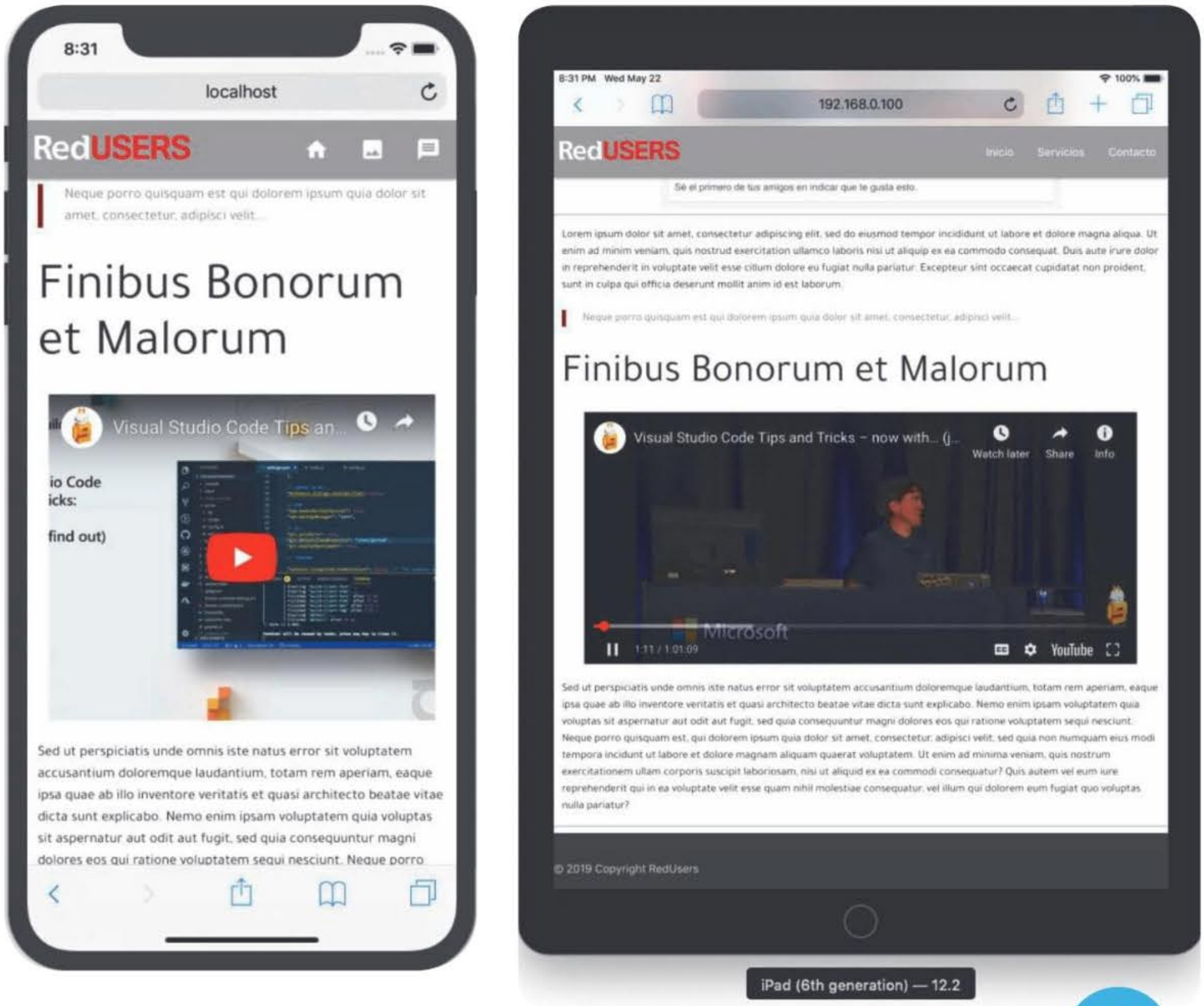
index.html

index.html ▸ html ▸ body ▸ main ▸ div.center

```
        cupidatat non proident, sunt in culpa qui officia deserunt mollit anim id
        est laborum.</p>
43      <blockquote>Neque porro quisquam est qui dolorem ipsum quia dolor sit amet,
         consectetur, adipisci velit...</blockquote>
44      <h2>Finibus Bonorum et Malorum</h2>
45      <div class="center">
46          <iframe width="560" height="315"
47              src="https://www.youtube.com/embed/tnSnVlbKtMk" frameborder="0"
48              allow="accelerometer; autoplay; encrypted-media; gyroscope;
                picture-in-picture" allowfullscreen>
49          </iframe>
50      </div>
51      <p>Sed ut perspiciatis unde omnis iste natus error sit voluptatem
        accusantium doloremque laudantium, totam rem aperiam, eaque ipsa quae ab
        illo inventore veritatis et quasi architecto beatae vitae dicta sunt
        explicabo. Nemo enim ipsam voluptatem quia voluptas sit aspernatur aut
        odit aut fugit, sed quia consequuntur magni dolores eos qui ratione
```

Ahora abrimos Visual Studio Code para editar el archivo ***index.html*** *de nuestro último proyecto. Vamos a eliminar de él el bloque de código que insertaba el video de Facebook y a reemplazarlo por este bloque de código copiado recientemente. El bloque de código insertado debe quedar como en la imagen.*

03

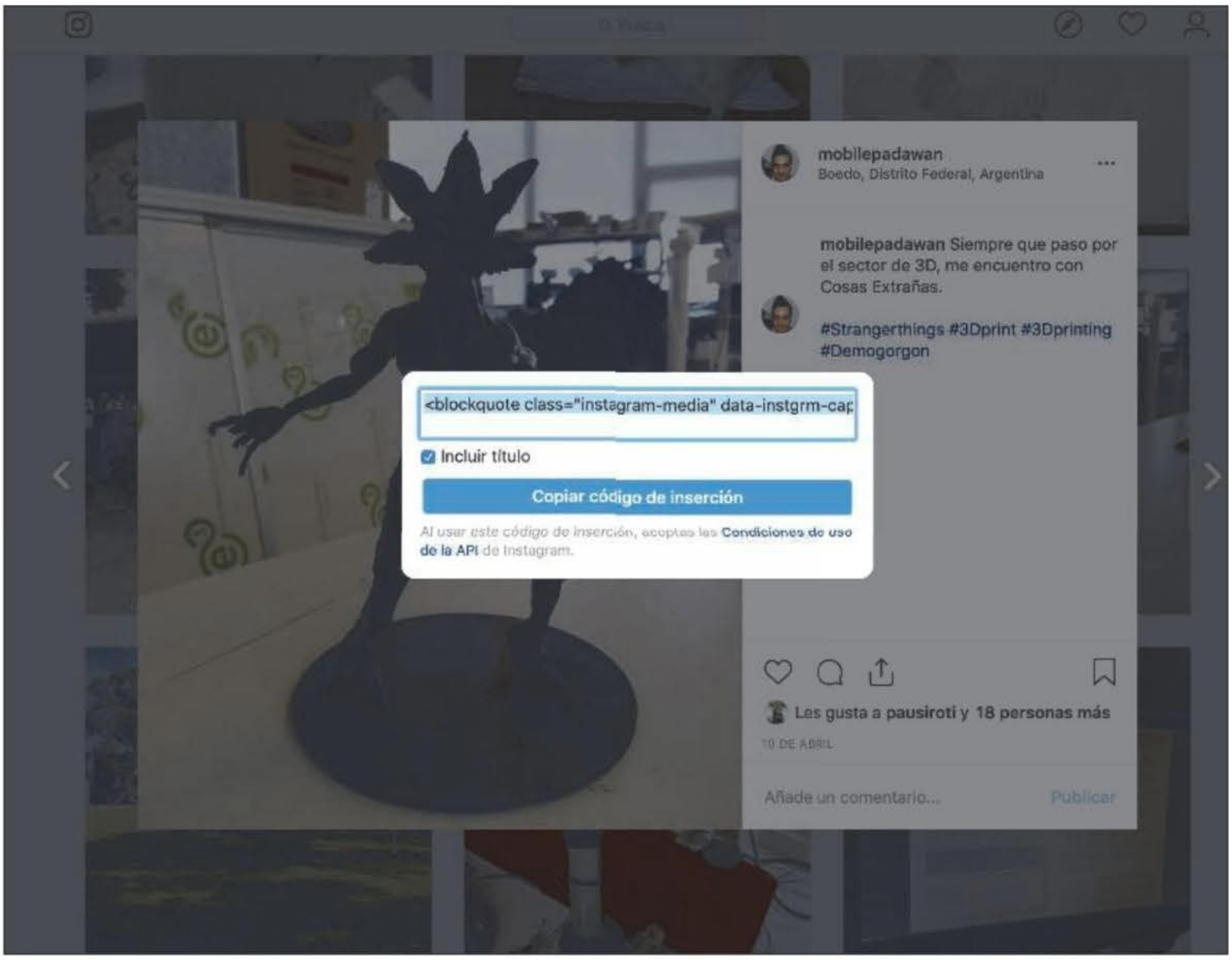


04

Por último, podemos guardar el proyecto y probarlo en un dispositivo móvil. Para adaptar automáticamente el tamaño del video, recomendamos modificar el valor de píxeles en width por 90%. De esta manera, el video ocupará el ancho de la pantalla sin que tengamos que complicarnos con cálculos.

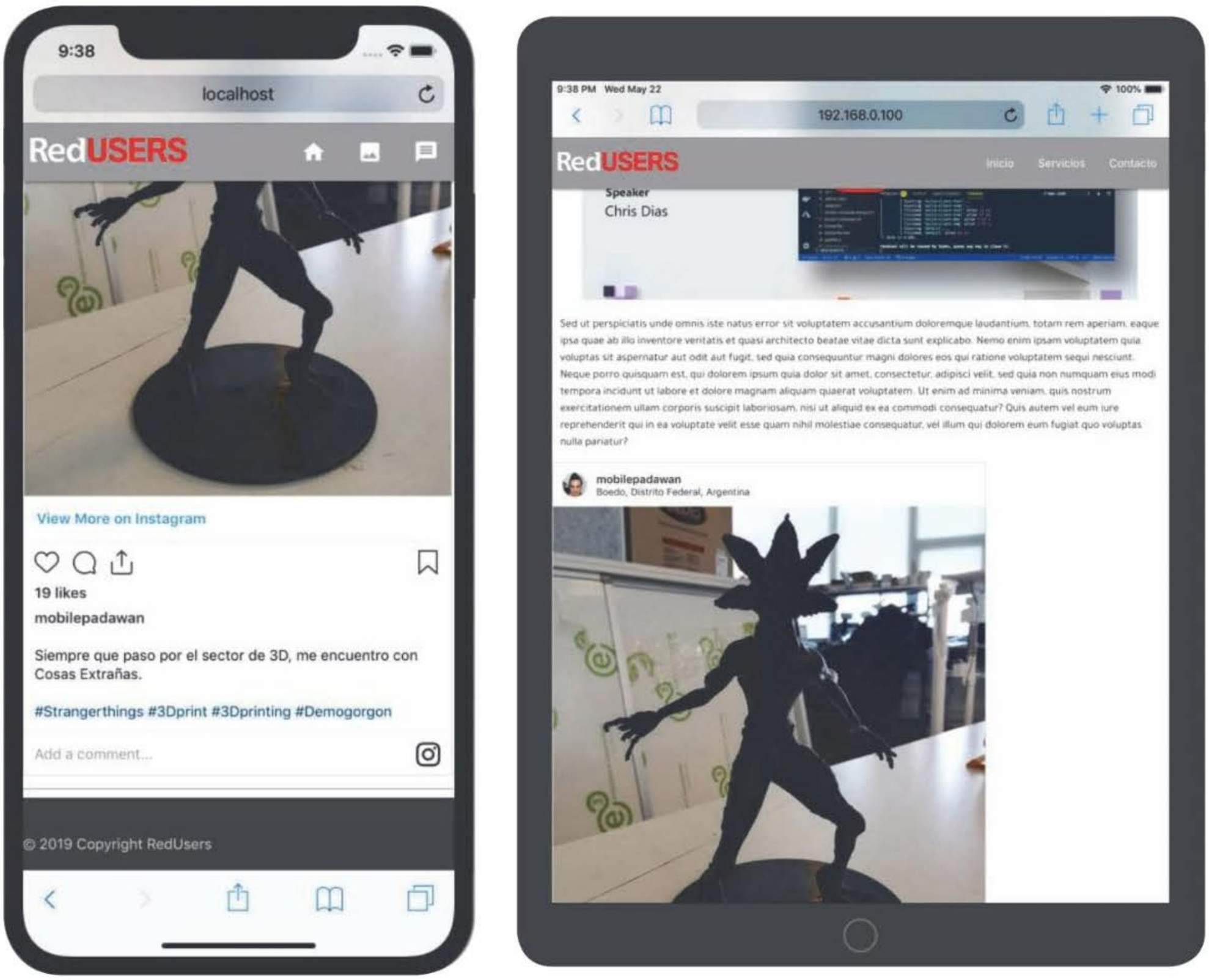
Insertar publicación de Instagram

Tal como vimos hasta aquí las diferentes opciones para agregar contenido social en nuestras webs, **Instagram** también dispone de un sistema similar para compartir publicaciones, con la posibilidad de insertarlas en diferentes sitios. Para hacer esto, ingresamos primero en esta red social; luego, ubicamos alguna publicación que deseamos utilizar y accedemos a ella. Lo ideal será realizar estos pasos desde la computadora y no, desde un dispositivo móvil. Pulsamos el botón **...**, ubicado en el extremo superior derecho de la ventana emergente. En el menú de opciones que se abre, seleccionamos **Código de inserción**. Por último, en la nueva ventana emergente que aparece, se mostrará el bloque de código para copiar. Pulsamos sobre el botón correspondiente y cerramos las ventanas de Instagram.



Pegamos el código copiado en nuestro documento HTML. Vamos a usar el proyecto en el cual venimos trabajando, de modo que pegamos el código al final del último párrafo de ejemplo, justo antes del cierre del tag **</main>**. No debemos asustarnos, porque el código copiado parecerá demasiado extenso. ¿El motivo? Instagram se ocupa de enviar, junto con el bloque de código correspondiente a la publicación que deseamos insertar, su propio código CSS y también su código JavaScript.

Como vemos en la siguiente imagen, el resultado es el esperado, y hasta podemos comentar desde allí y darle **Like** a la imagen siempre que estemos logueados en Instagram. Si no es así, se abrirá la publicación en la app o webapp. Solo nos queda incluir este código en un ** ** centrado, y así la página web quedará optimizada.



Integrar contenido de Twitter

Finalmente, la red social del pajarito también nos brinda herramientas para aprovechar su contenido. En la URL https://developer.twitter.com/en/docs, hallaremos las herramientas que podemos aprovechar en este caso. Entre las opciones más habituales, podemos insertar timelines, momentos, tuits, un botón de tuitear, y un botón para seguir a un usuario, entre otras tantas opciones.

Veamos a continuación el código para insertar un timeline de Twitter en nuestro website:

```
<a class="twitter-timeline"
href="https://twitter.com/mobilepadawan?ref_src=twsrc%5Etfw">Tweets de
mobilepadawan</a>
<script async src="https://platform.twitter.com/widgets.js"
charset="utf-8"></script>
```

Como podemos apreciar, el código simplemente contiene un tag **<a>** junto con una URL, en la que especificamos el usuario cuyo timeline deseamos obtener. Luego existe un script JS, que llama al archivo JavaScript correspondiente, para que transforme este hipervínculo en un verdadero timeline. El resultado de este código se muestra en la siguiente figura:

Recursos sociales

Los archivos **social-base.zip**, **Social-plugin.zip**, **Social-twitter.zip**, **Social-video.zip** y **Social-Youtube-instagram.zip** contienen los ejercicios resueltos de todo lo desarrollado a lo largo de este último capítulo. Pueden descargarlos para comparar su código o mejorar lo desarrollado con otras funcionalidades ofrecidas por el ecosistema social para desarrolladores de cada plataforma.

RESUMEN

Con lo aprendido a lo largo de este e-book, ya estamos en condiciones de desarrollar una web con todo el look and feel del **Diseño Material**, tan popular en estos últimos años. Aprendimos también a integrarnos con redes sociales, y a sacar provecho de los formularios de contacto y del audio y video, elementos muy demandados en la actualidad.

Aun así, nos queda un largo camino por recorrer. En las próximas ediciones de este e-book, incrementaremos más nuestros conocimiento sobre Materialize CSS, y seguiremos incorporando todo el know how necesario para convertirnos en verdaderos desarrolladores web que profesan los paradigmas **mobile-first** y **offline-first**.

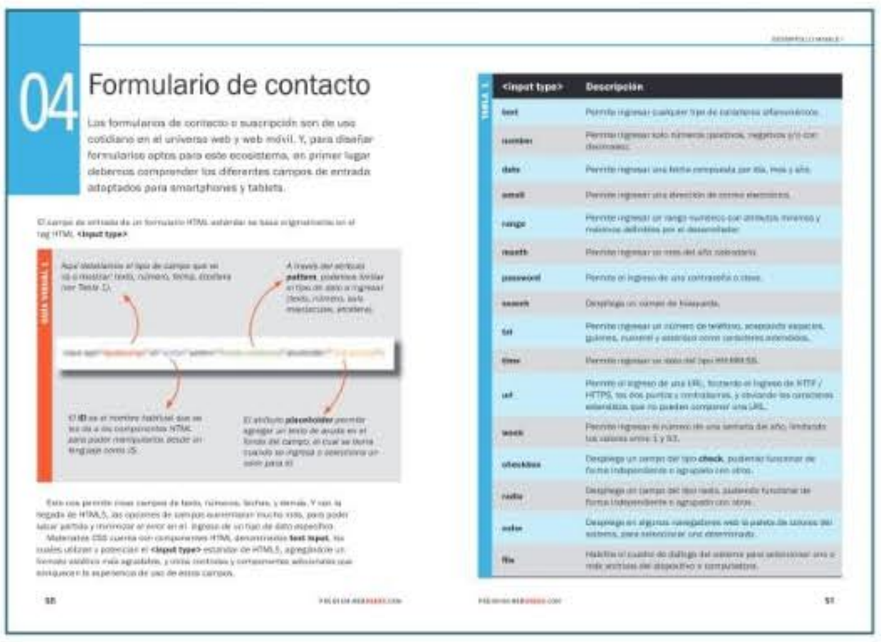
USERS

Desarrollo Mobile

ACERCA DE ESTE CURSO

Este curso nos enseñará a programar aplicaciones web bajo la filosofía **Mobile First** utilizando un framework basado en **Material Design**. Abordaremos desde un diseño básico que permite crear una web estática, e iremos escalando en componentes y conocimientos hasta crear una web 100% dinámica. Finalmente aprovecharemos características de los dispositivos móviles y convertiremos nuestra web en una **PWA**, apta para el ecosistema **Android** así como también para **iOS** y **iPadOS**.

EL VOLUMEN I



En este primer e-book de la colección, conoceremos **Materialize CSS**, y comenzaremos a ver cómo implementar la barra de navegación, los títulos, los párrafos y las imágenes en el diseño de un sitio web adaptado para móviles. Luego profundizaremos en la manera de integrar audio y video en nuestros desarrollos, y en la forma de sumar contenido desde y hacia las diferentes redes sociales más populares a la fecha.

SOBRE EL AUTOR

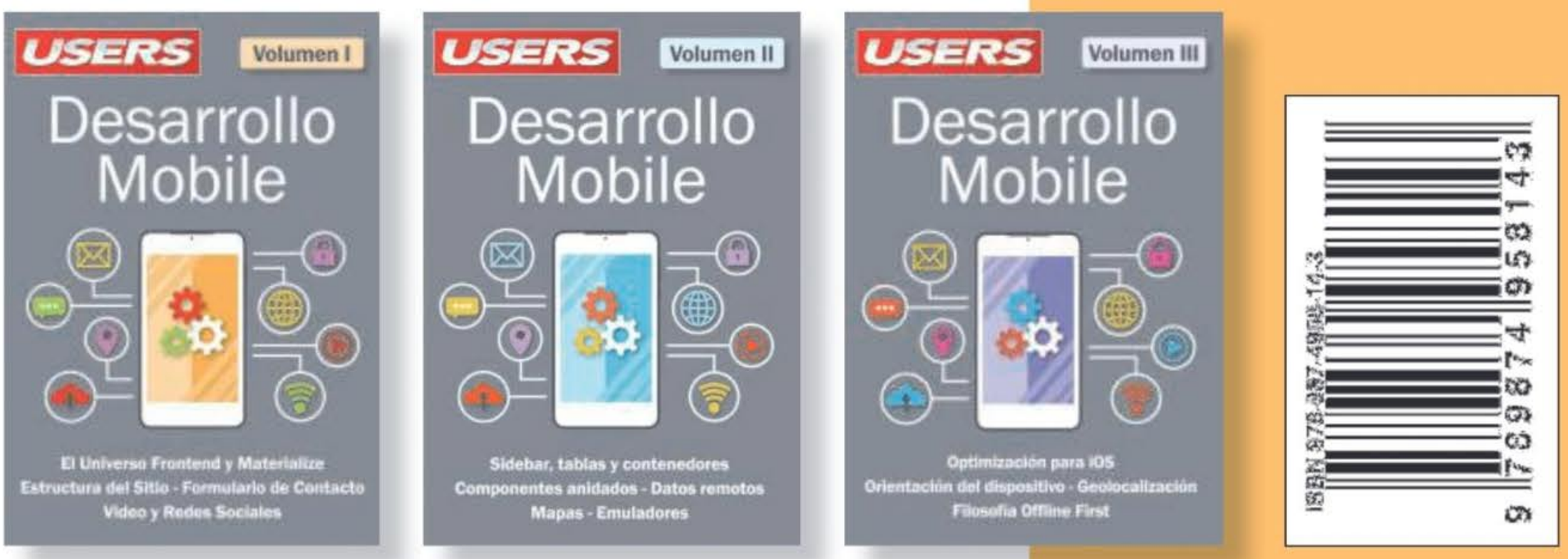
Fernando Luna es Analista de Sistemas y Technical Writer. Lleva más de 25 años desarrollando software para diferentes plataformas, y algo más de una década colaborando como autor de publicaciones técnicas orientadas a la programación y las nuevas tecnologías. Su pasión se divide entre la programación, la electrónica y la educación.

REDUSERS.PREMIUM.COM

RedUSERS PREMIUM es la biblioteca digital de USERS, desde donde podrán acceder a cientos de publicaciones: informes, e-books, guías, revistas y cursos. Todo el contenido está disponible online y offline, y para cualquier dispositivo. Publicamos una novedad, al menos, cada siete días.

REDUSERS.COM

En nuestro sitio podrán encontrar noticias relacionadas y participar de la comunidad de tecnología más importante de América Latina.



ISBN 978-987-4958143