

DESARROLLO DE APLICACIONES WEB



PROYECTO 1

Autores:

Brais Míguez Varela

Leticia Martínez Limeres

1 de abril de 2023

Índice

1. Introducción	2
2. HTML	7
3. Estructura de ficheros	11
4. CSS	12
4.1. General	12
4.2. Inicio	13
4.3. About	15
4.4. Portfolio	17
4.5. Subpáginas de portfolio	19
4.6. Contacto	21
4.7. Responsive	23
4.8. Estructura de ficheros	23
5. JavaScript	25
5.1. Métodos de acceso al DOM utilizados	25
5.2. Eventos utilizados	25
5.2.1. Evento <code>click</code>	25
5.2.2. Evento <code>scroll</code>	27
5.2.3. Eventos <code>mouseenter</code> y <code>mouseout</code>	29
5.3. jQuery	30
5.4. JES6	34
5.5. Expresiones regulares	34
5.6. Carga de contenido mediante servidor Apache	36
5.6.1. XML	36
5.6.2. JSON	37
6. Estructura ficheros final	39

1. Introducción

Se ha elegido una empresa de fotografía para diseñar un sitio web. Esta tiene como objetivo presentar los servicios de nuestra empresa ficticia “Fotografías Lemi” a los clientes.

En un primer momento, al pensar en una página del ámbito fotográfico las ideas que se asoman a nuestras mentes son imágenes de alta calidad, arte y creatividad, entre otros términos como los de la figura 1. Es por ello que el diseño del sitio web debe reflejar estos valores y transmitirlos al usuario.



Figura 1: Inventario de Contenido

Su estructura se dividirá en diferentes secciones para poder mostrar los servicios ofrecidos por la empresa, los miembros del equipo y las opiniones de otros clientes, así como una galería de fotos para que los visitantes puedan apreciar la calidad de nuestro trabajo. Además, se incluirán enlaces a nuestras redes sociales para que los usuarios puedan conocer más sobre nuestra empresa y un formulario de contacto para que los clientes puedan realizar consultas y solicitar presupuestos personalizados. De este modo, el diseño base de la página web es el que se muestra en los bocetos siguientes.

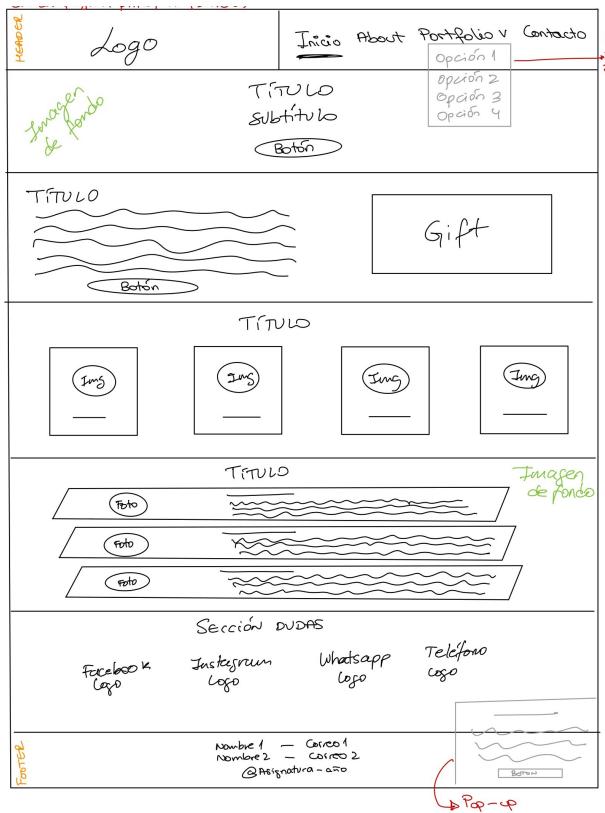


Figura 2: Sketch página *Inicio*

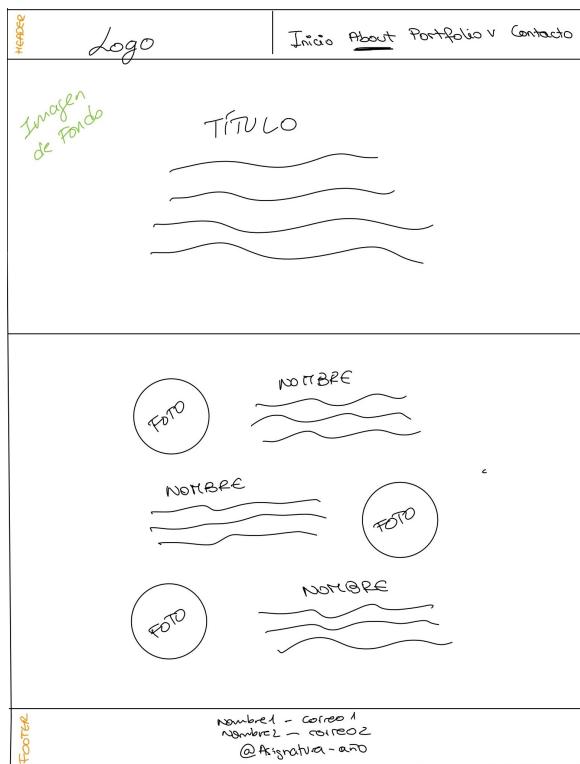


Figura 3: Sketch página *About*

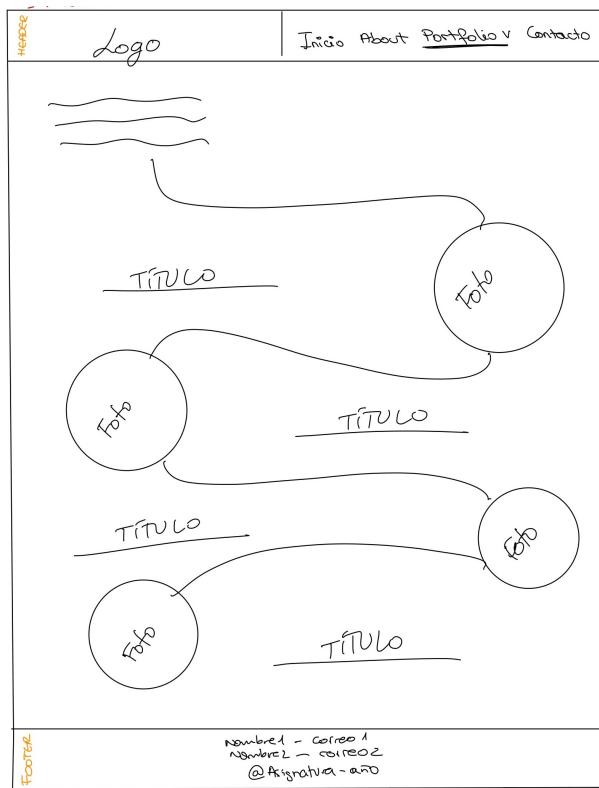


Figura 4: Sketch página *Portfolio*

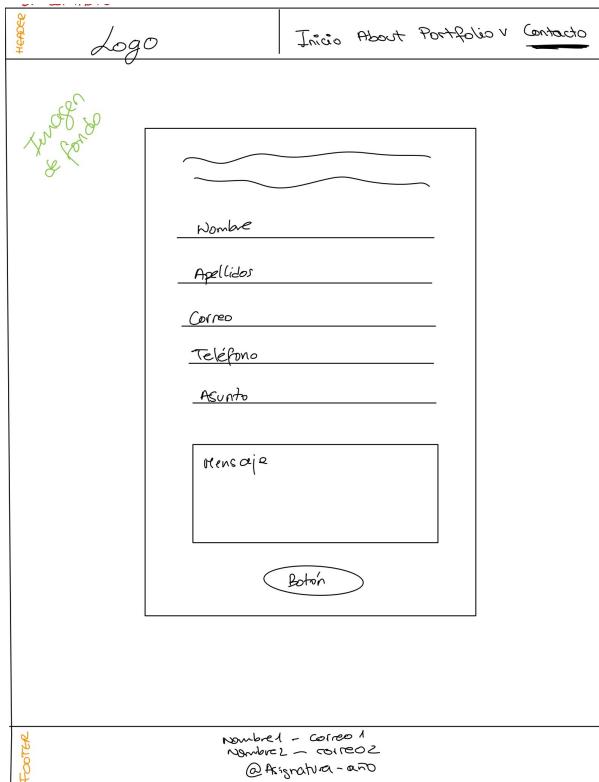


Figura 5: Sketch página *Contacto*

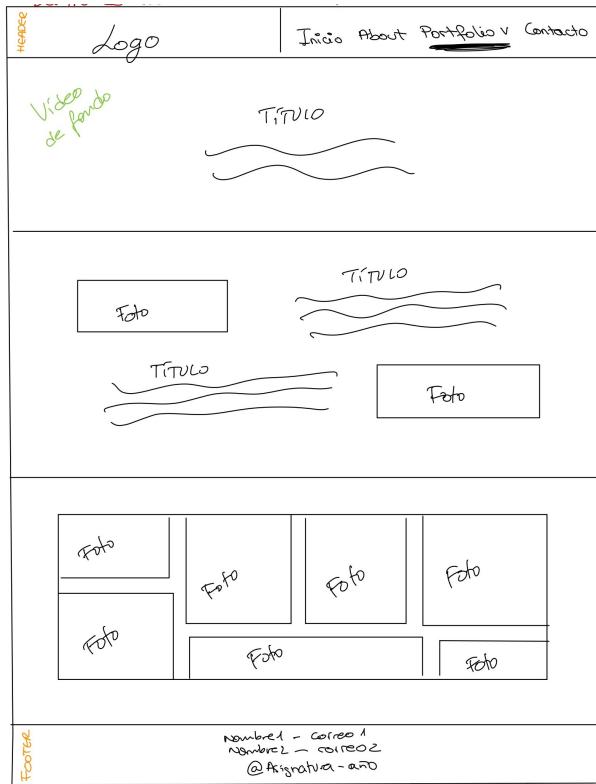


Figura 6: Sketch subpáginas dentro del *Portfolio*

Una vez se han generado estos diseños, ya es posible crearse una idea mental básica de a lo que se quiere llegar, que se irá remodelando con la combinación de tres herramientas fundamentales: HTML, CSS y JavaScript.

HTML se utiliza para crear la estructura y el contenido de la página web. CSS se emplea para darle estilo y diseño al sitio web, permitiendo cambiar el aspecto visual de los elementos HTML, como el color, la tipografía, el tamaño y la posición. Finalmente, JavaScript se utiliza para agregar interactividad y dinamismo al sitio web. Permite a los usuarios interactuar con los elementos de la página como los botones, formularios y menús, y hace posible que el entorno web responda a diferentes eventos, como clicks de ratón, desplazamientos...

Por lo tanto, con los tres recursos fundamentales mencionados, es posible crear una web completamente funcional y atractiva que cumpla con los objetivos propuestos inicialmente para la página web de fotografía. De esta manera, el sitio web será capaz de proporcionar una experiencia satisfactoria para el usuario visitante, permitiéndole realizar diversas acciones, como visualizar la información o acceder a las redes sociales (figura 7).

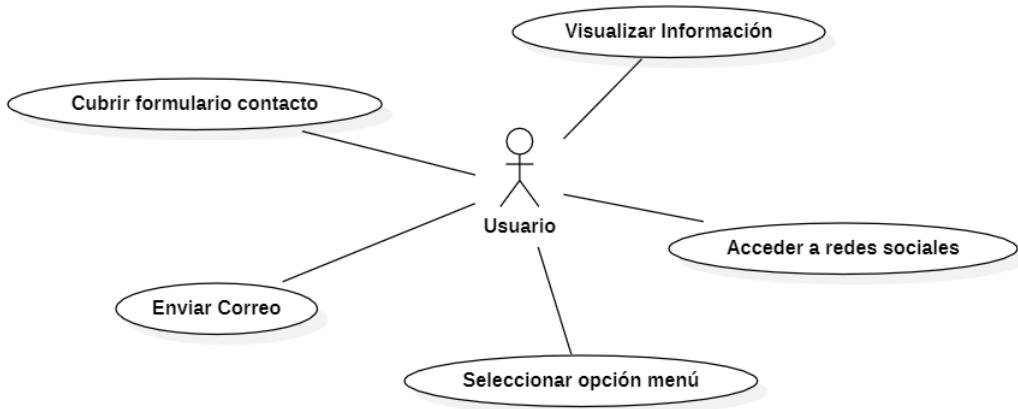


Figura 7: Casos de uso

Ejecutando cualquiera de los casos de uso anteriores, se hace posible interaccionar de forma dinámica con la web. Algunas acciones que se pueden realizar son las que se muestran en la figura 8.

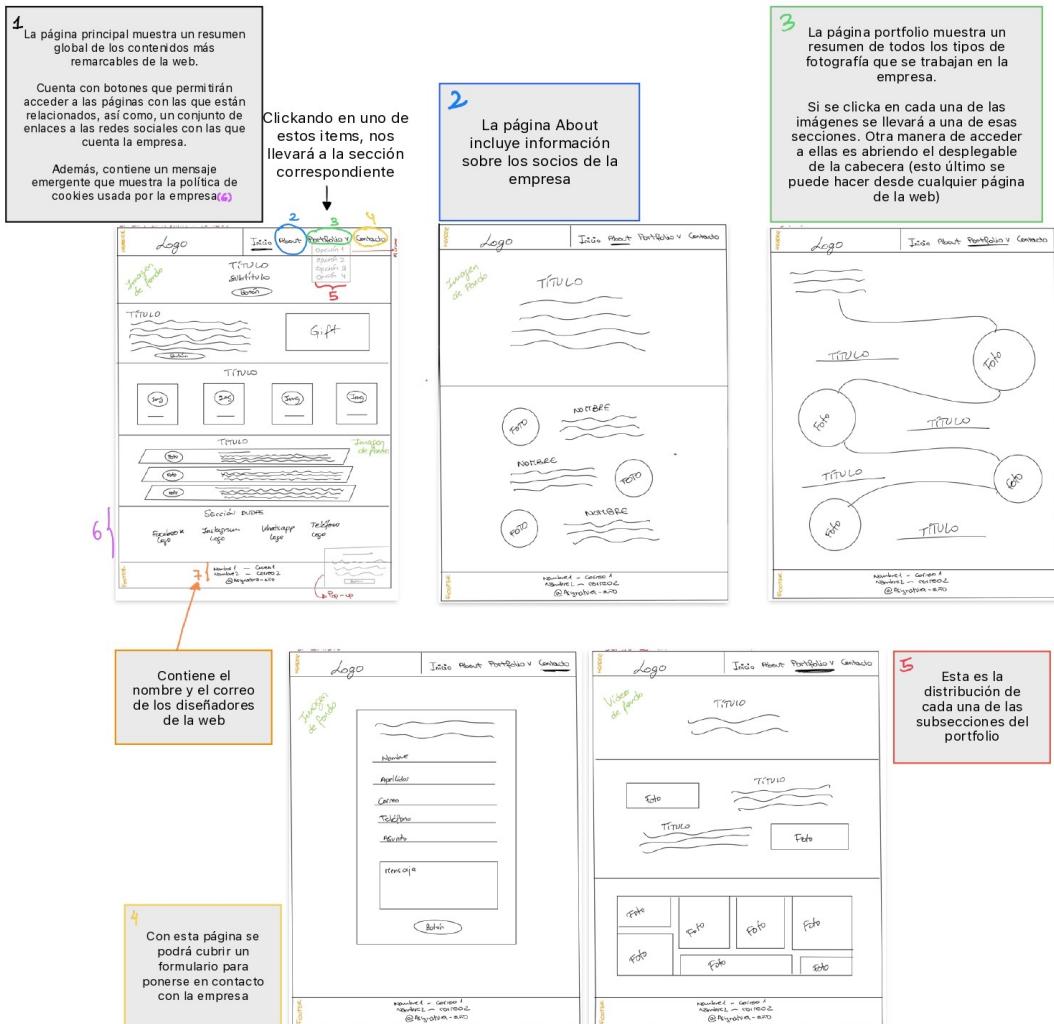


Figura 8: Storyboard resumen de la dinámica del sitio web

Adicionalmente, es esencial establecer una estructura de archivos para tener una visión clara de cómo se organizarán todos los ficheros relacionados con la página web. En nuestro caso, hemos decidido generar cinco carpetas en el proyecto, que son: fonts, images, html, css y js. En estas carpetas se irán añadiendo los archivos correspondientes a medida que se avanza en el proyecto. Esta estructura ayudará a mantener los documentos organizados y a facilitar su gestión. Además, seguir una estructura coherente permitirá a personas ajenas entender el proyecto de manera más eficiente.

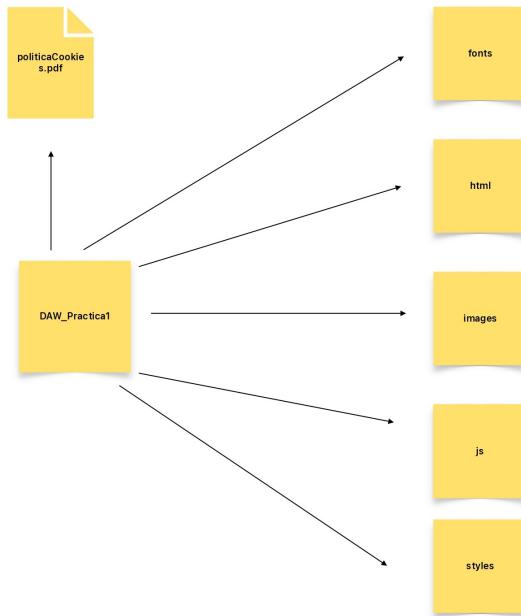


Figura 9: Estructura inicial de ficheros

2. HTML

HTML es un lenguaje de marcado de texto que ha sido fundamental para la creación de páginas web desde su aparición. En la actualidad, sigue siendo una herramienta esencial para el diseño de páginas web, debido a su compatibilidad con otros lenguajes de programación y a su capacidad para crear sitios web accesibles y optimizados para motores de búsqueda.

En este proyecto, todas las páginas están programadas usando la última versión de HTML, HTML5, que ofrece nuevas funcionalidades y mejoras. Para establecer esta versión, se utiliza la instrucción: `<!DOCTYPE html>`

A continuación, se presentarán los aspectos básicos de HTML, incluyendo sus elementos esenciales, como etiquetas, atributos y elementos de estructura utilizados.

Todo código HTML está formado por dos secciones fundamentales: `<head>` y `<body>`. Ambas son esenciales para la construcción de páginas web con este lenguaje.

La etiqueta `<head>` se utiliza para incluir información importante pero no visible en la página web. En la figura 11 se muestra un ejemplo de esta etiqueta en uno de los archivos HTML del proyecto. Dicha sección contiene diferentes etiquetas y atributos que proporcionan información relevante acerca de la página web y su apariencia.

↪ `<title>Fotografías Lemi</title>`

Define el título de la página web que se mostrará en la pestaña del navegador y en los resultados de búsqueda.

↪ `<meta charset="UTF-8">`

Especifica el conjunto de caracteres que se utilizarán para interpretar los caracteres especiales del documento. En este caso, se utiliza el conjunto de caracteres UTF-8, que admite la mayoría de los idiomas del mundo.

↪ `<meta name="viewport" content="width=device-width, initial-scale=1.0">`

Define la configuración de la ventana de visualización en dispositivos móviles, asegurando que el ancho del contenido se ajuste al ancho de la pantalla y que la escala inicial sea del 100 %.

↪ `<link rel="stylesheet" href="">`

Crea una conexión con un archivo CSS externo que contiene reglas de estilo específicas para la página web.

↪ El resto de etiquetas `<meta>`

Define el color de fondo que se mostrará en la barra de herramientas del navegador de dispositivos y sistemas.

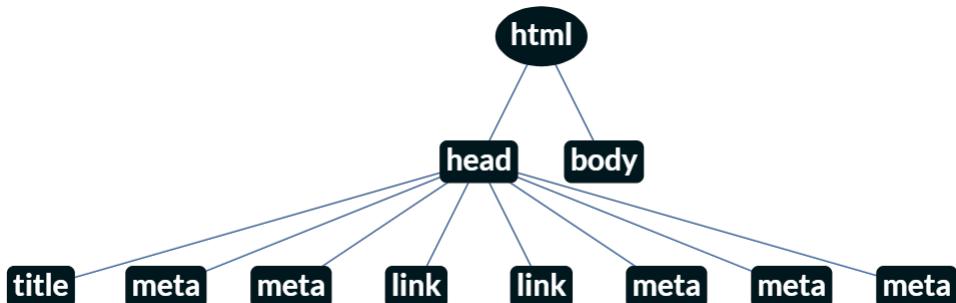


Figura 10: Mapa de Etiquetas de la parte `<head>`

```
<head>
  <title>Fotografías Lemi</title>
  <meta charset="UTF-8">
  <meta name="viewport" content="width=device-width, initial-scale=1.0">
  <link rel="stylesheet" href="../styles/generalStyle.css">
  <link rel="stylesheet" href="../styles/mainStyle.css">

  <!-- Estas líneas son para cambiar el color del navegador en android y ios. -->
  <!-- Ni en google chrome ni en Firefox se ve afectado -->
  <meta name="theme-color" content="#f4ede6">
  <meta name="msapplication-navbutton-color" content="#f4ede6">
  <meta name="apple-mobile-web-app-status-bar-style" content="#f4ede6">
</head>
```

Figura 11: Código de la parte `<head>`

En cambio, la etiqueta que `<body>` se usa para incluir todo el contenido visible en la página. A continuación, se muestra los mapas de etiquetas utilizados en la sección `<body>` en cada una de las páginas más representativas de la web.

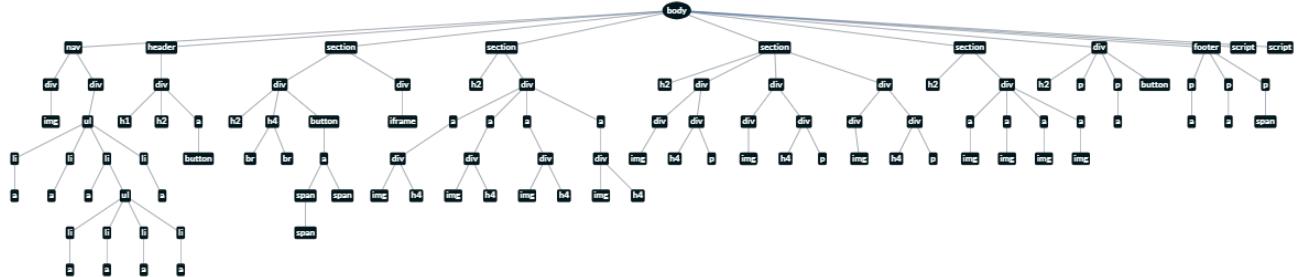


Figura 12: Mapa de Etiquetas de la página *Inicio*

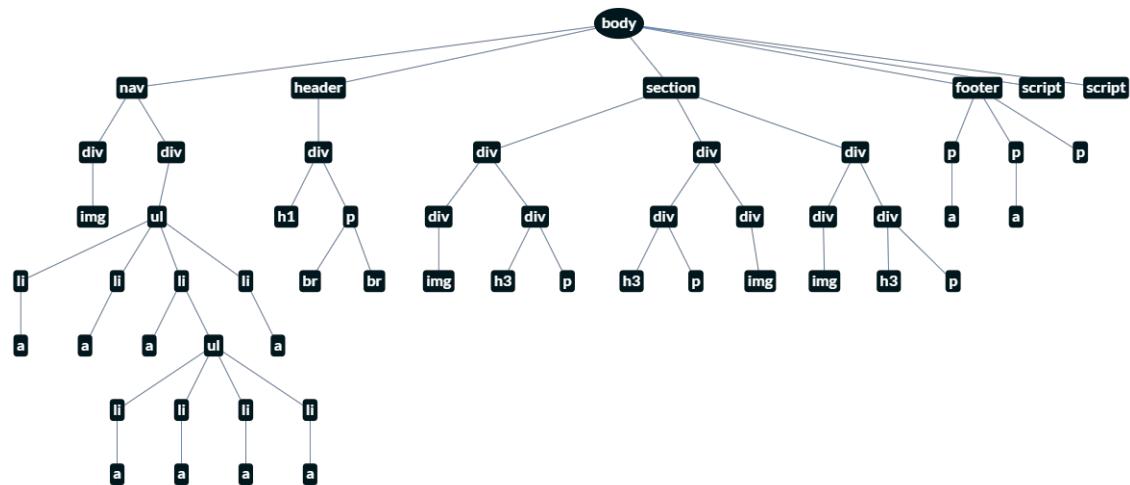


Figura 13: Mapa de Etiquetas de la página *About*

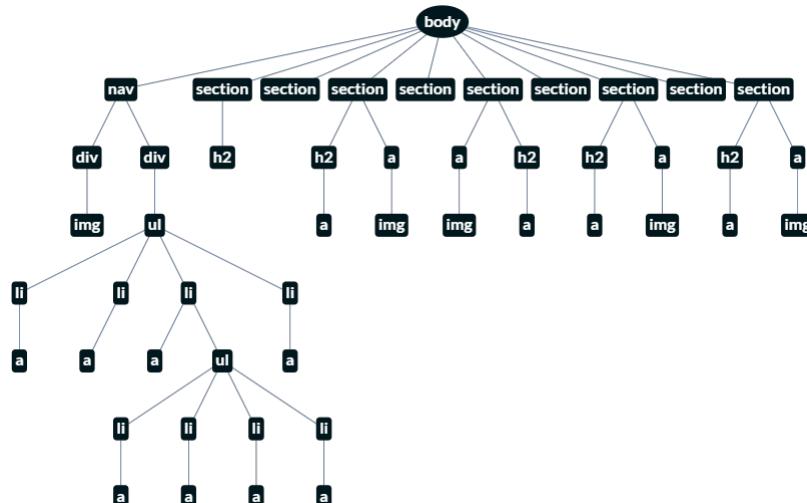


Figura 14: Mapa de Etiquetas de la página *Portfolio*

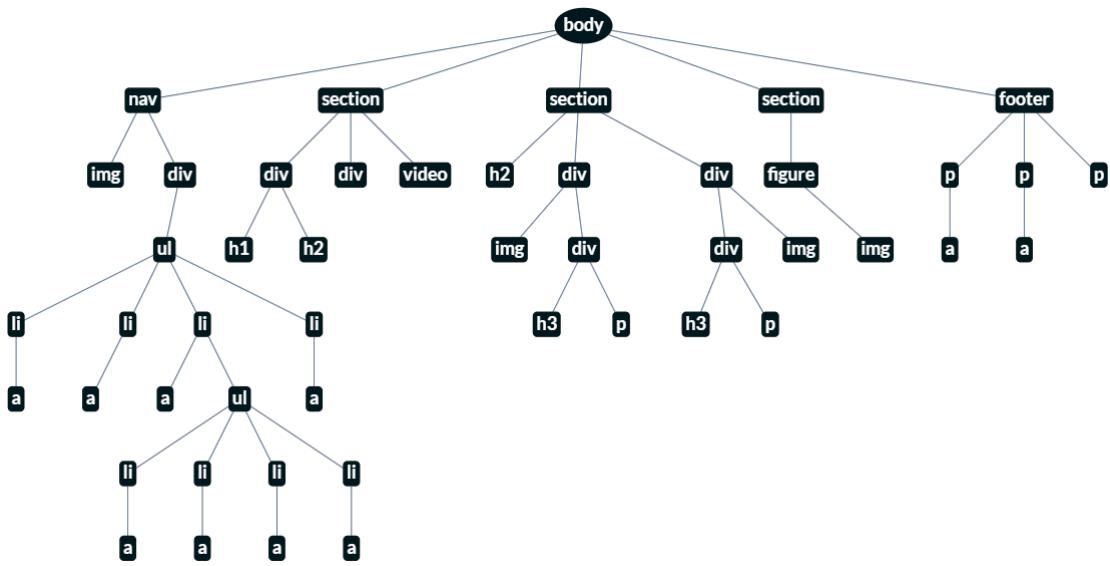


Figura 15: Mapa de Etiquetas de las subpáginas de *Portfolio*

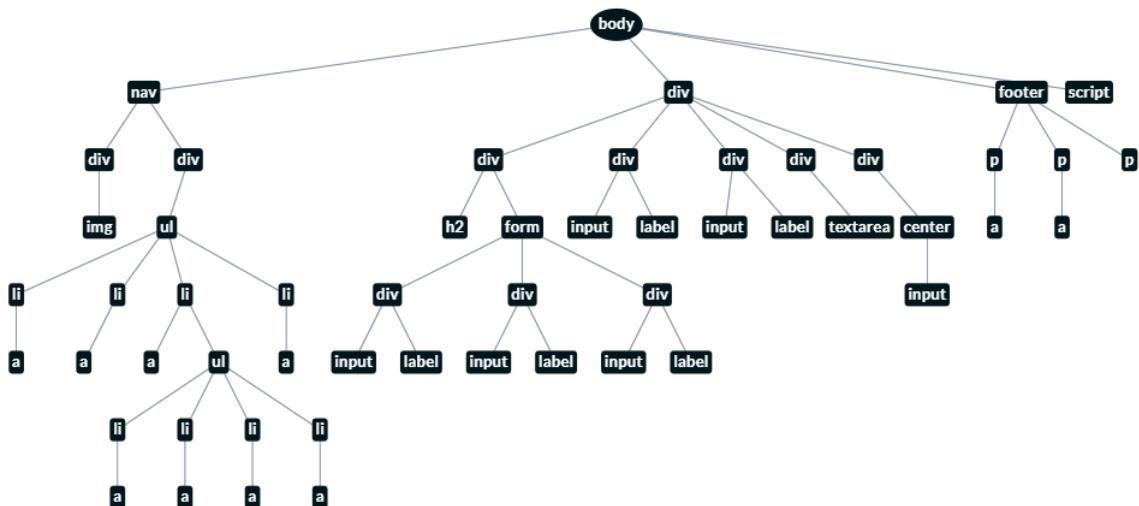


Figura 16: Mapa de Etiquetas de la página *Contacto*

3. Estructura de ficheros

Finalmente, tras haber generado todo el código HTML de las distintas páginas que componen el proyecto, la estructura de archivos se presenta de la siguiente manera:

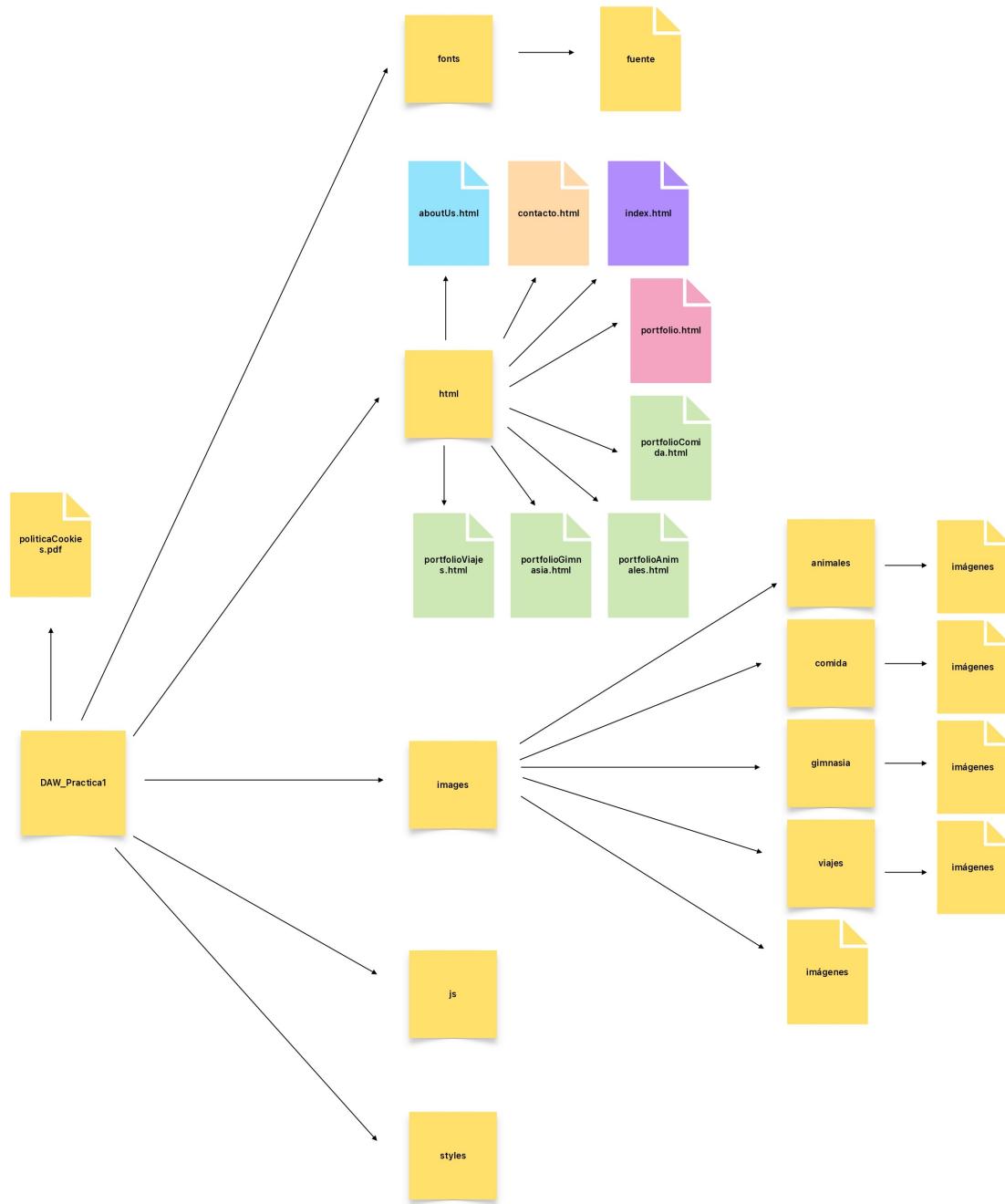


Figura 17: Estructura de ficheros actualización 1

4. CSS

La función principal de CSS es permitir personalizar la presentación visual de una página web. Con CSS, es posible aplicar diferentes estilos a los elementos HTML de las páginas, como colores, fuentes, tamaños, márgenes y posiciones, lo que permite crear un diseño atractivo y coherente en toda la página.

Para usar estas hojas de estilo se pueden incluir todas las reglas en el documento HTML usando la etiqueta `<style>` o se puede usar un fichero separado y enlazarlo al documento HTML mediante una etiqueta de tipo `<link>`. En el caso de nuestra página se ha optado por emplear esta última opción, ya que permite reutilizar los estilos en diferentes páginas, facilitando el mantenimiento y la coherencia visual del sitio web en su conjunto.

4.1. General

Para poder reutilizar una serie de estilos en todas las páginas se ha creado una hoja de estilos general que se va a utilizar para poder dar estilo al menú de navegación (presente en todas las páginas). En esta hoja general se tienen reglas como las que se detallan a continuación.

```
*,
*:::after,
*:::before {
    margin: 0;
    padding: 0;
    box-sizing: border-box;
    font-family: "Roboto", sans-serif;
}
```

Figura 18: Primera regla del documento

Añadir la regla anterior al comienzo de un documento se considera una buena práctica dado que permite eliminar los valores por defecto que un navegador pueda añadir a la página.

```
:root {
    --color-principal-claro: rgb(244, 237, 230);
    --color-principal-claro-transparente: rgba(244, 237, 230, 0.6);
    --color-principal-claro-semitransparente: rgba(244, 237, 230, 0.9);
    --color-complementario-claro: #3aaa38;
    --color-complementario-mas-claro: rgb(216, 238, 215);
    --color-white: #fff;
    --color-black: #2b2a2a;
    --color-grey-body: #43484e;
    --color-grey-light: #ccc;
    --enviar-mensaje-shadow: rgba(0, 0, 0, 0.4);
}
```

Figura 19: Definición de variables CSS

En este caso, las variables se definen en la pseudo-clase “:root” que puedan ser utilizadas a lo largo de todas las páginas para facilitar la gestión de cambios en el documento. Estas variables se recomiendan emplear en propiedades o valores que se repiten en varias partes de la página, como pueden ser la paleta de colores o las sombras empleadas. Todas los nombres de variables deben comenzar con “- -” y se pueden usar con la función var()

```
@font-face {
    font-family: "Karatone";
    src: url("../fonts/Karatonedemo-0Wr79.ttf");
}
```

Figura 20: Definición de tipografías

Con reglas de este tipo, se puede enlazar a un archivo que contenga una tipografía determinada para poder usarla en todo el documento. Para utilizar estos tipos de letra definidos se usa la propiedad `font-family` seguida del nombre que se le ha asignado a la fuente entre comillas dobles.

4.2. Inicio

La página *Inicio* es la página de principal la web y es dónde se proporciona la información más relevante. Así, aporta una vista general del contenido y los servicios que ofrece el sitio web.

El resultado que se pretende obtener mediante las herramientas CSS para esta página es el mostrado en la figura 2. Para poder conseguirlo, es necesario utilizar el mapa de etiquetas que se puede ver en la figura 22. Una de las reglas más salientes de esta hoja de estilos es:

```
#reviewsResumen {
    padding: 100px;
    text-align: center;
    background-image: linear-gradient(
        to left bottom,
        rgba(225, 224, 223, 0.495),
        rgba(213, 218, 212, 0.742)
    ),
    url("../images/nat-9.jpg");
    background-size: cover;
    box-shadow: inset 0px 0px 80px rgba(0, 0, 0, 0.68);
    background-attachment: fixed;
}
```

Figura 21: Reglas de la sección de *valoraciones* en la página de inicio

En ella se usa el ID de la sección como selector para aplicar las siguientes reglas. Se aplica un padding (distancia desde el borde hasta el elemento en sí) de 100px y se centra el texto que se tiene dentro de la sección. Luego se establece un degradado y una imagen como fondo de

la sección, donde el degradado actúa como un efecto de overlay para la imagen. En adición a eso, se especifica que el fondo debe cubrir toda la sección y se fija el `background-attachment` para que la imagen no se desplace conforme se avanza en la página. Por último, se añade una pequeña sombra al interior de la sección para mejorar su apariencia visual.

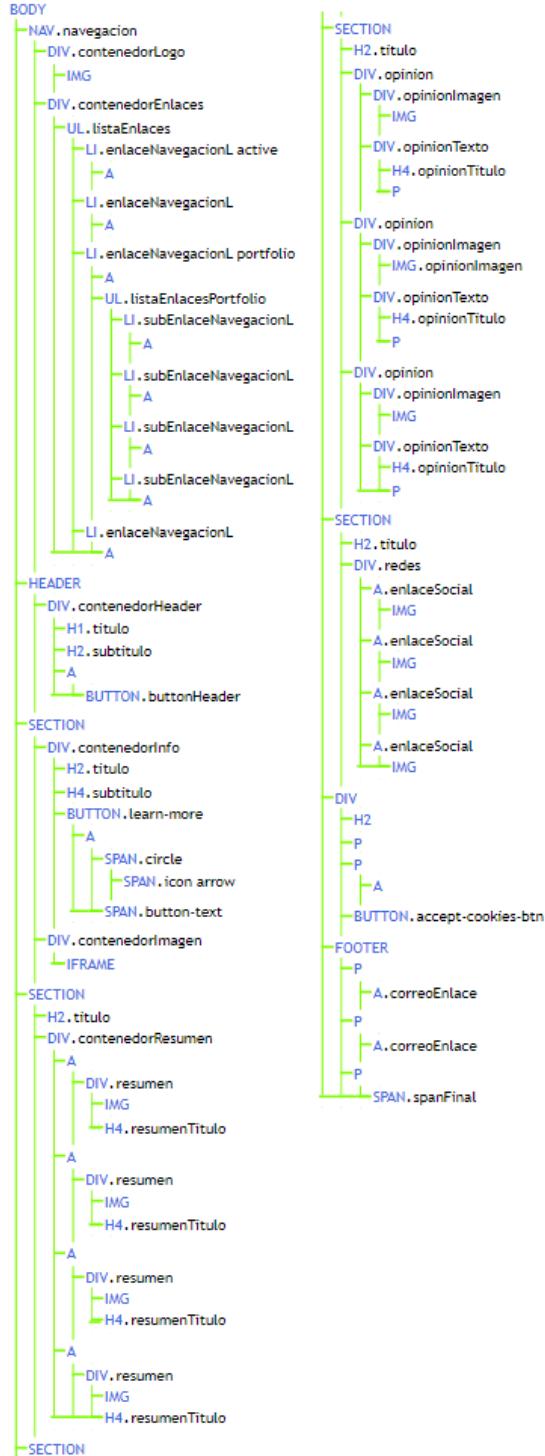


Figura 22: Mapa de Etiquetas de la página *Inicio*

4.3. About

La página *About* ofrece información acerca de los miembros que conforman el equipo de trabajo de la empresa “Fotografías Lemi”.

Al igual que en el caso anterior, el objetivo en CSS es lograr el resultado que se muestra en la figura 3, utilizando el mapa de etiquetas que se puede ver en la figura 24. Una de las reglas más destacables que se presenta en la hoja de estilos de esta página es:

```
header {  
    padding-top: 80px;  
    background-size: cover;  
    background-attachment: fixed;  
    position: relative;  
    background-image: linear-gradient(  
        to left bottom,  
        rgba(26, 27, 25, 0.784),  
        rgba(26, 27, 25, 0.784)  
    ),  
    url("../images/fotografoFondo.webp");  
}
```

Figura 23: Reglas del header de la página *About*

Para este caso se selecciona la etiqueta `<header>`. También, se aplica un padding y se fija el fondo del elemento, especificando el tamaño que debe tener y agregando un efecto de overlay. Se especifica una posición relativa dado que este elemento tiene dentro otros elementos con posición absoluta.

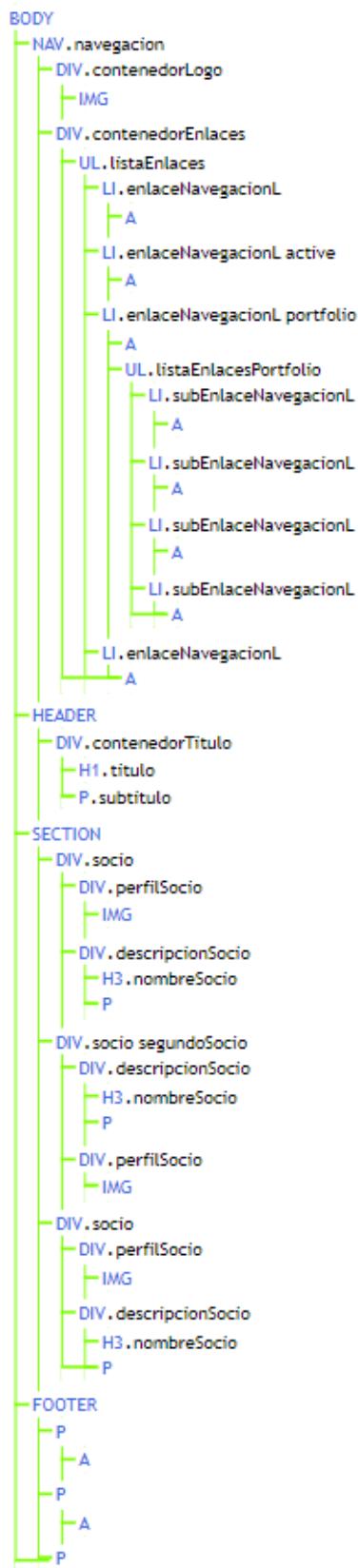


Figura 24: Mapa de Etiquetas de la página *About*

4.4. Portfolio

La página *Portfolio* ofrece información acerca de los tipos de fotografías en los que se centra la empresa.

Igual que en los casos anteriores, con el uso de diversas propiedades CSS se pretende obtener el resultado mostrado en la figura 4, usando el mapa de etiquetas representado en la figura 26. En esta sección, una de las reglas más salienteble de la hoja de estilos es:

```
imagenRecuadro{
    width: 100%;
    max-width: 35vw;
    height: auto;
    border: 10px double var(--color-grey-body);
    border-radius: 50%;
    box-shadow: 0 0 20px rgba(0, 0, 0, 0.55);
    background-color: var(--color-principal-claro);
    transform: scale(0.8);
    transition: all 0.5s ease;
}

.imagenRecuadro:hover{
    transform: scale(1);
    border-radius: 30px;
}
```

Figura 25: Reglas de una imagen en *Portfolio*

En estos conjuntos de reglas se estila una imagen que actúa a modo de enlace para cada una de las subsecciones. Primero se especifica el tamaño de la imagen y se le aplica un doble borde de 10px con un radio del 50 %. A continuación, se le añade una pequeña sombra para dar cierta profundidad a la imagen. Después se reduce la escala. En la siguiente regla se añade un efecto de hovering de forma que la imagen vuelve a su tamaño original y se reduce el radio del borde a 30px.

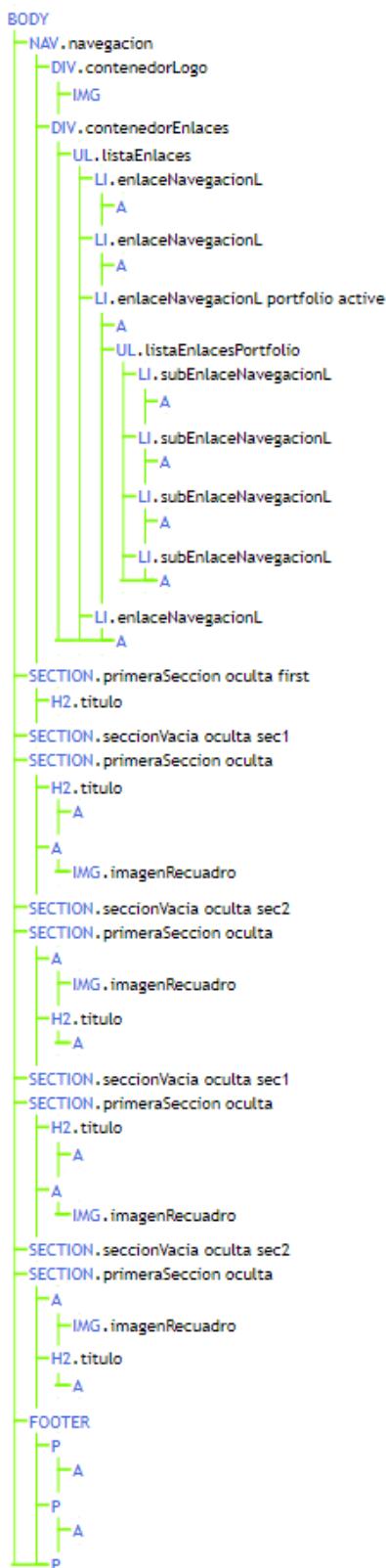


Figura 26: Mapa de Etiquetas de la página *Portfolio*

4.5. Subpáginas de portfolio

Las subpáginas del portfolio tienen como objetivo mostrar en mayor detalle cada uno de los tipos de fotografía en los que somos especialistas. Estos son: los viajes, los animales, la comida y la gimnasia. Con estas subpáginas se permite que los visitantes puedan explorar de forma más sencilla el trabajo en el área de interés que deseen.

Esta sección se corresponde con la figura 6 y con el mapa de etiquetas de la figura 15. La regla más notable aplicada en esta página es:

```
.front-page {  
    width: 100%;  
    height: 90vh;  
    color: white;  
    clip-path: circle(20%);  
    transition: all 0.6s ease-in-out;  
    position: relative;  
    margin-top: 30px;  
}
```

Figura 27: Regla de una parte del header de la subpágina

El único atributo que no se ha explicado en las páginas anteriores es `clip-path`. Esta propiedad recorta la parte visible en la forma especificada. En este caso, se recorta en un círculo que tiene un radio del 20 % de la sección. Después, con JavaScript se le dará dinamismo a este `clip-path`.



Figura 28: Mapa de Etiquetas de las subpáginas de *Portfolio*

4.6. Contacto

La página *Contacto* sirve para proporcionar un medio de comunicación entre el fotógrafo y sus clientes. Es decir, permite a los visitantes ponerse en contacto con el fotógrafo para hacer preguntas, solicitar información adicional sobre sus servicios, hacer reservas o programar sesiones de fotografía.

Esta sección se corresponde con la figura 5 y con el mapa de etiquetas de la figura 16. Una regla de esta página es:

```
.contenedorFormulario {  
    position: absolute;  
    transform: translate(-50%, -50%);  
    background-color: rgba(255, 255, 255, 0.7);  
    border-radius: 20px;  
    box-sizing: border-box;  
    box-shadow: 0 8px 32px 0 rgba(3, 7, 5, 0.847);  
    backdrop-filter: blur(4px);  
    -webkit-backdrop-filter: blur(4px);  
    top: 85%;  
    left: 50%;  
    max-width: 40vw;  
    width: 100%;  
    padding: 40px;  
}
```

Figura 29: Regla del contenedor del formulario de la página de *Contacto*

El nuevo atributo aplicado es el **backdrop-filter**. Este atributo aplica un filtro al fondo de la sección. En este caso, se aplica un ligero desenfoque para dar la sensación de ser un cristal (efecto conocido como Glassmorphism). El atributo de **-webkit-backdrop-filter** hace lo mismo pero asegurando la compatibilidad con los navegadores Google Chrome y Safari.

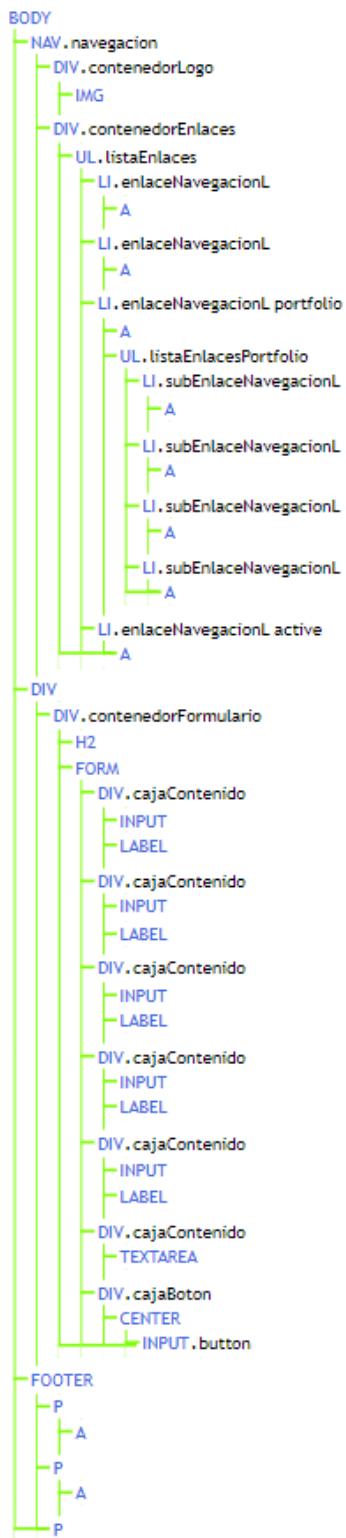


Figura 30: Mapa de Etiquetas de la página *Contacto*

4.7. Responsive

Un diseño es responsive cuando se adapta a diferentes tamaños de pantalla y dispositivos, manteniendo el contenido y la funcionalidad del sitio web. Es decir, se ajusta automáticamente a las dimensiones de la pantalla del dispositivo en el que se está viendo, lo que proporciona una visión óptima y coherente. Esto se consigue con el uso de diferentes técnicas que hacen lo posible sin sacrificar la legibilidad, la usabilidad o la accesibilidad.

Por ello, se debe asegurar un diseño responsivo. Para lograrlo, se pueden definir puntos en los que la página debe cambiar cierto atributo, modificándose así en función del tamaño de la ventana. Para esto se han usado media-queries. Una media-query es un tipo especial de regla de CSS que tiene la siguiente estructura:

```
@media only screen and (max-width: 950px){  
    h2.titulo{  
        font-size: 7vw !important;  
    }  
    .sec1{  
        transform: rotate(10deg);  
    }  
    .sec2{  
        transform: rotate(-10deg);  
    }  
}
```

Figura 31: Ejemplo de media-query

En este ejemplo, `@media` se usa para indicar al navegador que lo que sigue es una media-query. En esa línea de código, `only screen` se usa para asegurar la compatibilidad con versiones más antiguas de los navegadores, `and` es un operador lógico para indicar que se deben cumplir ambas condiciones y `(max-width: 950px)` indica que las reglas que están dentro de esta media-query solo se aplican cuando el tamaño de la ventana es de 950px o menos. Mediante el uso de estos elementos se puede asegurar que la página sea consistente en ventanas de una anchura menor y, también, de una anchura mayor.

4.8. Estructura de ficheros

Finalmente, después de haber agregado diferentes elementos CSS al código HTML en las distintas páginas que conforman el proyecto, la estructura de archivos se presenta de la siguiente manera:

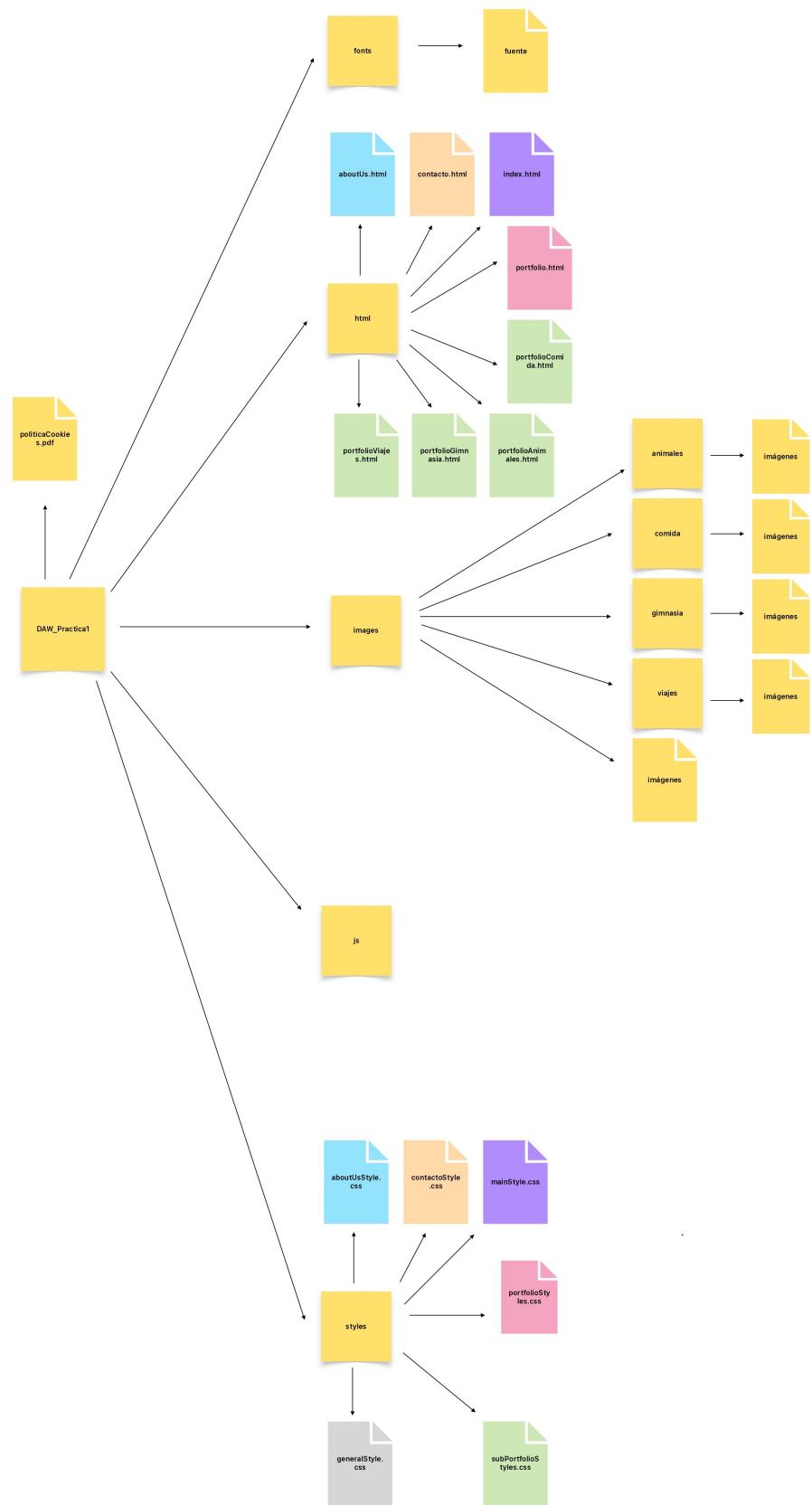


Figura 32: Estructura de ficheros actualización 2

5. JavaScript

Ahora que la página ya contiene HTML y CSS, es hora de adentrarse en el uso de JavaScript.

JavaScript es un lenguaje de programación interpretado y orientado a objetos que se utiliza principalmente en el lado del cliente para interactuar con el usuario y mejorar la funcionalidad de una página web. Con JavaScript, es posible crear animaciones, validar formularios y manipular elementos del DOM (Document Object Model), entre otras características que proporcionan interactividad y dinamismo a la experiencia del usuario en la web.

En este informe, se presentarán algunos de los detalles utilizados en este ámbito, tales como los métodos de acceso al DOM, los eventos y los distintos frameworks empleados.

5.1. Métodos de acceso al DOM utilizados

Los métodos de acceso al DOM son funciones que se utilizan en JavaScript para acceder y manipular los elementos de una página web. En nuestro caso, hemos usado los siguientes:

- `getElementById(id)`: devuelve el elemento HTML que tiene el ID especificado. Esto se puede ver en los documentos: `aboutUs1.js` y `general.js`
- `getElementsByName(name)`: devuelve una colección de elementos HTML que tienen el nombre de etiqueta especificado. Esto se puede ver en el documento `general.js`
- `querySelector(selector)`: devuelve el primer elemento HTML que coincide con el selector CSS especificado. Esto se puede ver en los documentos: `contacto.js` y `general.js`
- `querySelectorAll(selector)`: devuelve una colección de elementos HTML que coinciden con el selector CSS especificado. Esto se puede ver en el documento `general.js`
- `getElementsByTagName(name)`: devuelve una colección de elementos HTML que tienen el mismo atributo name. Esto se puede ver en el documento `contacto.js`

5.2. Eventos utilizados

En JavaScript, los eventos son acciones que suceden en una página web, como hacer click en un botón, mover el cursor sobre un elemento o cargar una página. Estos son una parte esencial de la interactividad en la web y permiten que los desarrolladores respondan a las acciones del usuario. En nuestro caso, hemos utilizado los siguientes: `click`, `scroll`, `mouseenter` y `mouseout`. El método que se utiliza para detectar la aparición de un evento es `addEventListener()`. Esta función toma dos argumentos: el primero es un string con el evento a escuchar y el segundo es la función encargada de manejar la aparición del evento.

5.2.1. Evento click

Este evento se activa cuando se hace click en un elemento y se usa en los archivos `contacto.js` y `general.js`. A modo de ejemplo, se va a analizar el caso de la función que se encuentra en `general.js`:

```

document.getElementById("botonAceptarCookies").addEventListener("click",() => {
  var popup = document.getElementById("cajaCookies");
  popup.style.display = "none";
  localStorage.setItem("cookieAccepted", "true");
});

```

Figura 33: Manejo del evento click

En esta función lo primero que se hace es seleccionar el elemento del DOM al que se le quiere añadir el manejador usando `document.getElementById('botonAceptarCookies')`. Esa función devuelve un elemento de la página al que se le añade el evento de click que se maneja con una función anónima usando `addEventListener`. Dentro de la función se selecciona otro elemento del DOM y se cambia su display a none para que no se muestre en la página. Después de hacer esto, se guarda en el almacenamiento local una cookie que se usa para comprobar si se debe mostrar el popup con la información sobre las cookies o si ya se han aceptado previamente.



Figura 34: Evento click previo en la página *Inicio*



Figura 35: Evento click posterior en la página *Inicio*

5.2.2. Evento scroll

Este evento se activa cuando se desplaza la página y se usa en los archivos `aboutUs1.js` (figura 42), `portfolio.js` y `subportfolio.js` (figura 46).

Para este caso se va analizar la función que se encuentra en `portfolio.js`:

```
const $ocultos = $(".oculta");
let screenSize = window.innerHeight;

function isInViewport(element) {
    if (element.getBoundingClientRect().top < screenSize) {
        return true;
    } else {
        return false;
    }
}

function handleScroll() {
    for (let i = 0; i < $ocultos.length; i++) {
        let element = $ocultos[i];
        if (isInViewport(element)) {
            element.classList.add("visible");
            if (element.classList.contains("seccionVacia")) {
                element.style.opacity="0.5";
            }
        } else {
            element.classList.remove("visible");
        }
    }
}

$(window).on("scroll", handleScroll);
```

Figura 36: Manejo del evento scroll

Al comienzo de este código se guardan los elementos que se van a animar: `$ocultos` almacena las secciones que están ocultas tras la carga de la página y `$seccionesVacias` hace referencia a las secciones que contienen las curvas que actúan a modo de camino. En esta parte del código también se guarda la altura de la ventana en la que se está visualizando. La función `isInViewPort(element)` comprueba que el elemento que se le pasa como argumento esté visible en el viewport. La función `handleScroll()` se encarga de manejar el evento de `scroll`. Para ello, recorre los componentes que se encuentran dentro de `$ocultos` y hace una llamada a la función antes mencionada para comprobar que el elemento que se está analizando en ese momento se encuentra dentro del viewport. Si está dentro del viewport, añade a las clases del elemento la clase “visible” para que el elemento pase a estar visible en la página. Además, si el elemento contiene en su lista de clases la clase “seccionVacia” cambia la opacidad de la sección a un 50 %. Para que el elemento esté a la espera del evento se utiliza la función de `$(window).on("scroll", handleScroll)`.



Figura 37: Evento scroll previo en la página *Portfolio*



Figura 38: Evento scroll posterior en la página *Portfolio*

5.2.3. Eventos mouseenter y mouseout

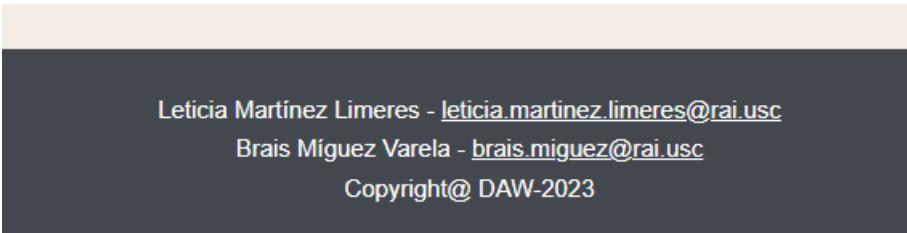
Estos eventos se activan cuando el cursor del mouse entra o sale de un elemento, respectivamente. Se usan en el archivo `general.js`:

```
document.querySelectorAll(".enlaceSocial").forEach((element) => {
    element.addEventListener("mouseenter", ()=>{
        element.style.transform = "translateY(-5px)";
        element.style.boxShadow = "0 10px 20px rgba(0,0,0,0.4)";
        element.style.borderRadius = "20px";
    })
});

document.querySelectorAll(".enlaceSocial").forEach((element) => {
    element.addEventListener("mouseout", ()=>{
        element.style.transform = "translateY(0px)";
        element.style.boxShadow = "0 0 0 rgba(0,0,0,0)";
        element.style.borderRadius = "10px";
    })
});
```

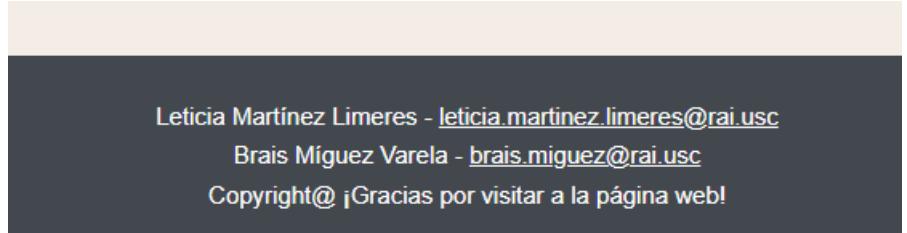
Figura 39: Manejo de los eventos `mouseenter` y `mouseout`

En la primera parte del código se seleccionan todos los elementos con la clase “enlaceSocial” y para cada uno de ellos se añade un manejador del evento `mouseenter` de forma que cuando esto suceda, se apliquen las reglas de CSS especificadas (figura 41). La segunda parte es la contraparte de lo anterior, es decir, vuelve a como estaba antes de que se el cursor entrase en el elemento (figura 40).



Leticia Martínez Limeres - leticia.martinez.limeres@rai.usc
Brais Míguez Varela - brais.miguez@rai.usc
Copyright@ DAW-2023

Figura 40: Evento `mouseout` en la página *Inicio*



Leticia Martínez Limeres - leticia.martinez.limeres@rai.usc
Brais Míguez Varela - brais.miguez@rai.usc
Copyright@ ¡Gracias por visitar a la página web!

Figura 41: Evento `mouseenter` en la página *Inicio*

5.3. jQuery

jQuery es una biblioteca de JavaScript que se utiliza para simplificar la manipulación de elementos HTML y la interacción con el DOM gracias a su facilidad de uso. Permite seleccionar elementos HTML, modificar su contenido y estilo, y crear animaciones de manera fácil y rápida. Además, aporta gran compatibilidad con múltiples navegadores, es decir, su código es muy robusto frente a la acción de la mayoría de los navegadores web modernos.

```
$document.ready(function () {
    $(".perfilSocio img").css("opacity", "0");
    $(".descripcionSocio").css("opacity", "0");

    $(".perfilSocio img").css("transform", `scale(0.5)`);
    $(".perfilSocio img").css("transform", `rotate(90deg)`);

    $(window).scroll(() => {
        const scrollTop = $(window).scrollTop();
        const windowHeight = $(window).height();

        $(".perfilSocio img").each((i, img) => {
            const offsetTop = $(img).offset().top;
            if (offsetTop < scrollTop + windowHeight) {
                $(img).css("transform", "scale(1)");
                $(img).css("transition", "2.2s");
                $(img).css("opacity", "1");
            }

            $(img).hover(function () {
                $(this).css("transform", `scale(1.1)`);
                $(this).css("transition", "1s");

            }, function () {
                $(this).css("transform", `scale(1)`);
                $(this).css("transition", "0.4s");
            });
        });

        $(".descripcionSocio").each((i, div) => {
            const offsetTop = $(div).offset().top;
            if (offsetTop < scrollTop + windowHeight) {
                $(div).css("opacity", "1");
                $(div).css("transition", "1.5s");
            }
        });
    });
});
```

Figura 42: Uso de jQuery

Este código JavaScript representa un ejemplo de esta biblioteca. Se ejecuta sobre la página *About* y se basa en la presentación gradual de las imágenes y las descripciones de los socios a medida que el usuario desplaza la página. El código está escrito dentro de una función que se ejecuta cuando el documento HTML está listo para ser usado por el JavaScript.

Para ello, se ocultan todas las imágenes y descripciones de socios estableciendo su opacidad al cero mediante la función `css()` (figura 43). También se asigna a las imágenes una escala del

50 % y una rotación de 90 grados.

Luego, la función utiliza el evento `scroll` para detectar cuándo el usuario desplaza la página y poder realizar la animación correctamente, rotando así la imagen, aumentando su opacidad y su escala mediante la función `css()` hasta que se vea completamente. Además, la imagen también cuenta con un efecto “hover” que aumenta su escala al 110 % y la reduce al 100 % cuando el cursor se retira (figura 45). Con el mismo evento, se muestran progresivamente las descripciones de los socios aumentando su opacidad (figura 44).

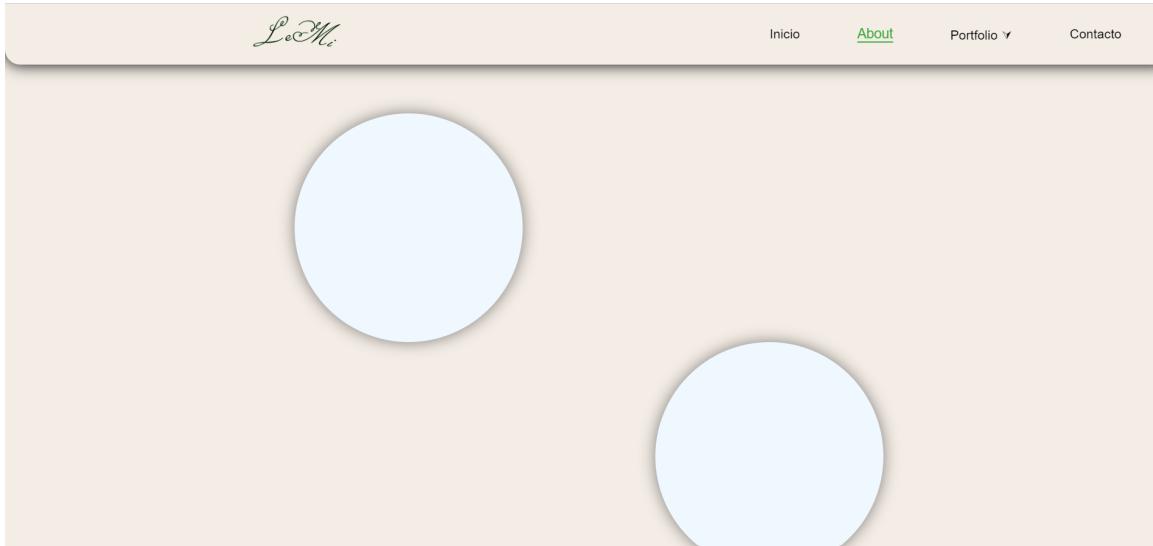


Figura 43: Efecto previo a la animación

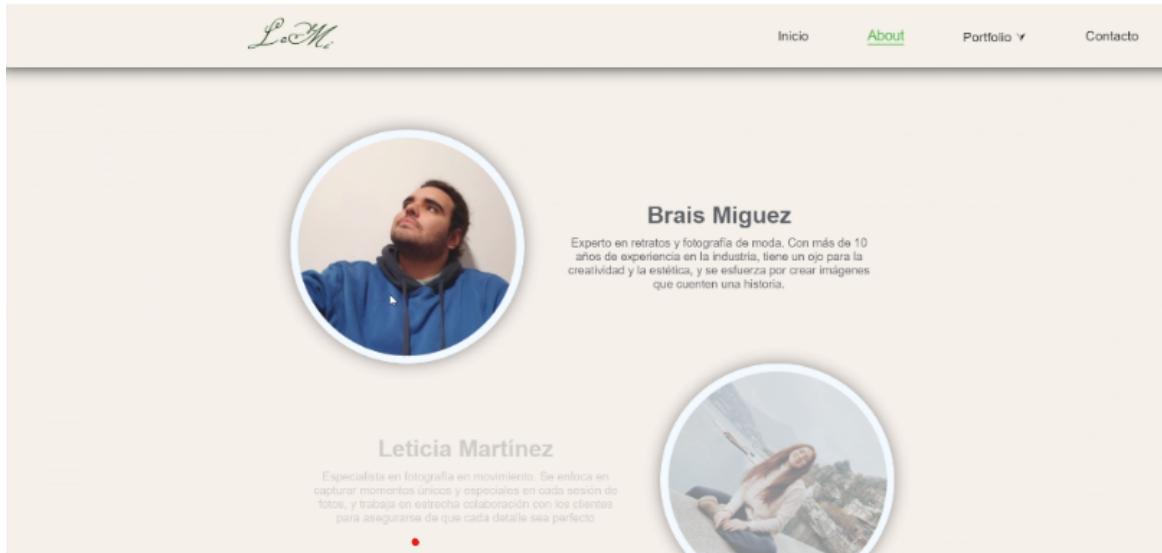


Figura 44: Efecto con la animación



Figura 45: Efecto hover

Otro ejemplo de uso de jQuery se puede ver en las subpáginas de la sección *Portfolio*.

```

const $seccion = $(".front-page");
let screenSize = window.innerHeight - 550;

function isInViewport() {
    if ($seccion.offset().top < screenSize) {
        return true;
    } else {
        return false;
    }
}

function handleScroll() {
    if (isInViewport()) {

        let scrollPosition = $(window).scrollTop() + 100;
        let percent = (scrollPosition / screenSize) * 100;

        $seccion.css("clip-path", `circle(${percent}%)`);
    }
}

$(window).on("scroll", handleScroll);

```

Figura 46: Otro uso de JQuery

Lo primero que se hace es definir las variables necesarias para la función: `$seccion` que es la primera sección de la página y `screenSize` que es la altura de la ventana, que será utilizada después para el cálculo del radio del círculo. La función `isInViewPort()` comprueba si la sección se encuentra en el viewport comparando si la posición de la parte de arriba es menor que el

tamaño de pantalla. La función `handleScroll()` es la función que se utiliza para manejar el evento. Dentro de esta función se hace una llamada a la función anterior. Si esa función devuelve true se guarda la posición de scroll en la variable `scrollPosition` y se guarda el porcentaje de scroll que se ha hecho en la sección en la variable `percent`. Una vez se tiene esos valores se usa la función `css()` para cambiar el radio del atributo `clip-path`. En jQuery los manejadores de eventos se añaden con la líneas del estilo de `$(window).on("scroll", handleScroll);`, donde "scroll" es el evento que va a lanzar la ejecución de la función `handleScroll`. Es decir, visualmente sería lo que se muestra en las figuras 47 y 48.



Figura 47: Efecto previo a la animación



Figura 48: Efecto posterior a la animación

5.4. JES6

JES6 es una versión del lenguaje de programación JavaScript que incluye numerosas mejoras y características nuevas para hacer que el lenguaje sea más eficiente, legible y fácil de usar. En nuestro caso, se recurre a las propiedades aportadas por JES6 a lo largo de todo el código del proyecto. Por ejemplo, en las diferentes imágenes incluidas en este informe se declaran variables con `let` y `const`, permitiendo que las variables tengan un alcance de bloque y no solo de función como ocurre con `var`. Además, `const` permite declarar variables constantes que no pueden ser reasignadas.

5.5. Expresiones regulares

Las expresiones regulares son importantes porque permiten buscar y manipular texto de manera precisa y eficiente. Hacén posible buscar patrones de caracteres en un texto, reemplazarlos, extraer información específica, validar entradas de usuario y realizar muchas otras operaciones.

En nuestro caso específico, en la página *Contacto* se incluye un formulario con campos, como nombre, apellidos, correo, teléfono de contacto, asunto y mensaje, que el usuario debe llenar si desea ponerse en contacto con nuestro equipo. Por lo que es esencial validar que el correo electrónico y el número de teléfono proporcionados sean válidos. Por lo tanto, incluir expresiones regulares en este caso es una excelente idea.

```
function sendEmail() {
    const nombre = document.getElementsByName('nombre')[0].value;
    const apellidos = document.getElementsByName('apellidos')[0].value;
    const email = document.getElementsByName('correo')[0].value;
    const telefono = document.getElementsByName('telefono')[0].value;
    const asunto = document.getElementsByName('asunto')[0].value;
    const mensaje = document.getElementsByName('mensaje')[0].value;

    const emailRegex = /^[^@]+@[^\s@]+\.\w+$/;
    const telefonoRegex = /\d{9}\$/;

    if (nombre != '' && apellidos != '' && asunto != '' && email != '' && telefono != '') {
        if (emailRegex.test(email) && telefonoRegex.test(telefono)) {
            alert('¡Mensaje enviado con éxito!');
        } else if (emailRegex.test(email) && !telefonoRegex.test(telefono)) {
            alert('Por favor, introduce un teléfono válido.');
            controlCampos();
        } else if (!emailRegex.test(email) && telefonoRegex.test(telefono)) {
            alert('Por favor, introduce un correo electrónico válido.');
            controlCampos();
        } else {
            alert('Por favor, introduce un correo electrónico y un teléfono válidos.');
            controlCampos();
        }
    } else {
        alert('Por favor, rellena todos los campos.');
        controlCampos();
    }
}

function controlCampos() {
    if (nombre == '') {
        nombre.classList.add('error');
    }
    if (apellidos == '') {
        apellidos.classList.add('error');
    }
    if (email == '') {
        email.classList.add('error');
    }
    if (telefono == '') {
        telefono.classList.add('error');
    }
    if (asunto == '') {
        asunto.classList.add('error');
    }
}

const botonEnviar = document.querySelector('.button');
botonEnviar.addEventListener('click', sendEmail);
```

Figura 49: Uso de expresiones regulares

En primer lugar, para el correo electrónico se utiliza el patrón de la variable `emailRegex` (figura 49) que se divide en tres partes:

1. La primera parte asegura que el correo electrónico no comience con un espacio en blanco o una arroba (@).
2. Asegura que el correo electrónico tenga una sola arroba y que no comience ni termine con una arroba ni tenga espacios en blanco.
3. Asegura que el dominio del correo electrónico tenga al menos un punto y no comience ni termine con un punto ni tenga espacios en blanco.

En el caso del teléfono, se utiliza la expresión regular `telefonoRegex` (figura 49) que valida que la entrada del número de teléfono del usuario empiece con un dígito y tenga exactamente 9 dígitos.

Por lo tanto, si se introduce alguno de estos campos de forma incorrecta el usuario será notificado como se puede apreciar en el ejemplo de la figura 50.

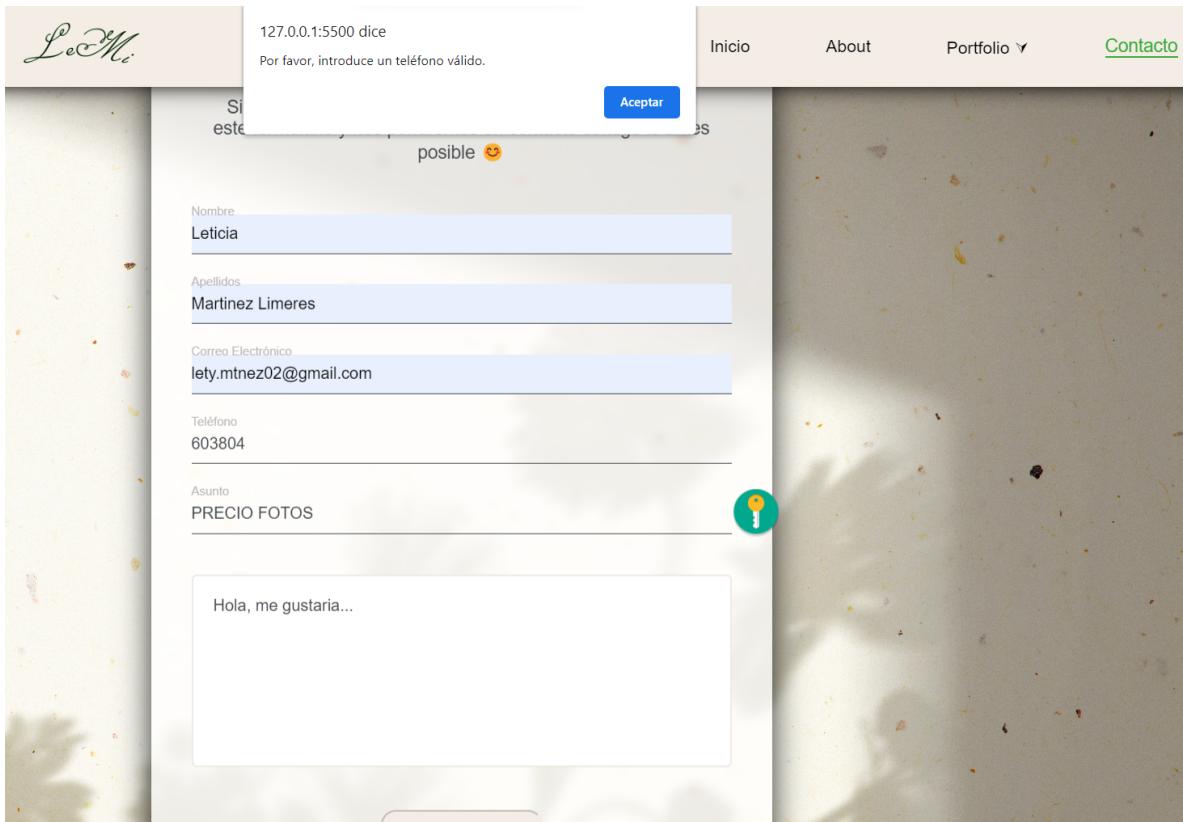


Figura 50: Configuración de las expresiones regulares en nuestra página

5.6. Carga de contenido mediante servidor Apache

5.6.1. XML

La carga de contenido usando formato XML se utiliza en la página *Inicio* en el apartado de opiniones de clientes para mostrar la información relativa a las diferentes valoraciones publicadas.

```
function loadDoc() {
    let xhttp = new XMLHttpRequest();
    xhttp.open("GET", "../js/ficheroXML.xml", true);
    xhttp.send();
    xhttp.onreadystatechange = function () {
        if (this.readyState == 4 && this.status == 200) {
            myFunction(this);
        }
    };
}

function myFunction(xml) {
    let i = 0;
    let xmlDoc = xml.responseXML;
    let x = xmlDoc.getElementsByTagName("Valoracion");
    for(i = 0; i < x.length; i++){
        document.getElementById('opiniones'+i+'Imagen').src = x[i].getElementsByTagName("Imagen")[0].childNodes[0].nodeValue;
        document.getElementById('opiniones'+i+'Titulo').innerHTML = x[i].getElementsByTagName("Titulo")[0].childNodes[0].nodeValue;
        document.getElementById('opiniones'+i+'Texto').innerHTML = x[i].getElementsByTagName("Review")[0].childNodes[0].nodeValue;
    }
}
loadDoc();
```

Figura 51: Uso de XML

```
<?xml version="1.0" encoding="UTF-8"?>
<Catalogo>

    <Valoracion>
        <Imagen>../images/martin.jpeg</Imagen>
        <Titulo>Calidad de fotografía inmejorable</Titulo>
        <Review>Estoy muy impresionado con la calidad de las fotografías.  
Sus imágenes son impresionantes y realmente capturan la belleza natural del mundo que nos rodea.  
Definitivamente trabajaré con ellos nuevamente en el futuro  
- Martin Omil</Review>
    </Valoracion>

    <Valoracion>
        <Imagen>../images/sabela.jpeg</Imagen>
        <Titulo>Unas grandes fotografías</Titulo>
        <Review>Brais es el mejor fotógrafo de retratos que he conocido.  
El realmente capture la personalidad de cada individuo en sus fotos y hace que la sesión sea divertida y relajante.  
Estoy muy feliz con las fotos que tomó de mi familia y las guardaré para siempre  
- Sabela López</Review>
    </Valoracion>

    <Valoracion>
        <Imagen>../images/raulma.jpg</Imagen>
        <Titulo>Artistas en toda regla</Titulo>
        <Review>Fue un placer trabajar con el equipo de fotógrafos de esta empresa.  
Me encantó cómo Lety capturó la emoción y la energía de mi clase de gimnasia.  
Definitivamente los recomiendo a cualquiera que busque fotografías de alta calidad  
- Raul Mariño</Review>
    </Valoracion>

</Catalogo>
```

Figura 52: Contenido para realizar la carga en formato XML

La función `loadDoc` crea una nueva instancia de `XMLHttpRequest` y utiliza los métodos `open` y `send` para enviar una solicitud GET del archivo XML llamado “`ficheroXML.xml`”.

Cuando se completa la solicitud, se activa el método `onreadystatechange`. Si el estado de la solicitud es `readyState==4` y el estado HTTP es `status==200`, se llama a la función `myFunction`.

Esta función analiza la respuesta utilizando el método `responseXML` para crear un objeto XML y luego utiliza el método `getElementsByName` para obtener un array de elementos “Valoración” del archivo XML. Luego, utiliza el método `getElementById` para asignar los valores de la imagen, el título y el texto de la valoración a los elementos correspondientes en la página web.

En resumen, este código (figura 51) carga un archivo XML (figura 52), analiza su contenido y actualiza la página web con los datos obtenidos del archivo XML (figura 53).



Figura 53: Opiniones obtenidas mediante XML

5.6.2. JSON

La carga de contenido usando formato JSON se utiliza en la sección *About us* para mostrar la información relativa a los socios de la empresa.

```

function cargaTexto() {
    fetch("../js/ficheroJSON.json").then/ajaxOK();
}

function ajaxOK(response) {
    response.json().then/muestraTexto();
}

function muestraTexto(json) {

    for(let i = 0 ; i < json.socios.length ; i++){
        document.getElementById(`socio${i+1}Nombre`).innerHTML = json.socios[i].nombre;
        document.getElementById(`socio${i+1}Texto`).innerHTML = json.socios[i].descripcion;
        document.getElementById(`socio${i+1}Imagen`).src = json.socios[i].foto;
    }
}

cargaTexto();

```

Figura 54: Uso de JSON

```

{
    "socios": [
        {
            "nombre": "Brais Miguez",
            "descripcion": "Experto en retratos y fotografía de moda.  
Con más de 10 años de experiencia en la industria,  
tiene un ojo para la creatividad y la estética,  
y se esfuerza por crear imágenes que cuenten una historia.",
            "foto": "../images/socio1.jpg"
        },
        {
            "nombre": "Leticia Martínez",
            "descripcion": "Especialista en fotografía en movimiento.  
Se enfoca en capturar momentos únicos y especiales en cada sesión de fotos,  
y trabaja en estrecha colaboración con los clientes para asegurarse de que  
cada detalle sea perfecto",
            "foto": "../images/socio2.jpg"
        },
        {
            "nombre": "Sandra Portela",
            "descripcion": "Fotógrafa de paisajes. Con un ojo para la composición y el color,  
captura imágenes impresionantes que transportan a los espectadores  
a lugares nuevos y emocionantes.",
            "foto": "../images/socio3.jpg"
        }
    ]
}

```

Figura 55: Contenido para realizar la carga en formato JSON

Este código en JavaScript utiliza la función `fetch`, una API web que permite realizar peticiones HTTP asíncronas para obtener recursos web, como archivos JSON. Por lo que, sirve para cargar el archivo JSON (figura 55) que contiene información sobre los socios de la empresa. Una vez que la respuesta de `fetch` está disponible, la función `ajaxOK` convierte la respuesta a formato JSON utilizando el método `json` y llama a la función `muestraTexto` pasando como argumento el objeto JSON.

La función `muestraTexto` recorre el objeto JSON y actualiza el contenido HTML de la página con la información de cada socio mediante el método `getElementById` para seleccionar cada uno los elementos HTML en la página que se verán afectados.

6. Estructura ficheros final

Para dar fin a este proyecto, la estructura resultante tras implementar todos los documentos generados para dotar a la página web de todas las etiquetas, métodos, atributos, eventos y acciones, explicados a lo largo de este informe, es la que se muestra a continuación.

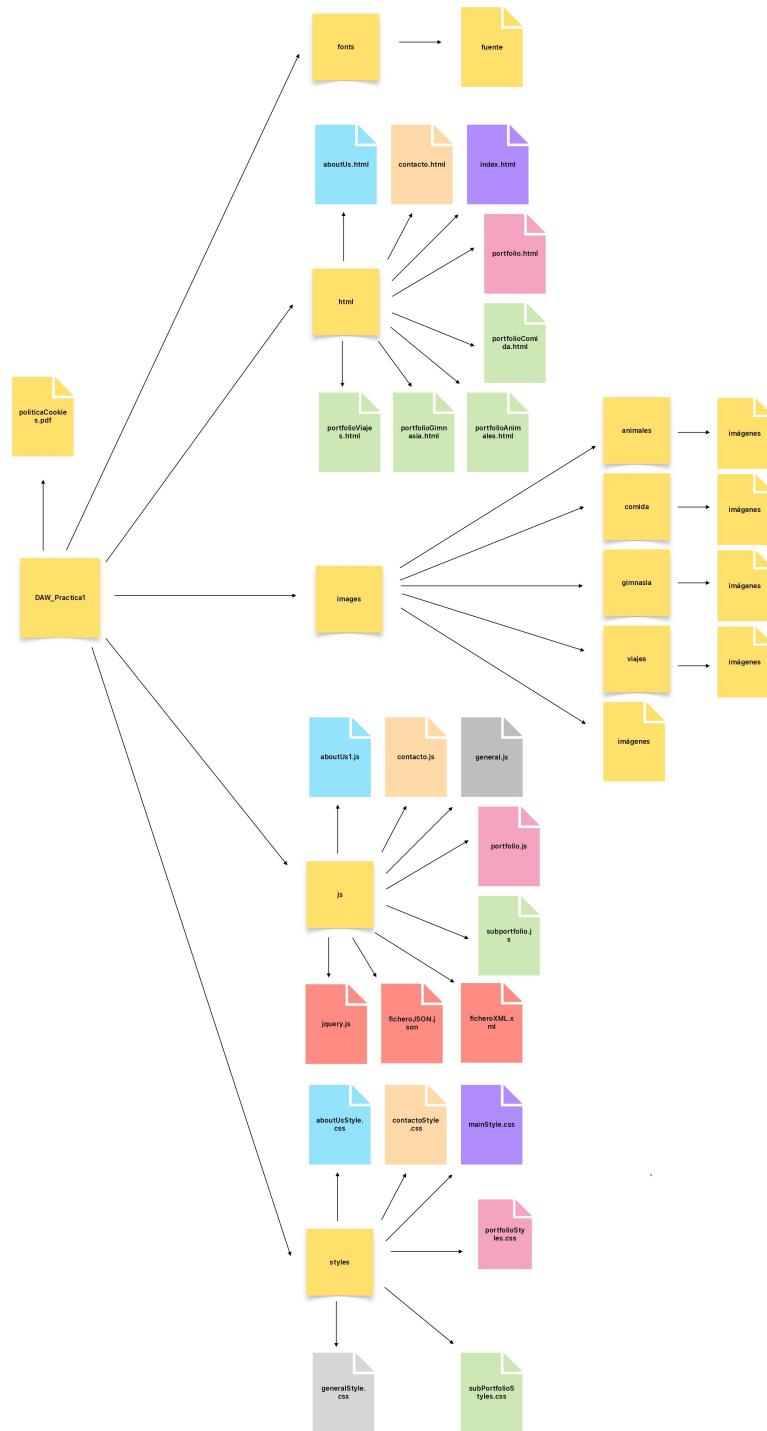


Figura 56: Estructura final de ficheros