Alarcon Moya, Teo

talarconmo@uoc.edu

PEC 2

Herramientas HTML y CSS II

**0. Links al proyecto**

|  |  |
| --- | --- |
| GitHub | <https://github.com/Moyat89/CocinaBDN> |
| Netlify | <https://cocinabdn.netlify.app/> |

**1. Documentación proceso de desarrollo**

**1.1 Creación de Proyecto utilizando UOC Boilerplate e instalación de dependencias adicionales**

1. Partiendo de que ya tengo Node.js instalado en el sistema, he creado una carpeta nueva para alojar el nuevo proyecto web.

2. He iniciado esta carpeta con git (comando git init) y he clonado el repositorio donde se aloja el *boileplate* de la UOC en esta carpeta (git clone <https://github.com/uoc-advanced-html-css/uoc-boilerplate.git>).

3. Tras esto, he añadido las dependencias necesarias para el proyecto (*Bootstrap y Stylelint*) y dos adicionales: *FontAwesome* para utilizar sus iconos de redes sociales y *Hover.css* para animar el estado *:hover* de los elementos del menú de navegación y así hacerlo más llamativo visualmente.

4. Para instalar *Bootstrap* he ejecutado npm i --save bootstrap @popperjs/core en la raíz del proyecto y he importado *Bootstrap.scss* (@import "bootstrap/scss/bootstrap") en nuestro archivo principal de estilo (main.scss). También he importado la dependencia en nuestro *main.js* (import \* as bootstrap from 'bootstrap';).

5. Luego he instalado *Stylelint* con la configuración recomendada para *SCSS* (npm install --save-dev stylelint-scss stylelint-config-recommended-scss). Al acabar la instalación, he creado el archivo .stylelintrc en la carpeta raíz del proyecto con la siguiente configuración, para asegurarme de que *Stylelint* comprobaba las reglas recomendadas:

{

"extends": ["stylelint-config-recommended-scss"],

"rules": {}

}

Por último, he añadido "stylelint": "stylelint src/\*\*/\*.scss" en la sección *scripts* del *package.json* para poder ejecutar *Stylelint* con *npm* (npm run stylelint).

6. La tercera dependencia que he instalado ha sido *FontAwesome*, para ello he ejecutado el comando npm install --save @fortawesome/fontawesome-free y he añadido la dependencia a nuestro *main.scss* (@import "npm:@fortawesome/fontawesome-free/css/all.css";).

7. Por último, he instalado *Hover.css* con el comando npm install hover.css –save y he añadido la dependencia a nuestro *main.js* (import \* as hover from 'hover.css';).

**1.2 Entorno de desarrollo**

Entorno de desarrollo:

* Sistema Operativo: Windows 10 Enterprise versión 22H2
* Node v21.6.2 x64 y npm 10.2.0. URL: <htps://nodejs.org/en/download>
* Visual Studio Code 1.89.0 x64. URL: <htps://code.visualstudio.com/Download>

1. Para el desarrollo se ha utilizado *Visual Studio* como *IDE* (entorno integrado de desarrollo) en el que he creado y he editado el código manualmente. En este, abría la carpeta entera del proyecto y navegaba a través de los diferentes archivos para modificarlos acorde a nuestras necesidades.

2. En paralelo, corría un servidor de desarrollo (comando npm run dev) para poder observar en tiempo real los cambios que llevaba a cabo en mi código *html* y *scss* hasta que lograba imitar a los *wireframes* proporcionados para esta PEC.

3. Mientras avanzaba con el proyecto, iba añadiendo los cambios (git add .) y confirmándolos con un comentario en el que explicaba el progreso (git commit -m”descripción del progreso”). Además de correr npm run stylelint para asegurarme de que seguía las pautas escogidas.

4. Como ya sabía que iba a hospedar la web en *Netlify*, no he estado compilando el proyecto localmente con npm run build además de npm run dev, ya que no me ha hecho falta.

**1.3 Publicación**

1. Al acabar el proyecto, y tras asegurarme que había confirmado todo los cambios, lo he subido a [*GitHub*](https://github.com/Moyat89/CocinaBDN)mediante el comando git remote add origin <https://github.com/Moyat89/CocinaBDN.git> y git push -u origin main.

2. Una vez subido en *GitHub*, ya estaba listo para publicarlo en *Netlify*. Para ello vamos a *Netlify*, hacemos *login* y hacemos *click* en *Sites* en el menú lateral izquierdo.

3. Vamos a *Add new site -> Import an existing project -> Deploy* *with GitHub*.

4. Introducimos nuestro usuarios y contraseña de *GitHub* y veremos nuestros repositorios. Si no vemos el de nuestra web, hacemos *click* en *Configure the Netlify app on GitHub -> Configure -> All Repositories / Only Select repositories (escogemos los repositorios) -> Save*.

5. Le damos un nombre al sitio, en nuestro caso cocinaBDN y nos aseguramos que en *Build command* tengamos npm run build, ya que es nuestro script para compilar nuestro proyecto para producción.

6. Pulsamos sobre *Deploy* y *Netlify* compilará el sitio y lo publicará. Podemos acceder haciendo en la URL bajo el título que le hemos dado (<https://cocinabdn.netlify.app/>).

7. Además de esto, como la página 3 de mi proyecto es un formulario que utiliza la funcionalidad *forms* de *Netlify* para automatizar el registro de los inputs, he ido a *Forms* en el panel izquierdo y he habilidad el *Form Detection*. Para que lo detecte *Netlify,* nos tenemos que asegurar de añadir el atributo data-netlify="true" o netlify a nuestro *form*.

8. El sitio web publicado se visualiza de la manera correcta y en la consola no se muestra ningún error. El formulario también se ha integrado perfectamente con *Netlify*.

**2. Decisiones de diseño**

Para esta PEC he decidido seguir la filosofía *mobile first* a la hora de desarrollar el diseño. No había ningún tipo de requisito, pero ya que la práctica pedía que el diseño fuese responsive, me ha parecido más adecuado partir siguiendo estas pautas para luego ir adaptando con *media* y *container queries* los tamaños y espacios.

Todos los archivos que se han creado para el sitio web, tanto de estilo o de contenido, han seguido las pautas de nomenclatura y formato marcas por la *UOC Boilerplate.*

Además del *footer* incluido en este como HTML parcial (*posthtml-include*), que ha servido para incluir la dirección de contacto y la barra de navegación inferior, también se ha creado otro HTML parcial (*header.html*) con la barra de navegación superior. Ambos se han incluido en todas las páginas de nuestro sitio web.

En cuanto a las dependencias adicionales añadidas, he decidido incluir *FontAwesome* para poder utilizar su librería de iconos de *Social Media* (ya que el *wireframe* mostraba un *footer* con links a las redes sociales) y *Hover.css* para hacer más llamativo el menú con animaciones. Este ha sido aplicado en el estado hover del logo (haciéndolo girar) y en los links del menú de navegación (creando una barra del color principal de la web bajo estos).

**3.Metodología elegida y/o guía de estilo**

Al ser un proyecto más o menos grande y complejo, se ha decidido utilizar la metodología *BEM (Block Element Modifier).* Teniendo identificadores únicos por bloques y elementos. Esto me ha permitido capturar específicamente de los elementos a los que aplicar estilos de una manera muy sencilla. Para la página del formulario (inscripción.htm), se han utilizado muchos elementos de Bootstrap que ya permite esta captura especifica de *BEM*, así que no me ha parecido adecuado cambiar su nomenclatura pues dificultaba la lectura.

En el siguiente ejemplo podemos un bloque de código en *BEM*:

<nav class="navbar">

    <a class="navbar\_\_logo hvr-icon-spin" href="./index.html">

        <div class="logo">

            <img src="./assets/images/badalona-logo.png" height="40"

                class="logo\_\_image d-inline-block align-top hvr-icon">

            <br>

            <span class="logo\_\_text">BadaPaellas</span>

        </div>

    </a>

    <div class="navbar\_\_spacer">&nbsp;</div>

    <div class="links text-center">

        <a href="./chefs.html" class="links\_\_item hvr-underline-from-center">Maestros Paelleros</a>

        <a href="./articulo.html" class="links\_\_item hvr-underline-from-center">La Tradición</a>

        <a href="./inscripcion.html" class="links\_\_item hvr-underline-from-center">Inscripción</a>

    </div>

</nav>

Además, se han utilizado muchos de las buenas prácticas de la *Code Guide de Mark Otto* (<https://codeguide.co/>), como la de no utilizar mayúsculas para las etiquetas, el utilizar las comillas dobles, no omitir las etiquetas opciones de cierre, etc.

**4. SASS**

Como se indica en la práctica, se han utilizado las funcionalidades propias de SASS tales como variables, nesting, funciones, parciales e importación.

* Variables: Se encuentran en el fichero \_variables.scss tal y como se encontraban en el UOC Boilerplate. Se han utilizado principalmente para colores recurrentes, tamaños y fuentes de letra.
* Nesting: Se ha utilizado el máximo anidado posible respetando la facilidad de lectura.

.footer {

  background-color: $background-color;

  background-color: $background-color-lch;

  &\_\_spacer {

    height: 1dvh;

    visibility: hidden;

  }

  &\_\_contact-info {

    height: 14dvh;

    padding-top: 5%;

    color: $white;

    color: $white-lch;

  }

  @media(min-width: $med-dev-min-width) {

    &\_\_contact-info {

      padding-top: 1.5%;

    }

  }

  @media(min-width: $big-dev-min-width) {

    &\_\_spacer {

      margin: auto 10dvw;

      visibility: visible;

    }

  }

}

* Parciales: Además del footer.html que se incluía en el boilerplate de la UOC, se ha decido que cada página contenga su estilo en un parcial (ya que todas muestran un contenido diferente), y tener la navegación y los estilos generales en otro.
* Importación: Tanto los parciales, las variables, las funciones y las dependencias externas (Bootstrap y FontAwesome) se han importado en main.scss:

/\*\*

 \* Use this file to import partials.

 \* Do not include SCSS code in this file.

 \*/

/\*\*

 \* Import variables

 \*/

@import "variables";

/\*\*

 \* Import functions

 \*/

@import "functions";

/\*\*

 \* Import npm dependencies

 \* see: https://v2.parceljs.org/features/module-resolution/

 \* see commented examples below

 \*/

@import "/node\_modules/bootstrap/scss/bootstrap";

@import "npm:@fortawesome/fontawesome-free/css/all.css";

// @import "some-node-module";

// @import "@some-company/some-node-module";

/\*\*

 \* Import layouts

 \* It's a best practice to use a different partial for each page

 \*/

@import "layouts/generals";

@import "layouts/navegation";

@import "layouts/cover";

@import "layouts/articles";

@import "layouts/authors";

@import "layouts/forms";

/\*\*

 \* Do NOT include anything else in this file!

 \*/

* Funciones: Se ha creado solamente una función en esta práctica: *tamaño-fuente-escalable*. Esta nos sirve para calcular el tamaño de las fuentes automáticamente dependiendo de un multiplicador introducido. Esto ha reducido el cálculo manual de los tamaños en los *media query*:

@function tamaño-fuente-escalable($tamaño-base, $factor-escala) {

    @return $tamaño-base \* $factor-escala;

  }

**5. Stylelint**

Para *Stylelint* se ha utilizado la configuración recomendada (npm install --save-dev stylelint-scss stylelint-config-recommended-scss) y además se ha añadido la regla de la nomenclatura BEM al archivo *.styleintrc: "selector-class-pattern": "^.[a-z]([a-z0-9-]+)?(\_\_([a-z0-9]+-?)+)?(--([a-z0-9]+-?)+){0,2}$".*

También se ha instalado *stylelint-config-recess-order* para comprobar que seguimos el orden de las declaraciones de la *Code Guide* de Mark Otto y se han añadido reglas personalizadas para cumplir con algunas de ellas. Además, se han añadido otras para asumir los objetivos propuestos por la PEC, como por ejemplo el no utilizar nombres para los colores, deshabilitar los formatos *rgb* y *rgba*, permitir solo unidades modernas de *viewport*:

 {

    "extends": [

        "stylelint-config-recommended-scss",

        "stylelint-config-recess-order"

    ],

    "rules": {

        "selector-class-pattern": "^.[a-z]([a-z0-9-]+)?(\_\_([a-z0-9]+-?)+)?(--([a-z0-9]+-?)+){0,2}$",

        "declaration-block-no-redundant-longhand-properties": true,

        "shorthand-property-no-redundant-values": true,

        "color-named":"never",

        "function-disallowed-list":["rgb, rgba"],

        "declaration-property-unit-allowed-list": {

            "width": ["dvw","%","px"],

            "max-width": ["dvw","%"],

            "min-width": ["dvw","%"],

            "height": ["dvh","%","px"],

            "max-height": ["dvh","%"],

            "min-height": ["dvh","%"]

            }

    }

}

**6. Justificación del uso de las pseudoclases funcionales y las capas de cascada**

**6.1. Pseudoclases funcionales**

* *:has()*: Se ha utilizado en varias ocasiones esta *pseudoclase*. Al utilizar la metodología *BEM*, no era muy necesaria ya que esta permite un nivel de granularidad muy alto, aún así, los he utilizado para distinguir entre elementos de la barra de navegación superior e inferior o para distinguir entre elementos dentro del formulario.

//Ejemplos

.navbar:not(:has(.social-media)) {

      background-image: linear-gradient(to bottom, rgba(0, 0, 0, 0.8), rgba(0, 0, 0, 0));

}

.navbar:has(.social-media) {

          .links {

            padding-right: initial;

          }

}

.navbar:has(.social-media) {

    .links {

      width: 40dvw;

      visibility: hidden;

    }

    .social-media {

      margin-top: auto;

      margin-bottom: auto;

      text-align: right;

      &\_\_item {

        margin-left: 2px;

        font-size: larger;

      }

    }

.form-check-input,

.form-control {

width: 75dvw;

height: 40px;

&:has(option) {

font-size: smaller;

}

&:focus {

border: $primary-color-lch;

border: $primary-color;

outline: none;

box-shadow: 0 0 0 0.2rem $primary-color;

box-shadow: 0 0 0 0.2rem $primary-color-lch;

}

}

* :where(): Lo he incluido varias veces para capturar varios elementos que tenían un nivel de granularidad más alto y aplicarles un mismo formato.

    :where(form.form-row) label,

    select,

    input {

      margin-top: 10px;

      margin-bottom: 10px;

    }

:where(section) p,

li,

figure {

margin-top: 15px;

margin-bottom: 15px;

}

* :is(): Me ha ayudado a especificar elementos muy granulares sin tener que hacer más complejo de los necesario el código o el anidamiento.

:is(section:last-child) p:last-child {

      margin-bottom: 60px;

    }

**6.2. Container Queries**

He utilizado varios container queries para poder aplicar diferentes estilos a los elementos dependiendo del tamaño de su contenedor. Esto me ha hecho posible cambiar la disposición o el tamaño de algunos selectores para que no sobresaliesen (o si, como el caso de la imagen del artículo en su versión para dispositivos grandes) de sus padres.

Ejemplo

container-type: inline-size;

@container(min-width: #{$big-dev-min-width}) {

      :is(.item) .item\_\_header,

      .item\_\_text {

        text-align: left;

      }

      .figure\_\_img {

        width: 120%;

        margin-left: -10%;

      }

    }

  }

**7. BootStrap**

He utilizado varios componentes BootStrap (container) o propiedades (text-center), pero principalmente lo he utilizado en la página inscripción.html. Sus elementos de lista (list-group, list-group-item) y de formularios (form-control, form-check-input,btn) me han ayudado a hacer más atractivo estos selectores HTML y no tener que preocuparme demasiado en sus tamaños. He personalizado estos selectores para que utilicen el color primario de la web en vez del azul que viene por defecto (y el estado hover de btn para ganar contraste) y en algunos casos los he personalizado con modificadores propios de BootStrap.

   Ejemplo CSS

  .form-label {

        display: block;

        font-weight: bold;

      }

      .form-check-input,

      .form-control {

        width: 75dvw;

        height: 40px;

        &:has(option) {

          font-size: smaller;

        }

        &:focus {

          border: $primary-color-lch;

          border: $primary-color;

          outline: none;

          box-shadow: 0 0 0 0.2rem $primary-color;

          box-shadow: 0 0 0 0.2rem $primary-color-lch;

        }

      }

      .btn {

        margin-top: 10px;

        font-weight: 700;

        background-color: $primary-color-lch;

        background-color: $primary-color;

        border: $background-color 1px solid;

      }

      .btn:hover {

        color: $white;

        color: $white-lch;

        background-color: $background-color;

      }

Ejemplo HTML

<div class="form-row">

                <input type="checkbox" class="form-check-input-sm" id="acceptTerms" required>

                <label class="form-check-label" for="acceptTerms">Acepto los términos y condiciones</label>

            </div>

Además, el input y el select tienen el estilo rounded-pill de BootStrap:

<input type="email" id="email" name="email" class="form-control rounded-pill text-center" required/>

Por todo el código HTML también se pueden ver “text-center” añadidos a las clases para que BootStrap centre sus elementos de texto.

**8. Observaciones**

* Para el *grid layout* utilizado en la portada se ha partido de un *layout* en *flex* y luego se ha transformado a *grid* si el navegador lo soportaba (*@supports*). El enunciado daba a entender que debíamos ir de *grid* a *flex* en caso de no compatibilidad. Aun así, me ha parecido más responsive comenzar en *flex* y luego, si el navegador lo soporta, aplicar el *grid*. De esta manera evitamos problemas de visualización en caso de que el *supports* y el *grid* fallen (para hacerlo de la otra manera solamente tenemos que añadir una negación).
* Los colores principales del sitio web han sido declarados en sus variantes *lch* y *hex* para evitar problemas de compatibilidad con el primero. Siempre que se ha declarado un color, se ha declarado primero su variante *hex* y después su variante *lch*, esto se ha hecho así pues si el navegador no es compatible con *lch*, este lo ignorará y dejará aplicada su variante *hex*.

.navbar {

    display: flex;

    min-height: $height-header-responsive;

    padding: 0 $padding-size-small 0 $padding-size-small;

    background-color: $background-color;

    background-color: $background-color-lch;

}

* En el footer se ha incluido un script en JS para resaltar la página donde nos encontramos en el menú de navegación a través de la clase current. Se ha hecho de esta manera y no manualmente en cada página debido a que quería mantener el header en otro archivo .html.

<script>

  document.addEventListener("DOMContentLoaded", function () {

    var currentPage = window.location.pathname;

    var pages = document.querySelectorAll('.links\_\_item');

    pages.forEach(function (link) {

      if (link.getAttribute('href') === currentPage) {

        link.classList.add('current');

      }

    });

  });

</script>

* El formulario utiliza la función de formularios de Netlify. En la sección 1.3 encontramos como activarla por parte del servicio.
* No se han incluido layers debido a que me ha parecido muy complicado con la metodología BEM, ya que no tenia conflictos de cascada por su granularidad. Cuando me dispuse a implementar esa parte del ejercicio, ya tenía el código casi acabado y hubiese implicado sobre complicarlo o rehacer alguna sección para poder poner un ejemplo práctico.