PEC 3. Herramientas HTML y CSS II aula 1

Ignacio Casares Ruiz

Junio de 2023

El enlace al respositorio de github utilizado es el siguiente: COMPLETAR

La página web subida en Netlify puede consultarse aquí: COMPLETAR

Instalación del boilerplate y creación del repositorio

Al igual que en las PECs anteriores, el primer paso que se llevó a cabo en la realización de esta PEC fue la instalación de UOC boilerplate.

Se creó un nuevo repositorio git local y lo vinculé a un repositorio remoto asociado a mi cuenta de Github. Este se puede consultar a través de la siguiente URL:

https://github.com/nacaru-w/h-II-P2

Esto permitiría tener un sistema de control de cambios, pudiendo revertir a un estado anterior en caso de necesitarlo. La habilitación de un repositorio público es necesaria para la publicación de la página a través de Netlify.

A continuación, instalé las dependencias del boilerplate a través de `npm` mediante el comando de terminal `npm install`. Se usaron las mismas dependencias que en la PEC anterior, eliminándose, además, `stylelint` y `bootstrap`. Después, se instaló Tailwind mediante el siguiente comando:

```
npm install tailwindcss --save-dev
```

Se añadió, por cuestiones de hábito, un script para el comando `npm run start`, que realiza las mismas funciones que `npm run build` en el archivo `package.json`.

Aunque para esta práctica no se utilizó una guía de estilo, se siguió usando el linter de código personalizado instalado localmente en mi dispositivo. Este realiza una serie de modificaciones (cambia aspectos como el indentado, el número de saltos de línea, etc) cada vez que realizo un guardado en la aplicación VSCode.

Configuración de Tailwimd

Se comenzó por crear un archivo de configuración de Tailwind, esto se realizó a través del siguiente comando en la consola:

```
npx tailwindcss init
```

Una vez creado el *boilerplate* del archivo de configuración, se procedió a introducir los colores que se habían asignado en la PEC2 a variables SCSS al archivo `tailwind.config.js`. Para tal fin se configuró el objeto `colors` de la siguiente manera:

```
module.exports = {
  content: ['./src/**/*.{html,js}'],
  theme: {
    colors: {
      transparent: 'transparent',
      current: 'currentColor',
      black: colors.black,
      white: colors.white,
      gray: colors.gray,
      emerald: colors.emerald,
      indigo: colors.indigo,
      yellow: colors.yellow,
      'normal-bg-color': 'rgb(245 245 245)',
      'text-color': 'rgb(31 31 31)',
      'secondary-contrast-color': 'rgb(204 204 204)',
      'link-color': 'rgb(51 51 51)',
      'colorful-element-color': 'rgb(248 221 201)',
      'lighter-colorful-element-color': 'rgb(254 248 244)',
      'colorful-element-border-color': 'rgb(253 245 239)',
    }
  }
}
```

Se puede observar como se añadieron las variables usadas en los archivos `blog.html` y `members.html` (estas aparecen en las posiciones finales del objeto). También se añadieron colores nativos de Tailwind, con el objetivo de usarlos puntualmente para ciertos tonos fuera de la temática de colores de la página web. Para poder acceder a ellos, hubo de importarse el módulo `colors`:

```
const colors = require('tailwindcss/colors')
```

Asimismo, se configuraron las dos fuentes utilizadas en la tipografía de la pagína web. Estas se configuraron como parte del objeto `FontFamily`, de la siguiente manera:[1]

```
fontFamily: {
  normal: ['JetBrains Mono', 'monospace'],
  serif: ['EB Garamond', 'serif']
}
```

[1] La fuente EbGaramond se añadió porque se utilizaba en la portada del proyecto. Al no incluirse la portada en las dos páginas incluidas en esta práctica, no se hizo ningún uso práctico de ella en la elaboración de esta PEC.

También se configuró un valor estándar (`normal`) para la propiedad `border-radius`, con el objetiov de proporcionar estandarización a este tipo de estilo de bordes.

```
extend: {
  borderRadius: {
    'normal': '1em'
  }
}
```

Además, se realizaron las importaciones necesarias en el archivo `main.scss` para que la herramienta funcione correctamente:

```
/** Tailwind */
@tailwind base;
@tailwind components;
@tailwind utilities;
```

Configuración de Postcss

Tras su instalación, Tailwind no funcionaba en el proyecto. Esto fue así porque se debía de configurar como un plugin de Postcss y no como una dependencia aislada. Para realizar tal acción, se creó un archivo JSON `.postcssrc`. En este, se especificó la inclusión de Tailwind como plugin.

```
{
    "plugins": {
        "tailwindcss": {},
        "posthtml-include": {}
    }
}
```

Esto permitió el buen funcionamiento de la librería de utilidades.

Configuración de VSCode

Con el objetivo de prevenir los errores generados por VSCode ante la inclusión de *at-rules* propias de Tailwind como `@apply`, `@layer` y `@tailwind`, se modificó la configuración de VSCode para ignorar este tipo de inclusiones en el código SCSS. Esto se realizó a través de la modificación de las preferencias, cambiando las preferencias sobre actuación de SCSS ante reglas *at-rule* no conocidas de «warning» a «ignore».

Una vez realizado el cambio, VSCode no alertó de ninguna *at rule* desconocida.

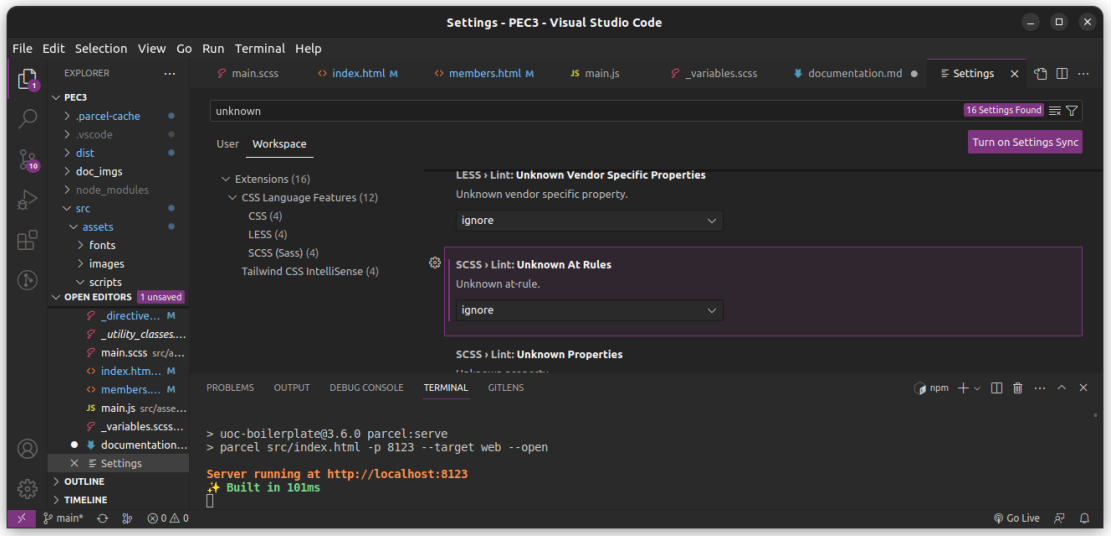

Imagen 1: menú en el que se configuran las alertas para las _at rules de VSCode

Además, adicionalmente y como ayuda ante la redacción del código relativo a las *utilities* de Tailwind, se instaló la extensión de VSCode Tailwind CSS IntelliSense. Esta incluye:

- Autocompletado de código relativo a Tailwind.
- Linteo, con resaltación de errores.
- Previsualización al realizar *hover* sobre el código.
- Modo de lenguaje Tailwind CSS.

La herramienta fue especialmente útil en la práctica, especialmente la opción de *hover* sobre el código, ya que te permite conocer el equivalente CSS al código redactado como *utility* de Tailwind.

Modificaciones en el sistema de ficheros

Se relizaron algunas modificaciones al sistema de ficheros respecto al del proyecto original. Se renombró la página SCSS en la que se situaba la hoja de estilos principal a `_directives.scss`, ya que es este tipo de contenido el que incluye en el proyecto actual. Se eliminaron los archivos de las páginas que no se recrearían, ya que el enunciado de la práctica requería la elaboración de dos de ellas. Se renombró `blog.html` a `index.html` para colocarla como página principal.

Preguntas específicas sobre utility-first CSS

- ¿Qué diferencias hay entre el enfoque de tipo CSS semántico (el que usaste en las otras PEC) y el CSS de utilidades? ¿Cómo afectó esto a tu proceso de desarrollo? ¿Y a tu código?
- ¿Qué diferencias encontraste entre usar una librería de componentes y una librería de utilidades?
- ¿Qué clases y componentes decidiste extraer y por qué?

Recreación de la página web

Una vez realizada la instalación de las dependencias, se procedió a recrear los archivos `index.html` (antes `blog.html`) y `members.html`.

Ya que la práctica requería un enfoque basado en Atomic CSS, con el objetivo de estandarizar la redacción de la misma, se tomaron las siguientes decisiones antes de dar comienzo a la redacción del código:

- A la hora de desarrollar el código relativo a elementos únicos, se realizaría un enfoque atómico puro, es decir, realizando adiciones de estilo a través del atributo `class` dentro de la línea de código HTML. Dependiendo de la complejidad del elemento y su situación respecto al resto de elementos de la página, en ocasiones de dejó la clase original aún cuando esta no posee ningún estilo asignado a la misma. Esto se hizo de tal forma con el objetivo de identificar más fácilmente los elementos en el código.
- En cuanto al desarrollo de elementos que se repiten en la página web o que poseen una amplia cantidad de estilos, este se realizó a través de un abordaje mediante directivas. En concreto, estas se desarrollarían en el archivo `_directives.scss`, a través de las `at-rules`: `@layer`, que sirve para identificar la función del elemento al que se le aplican los estilos dentro del código, pudiendo ser este `@component` o `@base`; y `@apply`, que se usa para introducir las propiedades y sus valores.

Para el desarrollo del proyecto en su totalidad, se hizo uso de la página a la que se había subido el proyecto de la PEC2. Mediante el inspector de estilos, se identificaron las reglas CSS aplicadas a los diferentes elementos y, haciendo uso de la extensa descripción de las utilidades proporcionada en la página de documentación de Tailwind. Se aplicaron los estilos originales mediante Tailwind con la mayor fidelidad posible.

Desarrollo en Tailwind de encabezado y pie de página

Al ser ambos elementos comunes a las dos páginas, estos se desarrollarán en esta sección individual.

Para el desarrollo del encabezado, Se sustituyeró el código original por clases Tailwind. Se dio preferencia a las unidades nativas de Tailwind, pero cuando estas no proporcionaban un resultado satifactorio, se hizo empleo de *bracket notation* para establecer la unidad original. Por ejemplo:

```
<div class="mx-auto w-[90%] max-w-[1000px] flex justify-between
↪  items-center">
```

Se eliminó el código relativo a la apertura y cierre del menú de navegación. Esto se realizó así porque, al haber ahora solo dos opciones, el menú permanece visible y sin problemas de formato incluso cuando la página web se visualiza a través de dispositivos de menor tamaño.

```
<nav class="z-950" id="primary-navigation">
  <ul class="flex" role="list">
    <li class="menu-item"><a href="members.html">Members</a></li>
    <li class="menu-item"><a href="index.html">Blog</a></li>
  </ul>
</nav>
```

La sección de navegación principal del sitio web está ahora contenida dentro de un elemento `nav` con una clase `z-950`, que establece un índice que asegurar que la navegación se muestre por encima de otros elementos. El nav contiene una lista desordenada `<ul>` con una clase `flex`, que utiliza Flexbox para alinear los elementos en una fila. Cada elemento de la lista `<li>` tiene una clase `menu-item` y contiene un enlace `<a>` que lleva a las dos páginas del sitio web. A la clase `menu-item` se le aplica la directiva siguiente, mediante `@apply`:

```
.menu-item {
  @apply px-8;
}
```

En cuanto al pie de página, al componerse este de elementos mayormente únicos, se tomó un enfoque similar:

```
<footer class="bg-secondary-contrast-color py-8 text-[.9em]">
```

Aquí se establece la clase `bg-secondary-contrast-color` que se aplicará al fondo del pie de página y la clase `py-8` para agregar un *padding* vertical de 8 unidades. También se establece el tamaño del texto en 0.9em mediante *bracket notation*, ya que Tailwind no posee una unidad nativa que se corresponda a tal cantidad de em.

```
<div class="w-[90%] max-w-[1000px] justify-around my-0 mx-auto pt-4 flex
↪  items-center">
```

Este div posee el contenido principal del pie de página. Se utiliza la clase `w-[90%]` para establecer un `width` del 90% del contenedor principal. La clase `max-w-[1000px]` establece un ancho máximo de 1000 píxeles. En el proyecto original, ambas clases se correspondían a una única propiedad de función CSS `min(90%, 1000px)`, pero al no admitirse en Tailwind, esta se tuvo que desarrollar así. Las clases `justify-around`, `my-0`, `mx-auto` y `pt-4` se utilizan para ajustar el diseño del contenido del pie de página. La clase `flex` crea un contenedor flexible que alinea los elementos horizontalmente.

```
<div class="px-2">
```

Este div contiene el contenido del lado izquierdo del pie de página. Se utiliza la clase `px-2` para agregar un relleno horizontal de 2 unidades, equivalente a `0.5em`.

```
<div class="button-box">
  <button class="button hover:bg-colorful-element-color type="
  ↪  button">Contact</button>
</div>
```

Aquí se encuentra un contenedor con la clase `button-box` que envuelve un botón. El botón tiene las clases `button` y `hover:bg-colorful-element-color` para aplicar estilos específicos y cambiar su fondo al pasar el cursor por encima.

```
<div class="py-4">
  <p class="my-4">
    Download our app
  </p>
  <div class="justify-evenly flex">
    <!-- Aquí se hallan los iconos -->
  </div>
</div>
```

Este div contiene un párrafo con la clase `my-4` para agregar un margen vertical de 4 unidades, lo que se corresponde a `1em`. Dentro del div hay otro div con las clases `justify-evenly` y `flex`, que se utilizan para alinear y distribuir equitativamente los elementos de su interior.

```
<div class="right-side-footer">
  <p class="font-bold my-4">Information</p>
  <div class="col3">
    <ul>
      <!-- Aquí se hallan los elementos de la lista -->
    </ul>
  </div>
</div>
```

En este div se encuentra el contenido del lado derecho del pie de página. El párrafo tiene la clase `font-bold` para aplicar un estilo de fuente en negrita. Dentro del `div` hay otro `div` con la clase `col3` que contiene una lista desordenada (`ul`).

```
<div class="sub-footer w-[90%] max-w-[1000px] border border-t-text-color
↪  border-b-transparent border-x-transparent flex justify-between mt-8
↪  mx-auto mb-0">
```

En este div se define la sección inferior del pie de página, que se coloca debajo del contenido principal. Se utilizan las clases `w-[90%]` y `max-w-[1000px]` para establecer el ancho de la sección. Además, se agrega un borde superior mediante las clases `border`, `border-t-text-color`, `border-b-transparent` y `border-x-transparent`. La clase `flex` se utiliza, de nuevo, para alinear los elementos del mismo horizontalmente, y la clase `justify-between` los separa `equitativamente`.

En el interior de este `div`, se halla el siguiente código:

```
<div class="logo-in-footer">
  <a href="index.html">
    <img class="main-logo w-[5em] h-auto"
    ↪  src="./assets/images/main_logo.svg" alt="Main logo of the
    ↪  website">
  </a>
</div>
<div class="pages-in-footer flex justify-center items-center">
  <ul class="flex">
    <li class="after:content-['|'] after:pl-4 pr-[1.25em]"><a
    ↪  href="">Members</a></li>
    <li class="after:content-['|'] after:pl-4 pr-[1.25em]"><a
    ↪  href="">Blog</a></li>
    <li><a href="">Contact & FAQ</a></li>
  </ul>
</div>
<div class="social-media-icons flex justify-evenly w-[8em]">
  <!-- Iconos -->
</div>
```

Que paso a explicar punto a punto:

- `logo-in-footer` contiene el logotipo principal del sitio web y se coloca en el lado izquierdo del subfooter. El logotipo está envuelto en un elemento `a` que conduce a la página principal `index.html`. El logotipo se engloba como elemento con la etiqueta `img`, con la clase main-logo que establece el ancho (`w-[5em]`) y la altura automática (`h-auto`).
- `pages-in-footer` muestra una lista de enlaces de las páginas del proyecto. La lista se coloca en el centro del subfooter utilizando las clases `flex`, `justify-center` e `items-center`, ya desarrolladas en elementos anteriores. Cada enlace de página se representa como un elemento de lista (`li`) y se muestra en línea utilizando la clase `flex`. Además, se agrega un separador vertical utilizando la pseudo-clase `after` y algunas clases de espaciado (`after:content-['|']`, `after:pl-4` y `pr-[1.25em]`).
- `social-media-icons` describe los íconos de las redes sociales. Los íconos son representados por etiquetas `svg` y se muestran en línea utilizando las clases `flex` y `justify-evenly`. Cada ícono de red social tiene una clase `mx-2` para agregar

un margen horizontal de 2 unidades y la clase `cursor-pointer` para establecer el cursor del mouse como un puntero al pasar por encima.

Desarrollo en Tailwind de `index.html`

Desarrollo en Tailwind de `members.html`

Publicación del sitio web

Para la publicación de la web en internet se utilizó el servicio Netlify. Este permite la publicación de la página a partir de un repositorio público alojado en GitHub.

Se configuró el repositorio a través de Netlify. No tuvo que especificarse la carpeta raíz.

Netlify realizó, entonces, el *deploy* del proyecto de forma exitosa y sin generar errores de ningún tipo.

Este se publicó en la siguiente URL:

https://delightful-kashata-ab932a.netlify.app/index.html

Propiededad intelectual: atribución

Aquí se realiza la atribución debida a los autores de los elementos externos utilizados en el proyecto. Todos ellos poseen una licencia compatible con su uso por parte de terceros, con una serie de condiciones según el tipo de licencia Creative Commons asignada.

- En cuanto a los iconos utilizados en la página (incluído el favicon), se usó el paquete de Sports de ainul muttaqin en The Noun Project (disponible en este enlace), liberados bajo licencia CC BY 3.0. Sirva este punto como atribución.
- Las imágenes de los miembros del club halladas en el archivo `members.html` han sido generadas por IA a través de la página Generated.Photos. Estas se encuentran en dominio público.[^3]
- Las foto de torneo «Players at a chess tournament» fue descargada de Wikimedia Commons. Su autor es Andreas Kontokanis, que la liberó bajo licencia CC BY-SA 2.0 (disponible en este enlace). Sirva este punto como atribución.