Xavier Morell Llorens,
Herramientas HTML y CSS II

PEC 3.
Recreación de la PEC2 con Tailwind

En esta PEC se recreará y publicará en internet usando Netlify una página web formada por 2 páginas de las 4 que forman el proyecto de la PEC2 de esta misma asignatura.
La primera de las dos páginas seleccionadas será el Home, la segunda será la página con la lista de grupos.

Se han seleccionado estas dos páginas porque, aunque visualmente son similares, siguen técnicas totalmente distintas, siendo la primera creada con un maquetado en flex mientras que la segunda tiene un maquetado con grid.

El contenido de esta PEC estará basado en el contenido del proyecto UOC Boilerplate (https://github.com/uoc-advanced-html-css/uoc-boilerplate/).

Preparación:

En primer lugar se ha clonado en github el proyecto de UOC Boilerplate para poder editarlo y trabajar con él: (https://github.com/xmorelll/UocBoilerplatePac3). Para el control de versiones y poder guardar con mucha facilidad los cambios realizados en el proyecto se ha decidido usar Sourcetree.

Una vez con el proyecto en local lo primero que se ha ejecutado es el comando *npm install* para descargar las dependencias del proyecto UOC Boilerplate.

Publicación:

Antes de continuar se ha decidido subir el proyecto a una url pública en internet mediante Netlify: https://hungry-hoover-79f661.netlify.app/

Publicar-lo a Netlify ha sido fácil, ya que al ya disponer de una cuenta con otros proyectos subidos solo se ha tenido de indicar el nuevo proyecto de Github que queríamos subir e indicar que para compilar este proyecto se usaría el comando *npm run build* y que el directorio a publicar es la carpeta /dist.

Dependencias instaladas:

- **Font-awesome:** Se ha añadido la dependencia de font-awesome para poder añadir algunos de los muchos iconos que proporciona esta librería. Para añadir font-awesome se ha ejecutado el comando: *npm install --save @fortawesome/fontawesome-free*.
- **Tailwind**: Tailwind se ha añadido con *npm install tailwindcss@compat --save-dev*. Después se ha ejecutado *npx tailwind init* para crear el documento de configuración de Tailwind.

Luego he añadido tailwind al documento “main.scss”.

```
@tailwind base;
@tailwind components;
...
@tailwind utilities;
```

I por último he añadido en el documento “postcss.config.js”:

```
require("tailwindcss")("./tailwind.config.js")
```

Se ha añadido PurgeCss que viene con Tailwind poniendo en el documento “tailwind.config.js”:

```
purge: [
  ‘./src/**/*.html'
],
```

Una vez realizada la instalación de Tailwind he visto que ya no podía compilar correctamente el código con *npm run dev*. Comentando el problema por el foro de la UOC he visto que el motivo era por la versión de node que era muy antigua así que la he actualizado a la última versión, y tampoco he podido compilar, nuevamente en el foro de la UOC he visto que el problema venia por ser esta una versión demasiado nueva. En este momento he decidido instalar la versión *v11.10.1* de node, y esta vez ya he podido compilar correctamente el proyecto en modo de development.

Stylelint:

En este proyecto se ha añadido también Stylelint con *npm install --save-dev stylelint* y *npm install --save-dev stylelint-config-standard stylelint-scss*.

Se ha creado un documento de configuración de Stylelint con la siguiente información:

stylelint.config.js

```
module.exports = {
 "extends": "stylelint-config-standard",
 "plugins": [
   "stylelint-scss"
 ],
 "rules": {
   "selector-nested-pattern": "^&",
   "indentation": 2,
   "no-descending-specificity": null,
   "no-eol-whitespace": null,
   "declaration-empty-line-before": null,
```

```
  "value-keyword-case": null,
  "at-rule-no-unknown": [
    true,
    {
      "ignoreAtRules": [
        "tailwind",
        "apply",
        "responsive",
        "variants",
        "screen",
        "use"
      ]
    }
  ]
 }
}
```

Diseño y desarrollo:

En este proyecto se usa el método CSS de Atomic CSS que consiste en usar clases de CSS pequeñas y que aportan un único estilo en cada situación. Las clases que se han usado mayoritariamente son las que ya proporciona la librería Tailwind, llegando a poder reescribir casi todo el código sin necesidad de añadir una sola línea de CSS. Para algunas clases se ha definido en el documento tailwind.config.js la definición de la clase de manera que luego se puede usar con la metodología propuesta por Tailwind.

Para el diseño de las 2 páginas web se han mantenido los mismos estilos ya definidos en la PEC2 para estas mismas páginas. Así que en esta ocasión el objetivo no era crear una página web sino recrearla con otro estilo de código. En la PEC2, se crearon una serie de clases CSS especialmente para la creación de las páginas propuestas en dicha PEC, en este caso se han usado clases genéricas y ya definidas por Tailwind para crear las mismas páginas que se crearon en la PEC2. El único problema lo he tenido en la página del HOME con una clase que modifica sus estilos según si el usuario realiza la acción de hoover o no sobre un div, así que estos estilos son los que se han definido con *@apply.*

Comentar también tener en cuenta que en la página del HOME en la versión de la PEC2 los estilos CSS tenían en cuenta browsers que no están preparados para ejecutar el estilo “grid”, mientras que en esta PEC3, no se ha tenido en cuenta estos browsers, ya que Tailwind no proporciona herramientas para esta situaciones.

En cuanto al desarrollo se ha decidido por un desarrollo enfocado a desktop first, pero sin olvidar que el proyecto debe ser responsive y poder visualizarse correctamente desde tablets y móviles.

Una vez terminado el proyecto con el código HTML y CSS escrito se ha comprobado que no exista ningún error con stylelint mediante el comando *npm run stylelint*. En la primera ejecución han salido una serie de errores:

- Dobles saltos de línea.

Se han corregido estos errores.

¿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?

El CSS de utilidades se basa en definir clases CSS que aportan únicamente un estilo o pocos estilos al componente, teniendo un nombre que informa sobre qué estilo está aportando. Ejemplo:

- “pr-1” para definir un PaddingRight
- “w-full” para definir un width de 100%
- “flex” para definir un display: flex

Mientras que el CSS semántico usado en la PEC2, se basa en definir clases que aportan más estilos, de manera que el HTML queda menos verbose y se pueden reutilizar estas clases, esta técnica a mi punto de vista permite modificaciones de estilos más rápidas, pero ralentiza la creación de la página en una primera instancia.

¿Qué diferencias encontraste entre usar una librería de componentes y una librería de utilidades?

A mi forma de entender entre Bootstrap (librería de componentes) y Tailwind (librería de utilidades) que son las dos librerías que se han usado en esta pec, Tailwind es bastante más fácil de utilizar, además no deja lugar a dudas sobre cuando usar una clase CSS o cuando intentar aprovechar una ya definida, ya que con bootstrap por cada estilo que el proyecto requiere usar una clase CSS ya predefinida.

¿Qué componentes decidiste extraer y por qué?

Decidí extraer en la página de Home los componentes requeridos para realizar la función del Hoover, ya que estos componentes se rehutilizan para cada elemento del grid, y porque solo usando las clases que ofrece Tailwind no podía conseguir el mismo efecto.