Xavier Ledesma Pons

Documentación del proceso

Para desarrollar esta práctica, primeramente, he clonado el repositorio de UOC Boilerplate de Github. A continuación, ejecutando el comando **npm install**, he instalado todas las dependencias necesarias para el desarrollo.

Una vez analizada la estructura del boilerplate, he decidido utilizar el IDE **Webstorm**, ya que estoy acostumbrado al entorno y se me hace más sencillo trabajar con el.

A continuación, he añadido la dependencia de **fontawesome-free** con el comando **npm install –save @fortawesome/fontawesome-free** para tener disponible los iconos que posteriormente utilizaré para el desarrollo del sitio web.

Para instalar **tailwind 2**, he ejecutado **npm install tailwindcss –save-dev**. A continuación, he ejecutado el comando **npx tailwind init** para crear automáticamente el fichero de configuración de **tailwind**. Por último, en mi fichero **main.scss** he introducido al inicio las directivas **@tailwind base; @tailwind components;** y al final **@tailwind utilities**. De esta manera, todas las paginas que incluyan este fichero del boilerplate, podrán hacer uso de las utilidades de **tailwind**.

Para que postprocesador de CSS de **parcel** incluya **tailwind**, en el fichero **.postcssrc** añadiremos el código **tailwindcss: {}** dentro de **plugins**.

Para configurar **tailwind** para eliminar todo el código CSS no utilizado, he configurado en el fichero **tailwind.config.js** la opción **purge**, indicando que este activada y que analice el contenido de todos los ficheros **html** de la carpeta **src**. Con anterioridad a este paso, he observado que el fichero de CSS generado por parcel pesaba **3.3MB** y después de eliminar el código no usado pesaba **66KB**, lo que supone un ahorro de memoria del **98%** aproximadamente.

Seguidamente, en el mismo fichero, en el apartado **theme**, he extendido la propiedad **flex**, con el siguiente código:

```
flex: {
  '4': '1 0 21%'
}
```

Con esto, he podido usar en una de mis páginas la utilidad **flex-4** que no viene definida por defecto en **tailwind**, ya que este solo incluye **flex-1**.

También he añadido el siguiente código para poder usar la utilidad **text-color-accent**:

```
textColor: {
  'accent': '#7e5c64'
}
```

En cuanto a estructura he seguido la misma que en la PEC anterior, pero solo he dejado las páginas **members.html** y **biography.html** (el resto de las páginas apuntan todas a **members**).

La primera página muestra el **flexbox** con los miembros del grupo y la segunda muestra 4 cards con información biográfica sobre el grupo igual que en la práctica anterior.

En este caso, cada página se estructura de la siguiente forma:

- Una sección **nav** que incluye la barra de navegación superior.
- Una sección **main** que incluye el contenido principal de la página.
- Una sección **footer** que incluye la navegación inferior.

Respondiendo a la pregunta de las diferencias entre un enfoque CSS semántico y uno de utilidad, cabe destacar que, en el caso semántico, las clases que se usan hacen referencia a una entidad con unas propiedades concretas. En cambio, en un enfoque de utilidad, las clases son lo más atómica posibles y solo modifican una propiedad concreta. Por ejemplo, en un enfoque semántico podemos tener la clase de **bootstrap card**, que hace referencia a un objeto con unas propiedades como el borde redondo o una sombra. En cambio, en un enfoque de utilidad, tenemos diversas clases como pueden **ser rounded-xl** para redondear y **shadow-md** para aplicar sombra al contenido.

Bajo mi opinión, una de las diferencias más claras que he encontrado, es que para construir ciertos contenidos de mi sitio web, al utilizar tailwind, he tenido que aplicar muchas clases, haciendo que el código HTML quedara muy largo. Por el contrario, leyendo solo el código HTML ya te puedes hacer una idea de como será el componente porque entiendes cada clase atómica que función realiza.

Como problemática, me gustaría destacar que al usar **tailwind**, he tenido que leerme bien la documentación, ya que es un poco complejo saber que proporciones se van a aplicar. Por ejemplo, cuando se hacen referencia a mt-3 no sabes exactamente el 3 que porcentaje de pixeles CSS implican ese **margin-top**.

Respondiendo a la pregunta de las diferencias que he encontrado, básicamente aquí he tenido que montar yo los elementos usando varias utilidades de **tailwind** en vez de usar un único componente que ya viene realizado.

He decidido extraer los siguientes componentes:

- **card-shadow**: este componente se asemeja al component card de bootstrap que había usado en el apartado biografia e incluye todas las utilidades para construirlo. Como hay bastantes utilidades y se repiten varias veces, lo he decidido extraer.

- **member-flex**: este componente hace referencia a cada miembro del grupo con su descripción que aparece en la página de miembros. Lo he decidido extraer por el mismo motivo anterior, se aplican varias utilidades y se repiten varias veces a lo largo de la página.

- **navbar**: este componente hace referencia a la barra de navegación que se incluye en la parte superior e inferior. Se ha extraído porque el código se repite a lo largo de todas las páginas.

- **navbar-logo:** este componente hace referencia al logo de la barra de navegación que se incluye en la parte superior e inferior. Se ha extraído porque el código se repite a lo largo de todas las páginas.

- **navbar-hambuger:** este componente hace referencia al menú de hamburguesa de la barra de navegación que se incluye en la parte superior e inferior. Se ha extraído porque el código se repite a lo largo de todas las páginas.

- **navbar-links**: este componente hace referencia al bloque de enlaces de la barra de navegación que se incluye en la parte superior. Se ha extraído porque el código se repite a lo largo de todas las páginas.

- **navbar-links-footer**: este componente hace referencia al bloque de enlaces de la barra de navegación que se incluye en la parte inferior. Se ha extraído porque el código se repite a lo largo de todas las páginas.

- **navbar-link**: este componente hace referencia a un enlace de la barra de navegación que se incluye en la parte superior e inferior. Se ha extraído porque el código se repite a lo largo de todas las páginas.

- **navbar-social**: este componente hace referencia al bloque de enlaces a redes sociales de la barra de navegación del **footer**. Se ha extraído porque el código se repite a lo largo de todas las páginas.

Cabe destacar que se ha verificado y adaptado el contenido para que sea responsive y se ha seguido una metodología mobile first y que cumpla con los estándares de accesibilidad recomendades para el desarrollo web.

Por último, he publicado mi sitio web en Netlify siguiendo los pasos detallados a continuación:

1) He creado una cuenta directamente con mi **Github**.

2) He seleccionado mi repositorio (https://github.com/xalepo4/HtmlAndCssII) y he indicado que la carpeta base es PEC2, ya que he subido la práctica en esa carpeta.

3) He indicado que se debe ejecutar el comando **npm run build** y que la web estará contenida en la carpeta **dist**.

4) Se ha generado el deploy con la siguiente URL: https://cranky-hamilton-a47123.netlify.app

5) Se adjunta la URL del deploy de la PEC anterior: https://hardcore-swartz-860f3e.netlify.app