Índice

Documentación del proceso de desarrollo

Para el desarrollo de la página se ha realizado una nueva instalación siguiendo los siguientes pasos:

1.- Descarga del UOC Boilerplate desde https://github.com/uoc-advanced-html-css/uoc-boilerplate con el botón Code → Download zip y posterior descompresión en directorio de trabajo.

2.- Instalación de las dependencias:
fontawesome : librería de iconos
accessibility: módulo para añadir opciones de accesibilidad al front-end

2.- He añadido la dependencia tailwindcss par poder utilizar la librería de utilidades
npm install tailwindcss --save.

3.- Para facilitar el desarrollo he continuado utilizado parcel versión dev para poder apreciar los cambios de forma dinámica en la página web. Parcel está integrado con el precompilador Sass.

También se han utilizado las herramientas para el desarrollador de Firefox Developer edition para analizar la aplicación de los estilos a los elementos html. Se ha utilizado asimismo la herramienta para comprobar como se ajustaba la página de forma responsive a los diferentes dispositivos móviles. En este aspecto cabe resaltar que se ha mejorado la adaptación responsive para que no aparezcan barras de scroll horizontal en dispositivos pequeños siguiendo las indicaciones de la corrección de la PEC2.

Una vez finalizado el desarrollo se ha procedido a generar la versión de producción con parcel build

Se ha utilizado Github como repositorio

Se puede acceder al repositorio en:
https://github.com/ptousd/eines2_pec3

Para paso a producción se ha utilizado Netlify.
Se puede acceder al proyecto en la url:

https://wizardly-hoover-d99977.netlify.app/

Una vez en producción se ha analizado el código con las siguientes herramientas:
Para corregir problemas de html
https://validator.w3.org/

Para corregir problemas de css
https://jigsaw.w3.org/css-validator

Para validar accesibilidad de las páginas online
He tenido que sustituir el que venía usando habitualmente por que ya no esta operativo
https://achecker.ca/checker/index.php
por este otro muy interesante
https://wave.webaim.org/

A partir de aquí se ha hecho Continuous Deployment hasta conseguir que los validadores no encontrasen problemas.

Indicar que con wave.webaim.org se han detectado errores de contraste que no aparecieron en la pec2 con achecker. Por este motivo las paletas de colores de la PEC3 varían ligeramente con respecto a la anterior para evitar estos errores de accesibilidad.

Justificación de las decisiones tomadas en el desarrollo

Decisiones sobre la estructura de archivos

Se ha utilizado la misma estructura de carpetas que la propuesta en el Boilerplate. Al principio estudié la posibilidad de utilizar la estructura de archivos de SMACSS o ITCSS pero se ha descartado dado lo reducido del proyecto. Al utilizar la librería de utilidades tailwindcss todavía se ha reducido más el tamaño de los archivos css. Siguiendo las indicaciones de la corrección de la PEC2 se han repartido en dos ficheros parciales: _home.scss contiene las clases de estilos propios del póster de la home (el póster no se ha pasado a tailwindcss) y el archivo apply.scss contiene los apply tailwind genéricos del site.

Configuración de tailwindcss

Se permite el uso de la librería completa, pero se ha configurado para que se haga un purgue al hacer un build. De esta forma se dispone de todo el potencial de la librería sin el inconveniente de que el archivo css final tenga un tamaño demasiado grande.

Se han configurado las fuentes y colores respetando el diseño de la PEC2. Recordar que las paletas de colores de la PEC3 varían ligeramente con respecto a la anterior para evitar estos errores de accesibilidad. Para ello:

Se ha cambiado la fuente serif para que por defecto, si está disponible, se utilice Montserrat. Se ha añadido la clase utilidad para aplicar fuentes al body para que sea el estilo utilizado por defecto en todo el cuerpo de la página.

```
<body class="sm:mx-4 font-serif">
```

Se han añadido dos nuevas utilidad de fuente con el objetivo de utilizarse en las cabeceras: serif-header y citas. Ejemplo de uso: `<h2 class="font-serif-header">La Historia</h2>`

Se ha definido la paleta de colores que se generarán como utilidades de la siguiente forma:

Se permite el uso de los colores : white, indigo, lima y cyan con su configuración por defecto de tailwindcss. Se mantiene el estilo de la práctica desarrollada con bootstrap.

Ejemplo de uso:

```
<span class="text-indigo-400 text-2xl ml-4 fab fa-google-plus-g"></span>
```

Se definen dos colores por compatibilidad con bootstrap: primary y secondary

Ejemplo de uso:

```
<header class="flex flex-row justify-around content-center bg-primary my-2">
```

Se redefine el color dark por el utilizado en la práctica anterior

Ejemplo de uso:

```
<a class="text-dark font-bold" href="lahistoria.html">La historia</a>
```

Se sustituye el color succes de bootstrap por yellow-300. En este caso he decidido mantener la paleta de escalas de yellow pero modificando la escala de 300. En la prácticas se usará básicamente para su uso en cabeceras.

Ejemplo de uso:

```
<h2 class="font-serif-header text-yellow-300">La Historia</h2>
```

Para que los items de los componentes de la banda de música se vean correctamente en todos los dispositivos, tras un estudio se determinó que se debía utilizar un minHeight de 13rem en el texto explicativo. Tailwind proporciona una serie de minHeight como utilidad pero como ninguno se corresponde exactamente con el necesitado se ha reconfigurado el que más se aproxima: el 52.

```
      <li class="category-item">
        <img class="w-full mb-4" src="assets/images/singer-1595864_1280.jpg" alt="Vocal"
/>
        <div class="min-h-52 px-4 pb-4">

          <div class="flex flex-row justify-between">
            <p class="font-bold">Pedro Tous Durán</p>
            <p>Vocal</p>
          </div>
          <p><span class="font-bold">Fecha incorporación:</span> 18/09/2018</p>
          <p><span class="font-bold">Historia:</span> Pedro empezó como vocal en "Los Si
ngers", pero el grupo se
          disolvió en 2018 y entonces fundó "Music Family"</p>
        </div>
      </li>
```

Para ajustar el uso de tailwind al estilo de la web se ha añadido el archivo: tailwind.config.js

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

module.exports = {
 purge: [
  './src/**/*.html',
 ],
 theme: {
  fontFamily: {
    'serif': ['Montserrat', 'serif'],
    'serif-header': ['Vollkorn', 'serif'],
   },
```

```
  minHeight: {
    '52': '13rem',
  },
  colors: {
    transparent: 'transparent',
    current: 'currentColor',
    white: colors.white,
    indigo: colors.indigo,
    lima: colors.lime,
    cyan: colors.cyan,
    'primary': '#007167',
    'secondary': '#6590aa',
    dark: '#222',
    yellow: {
      300: '#8f6a00'
    },
  }
 },
 variants: {},
 plugins: [],
}
```

Decisiones de diseño del sitio web

La página de ha dividido en 4 grandes áreas. Una cabecera, un menú de navegación, un contenido y un pie de página. Utilizando las etiquetas semánticas propias de HTML-5 para mejorar la accesibilidad.

Aunque el diseño inicial según los wireframes es sobre dispositivos de escritorio, se ha utilizado un desarrollo first-mobile ya que he considerado que es la mejor manera de que se vea bien en dispositivos pequeños.

Se han utilizado tres puntos de los proporcionados por tailwindcss, ya que son los típicos de los diferentes tamaños de dispositivos:
'sm': '640px',
'md': '768px',
'lg': '1024px',

Primero se ha desarrollado la web sin puntos de ruptura y probado en dispositivos tipo móvil (sm). En segundo lugar se ha añadido el punto a (md) para adaptar el funcionamiento a tables, para finalmente añadir el punto de ruptura para visualizar en formato escritorio (lg).

Todo ello para seguir la metodología first-mobile.

Se ha añadido javascript para utilizar la función de accesibilidad de la dependencia accessibility. Cualquier esfuerzo por mejorar la accesibilidad de una página vale la pena. Aunque hay que decir que con este módulo el esfuerzo ha sido mínimo.

Páginas implementadas con tailwindcss

Se han aplicado completamente las utilidades de tailwindcss en las páginas:

La Banda

Se ha utilizado un diseño en flex wrap para dispositivos a partir de tamaño tablet (md)
Teniendo en cuenta los márgenes para cada componente del grupo se ha asignado un tamaño fijo de 2/5 - 40% en tablet (2 columnas) y 1/4 - 25% en escritorio (3 columnas). Al tener 8 miembros en una distribución de 3 columnas quedan 2 miembros en la última fila. Para mantener una apariencia de tamaño de columna homogénea se han utilizado para flex-grow y flex-shrink el valor 0 que impide que se reparta el espacio disponible. Se ha considerado la mejor opción de diseño.

```
 .category-list {
  @apply m-0 px-2 list-none md:flex md:flex-row md:flex-wrap md:justify-center;
 }
```

```
 .category-item {
  @apply mb-8 md:flex md:flex-col md:justify-between md:w-2/5 xl:w-1/4 mx-4 shadow-
2xl;
 }
```

La Historia

Página muy sencilla en la que se ha explorado el uso de las utilidades tailwindcss. El esfuerzo ha sido mínimo ya que en este caso ya se utilizaron las utilidades de bootstrap, por lo que sólo se han tenido que cambiar algunas clases para adaptarse a la nueva terminología.

Index (Portada)

Para mantener una portada funcional se han cambiado el header y el footer para trabajar con tailwindcss ya que estaban implementados con bootstrap, para el poster se ha mantenido el css semántico.

Problema con el menú de navegación

El menú de navegación se ha configurado para que sea visible el icono en móviles y tablets y desplegado a partir del punto lg.

```
<button class="lg:invisible" type="button" aria-label="Toggle navigation">
        <span class="fas fa-bars"></span>
      </button>
      <div class="invisible lg:visible" id="navbarSupportedContent">
                  ....... . .
      </div>
```

Indicar que no funciona correctamente al no estar implementada la funcionalidad en JS. Se convertirse en un icono en dispositivos pequeños, pero no se despliega el menú al pulsar sobre el.

Es uno de los aspectos que se ha perdido al pasar de una librería de componentes a una de utilidades.

Gira 2021 (no implementada)

No se ha modificado la página lagira.html ya que para su implementación se habían utilizado componentes propios de bootstrap: carousel, formularios y ventanas modales. Al no tener réplica en tailwindcss por ser una librería de utilidades ha dejado de funcionar correctamente.

Respuesta a las cuestiones planteadas

¿Que diferencias hay entre el enfoque de el tipo CSS semántico y el CSS de utilidades? ¿Como ha afectado esto al proceso de desarrollo? ¿Y al código?

La principales diferencias es donde se codifican los estilos de las páginas y en que momento se generan los componentes del ciclo de desarrollo.

CSS semántico

Con el css semántico los estilos se aplican de la forma tradicional utilizando selectores a nivel de elemento, clase o id que se codifican en las páginas de estilos.

Gracias a las recomendaciones y a las metodologías es fácil mejorar la abstracción de los selectores y reducir su especificidad. Se evita así que mínimas modificaciones del diseño repercutan en complejas reescrituras de las hojas de estilos.

Sin embargo obligan a tener una visión inicial muy completa de la web para poder determinar su semántica y poder definir selectores abstractos, poco específicos y reutilizables. Esto significa que con este enfoque antes de empezar a desarrollar es obligatorio un estudio en profundidad del diseño de la web.

Hay que indicar que con css semántico se puede obtener una homogeneidad muy completa de toda la web. Al trabajar con bloques y elementos prefijados los estilos son los mismos para toda el site. Considero que este un factor muy importante en sitios de páginas comerciales con una imagen de marca bien definida.

CSS de utilidades

En el caso del css de utilidades o atomic css se utilizan un conjunto de clases ya predefinidos por la librería (aunque configurables) y nos olvidamos de la codificación de los selectores en la hoja de estilos. Se codifican los estilos directamente sobre el HTML aunque sin utilizar “style” sino las clases proporcionada por la librería.

La gran ventaja de este enfoque es que al ser a tan bajo nivel no se producen los típicos problemas motivados por la especificidad de los selectores. Aunque pueda parecer que para trabajar así se podrían trabajar directamente con styles existe una diferencia fundamental: con los styles tenemos a

nuestra disposición todo el abanico de posibilidades de css, pero con las librerías de utilidades restringimos los estilos a los proporcionados por la herramienta.

Si detectamos que algún conjunto de estilos se está utilizando muy asiduamente es el momento de pensar en crear un componente. Los componentes, a diferencia del enfoque semántico en el que se especifican desde el inicio, aparecen en etapas más tardías.

En mi opinión el inconveniente más importante es que al dar más libertad a los desarrolladores sobre la aplicación de estilos es más fácil que la web acabe degenerando perdiéndose la homogeneidad de las páginas.

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

Está claro que las librerías de componentes permiten trabajar a más alto nivel, proporcionando componentes de interface de usuario con funcionalidad completa; incluido javascript que permite la interacción del usuario. En bootstrap son ejemplos: menús de navegación responsive, formularios, paginaciones, ventanas modales o popovers.

Algunas de estas librerías como bootstrap incorporan también su propia librería de utilidades, lo que permite escribir páginas html sin apenas conocer css.

Las librerías de utilidades, como tailwindcss, trabajan a muy bajo nivel aunque a diferencia de las de componentes ofrecen la posibilidad de crear css semántico a partir del css atómico. Tailwindcss proporciona utilidades como apply que permite escalar el código hacia un css más semántico. No he visto la misma posibilidad con bootstrap.

Por otro lado las librerías de utilidades no van más allá de proporcionar una base de clases css, sin proporcionar ningún tipo de componente completo de interface de usuario con sus librerías de javascript.

¿Que componentes has decidido extraer y porqué?

Cuando de trabaja con atomic css, a medida que la web se va implantando se detectan combinaciones de estilos que se repiten cuando los elementos tienen la misma semántica. Esta repetición es debida principalmente al intento de mantener homogénea la imagen de la web.

Si en algún momento deseamos modificar el estilo de estos elementos, de forma global en toda la web para mantener una imagen homogénea, no hay otra solución que utilizar herramientas de reemplazo de texto que busquen todas las coincidencias con un patrón y las sustituyan por el nuevo estilo.

Para evitar este problema se pueden crear componentes y utilizarlos en los elementos html en lugar de atomic css.

En esta web en particular se han creado componentes para las cabeceras, ya que aparecen en diferentes páginas y se considera que, por razones de continuidad de imagen, deben tener el mismo aspecto en todo el site.

Otro componente que se repite, aunque en este caso en la misma página, son los miembros le la banda. Son 10 miembros en los que se repiten exactamente los mismos estilos atómicos. Si se quiere hacer una modificación de estilo es evidente que será la misma para todos los miembros de la banda. Como el estilo de los elementos de una lista <li> siempre va asociado a la lista <ul> también se ha creado un componente para la lista en si.

```
@layer components {
 .title {
  @apply pl-2 mb-0 text-2xl font-serif-header font-bold;
 }

 .subtitle {
  @apply font-serif-header text-4xl text-yellow-300 text-center;
 }

 .subtitle2 {
  @apply font-serif-header text-2xl text-secondary my-4;
 }

 .category-list {
  @apply m-0 px-2 list-none md:flex md:flex-row md:flex-wrap md:justify-center;
 }

 .category-item {
  @apply mb-8 md:flex md:flex-col md:justify-between md:w-2/5 xl:w-1/4 mx-4 shadow-2xl;
 }
}
```

Conclusiones

El objetivo de las librerías de utilidades es claro: olvidarse de codificar páginas de estilos y sobretodo de los posibles problemas derivados de la especificidad de los selectores. Además retrasa a fases más tardías de diseño las decisiones de determinar que consideramos como componentes.
El uso de css atómico no es equiparable a usar styles, ya que las librerías de utilidades solo ofrecen un subconjunto, previamente configurado, de los estilos disponibles para la web. Esto hace que en la web se pueda mantener cierta homogeneidad de imagen.
Dicho esto, exponer que con un mal uso se abre la puerta a la creación de páginas que no mantienen un estilo conexo. Este riesgo está más limitado con el uso de css semántico.

Anexo I. Enlaces a las fuentes de las imágenes con licencias gratuitas o Creative Commons

Conciertos
https://upload.wikimedia.org/wikipedia/commons/7/78/Concierto_OT_BCN_2018.png
https://upload.wikimedia.org/wikipedia/commons/6/63/Natalia_Lafourcade_en_concierto.jpg

La historia
https://pixnio.com/people/crowd/music-concert-performance-music-stage-festival-nightclub-audience

Guitarra logo
https://pixabay.com/es/photos/guitarra-guitarra-el%C3%A9ctrica-2957224/

Gutiarra eléctrica
https://pixnio.com/es/fondos-de-pantalla/guitarrista-musica-guitarra-electrica-altavoz-musico-instrumento-sonido

Acustica
https://pixnio.com/es/gente/mujeres/mujer-guitarra-clasica-guitarra-acustica-bajo-musica-guitarrista-melodia

Saxo
https://pixabay.com/es/photos/saxof%C3%B3n-instrumento-musical-m%C3%BAsica-3246650/

bateria
https://pixabay.com/es/photos/bater%C3%ADa-conjunto-personas-hombre-2599508/

piano
https://pixnio.com/es/diverso/piano-musico-artista-sonido-mano-pianista

acordeon
https://pixnio.com/es/media/musico-acordeon-contacto-directo-musica-instrumento

violin
https://pixnio.com/es/gente/mujeres/mujer-artista-violin-musica

cantante
https://pixabay.com/es/photos/cantante-sillouette-concierto-1595864/

Campo mallorca
https://commons.wikimedia.org/wiki/File:Mallorca0-2Leganes.jpg

Plaza Toros
https://commons.wikimedia.org/wiki/File:Die_Arena_des_Colisseu_Balear_umfasst_11.000_Quadratmeter_-_panoramio.jpg

Pabellón multiusos
https://commons.wikimedia.org/wiki/File:Multiusos_Fontes_do_Sar_-_02.jpg