# ÍNDICE

# 1 CONFIGURACIÓN DE LOS ENTORNOS

## 1.1 Creación de un boilerplate basado en parcel

En primer lugar, es necesaria la instalación de un buen editor de texto. Se ha optado por el uso de Visual Studio Code, debido a que es el editor que utilizo normalmente y que ya tengo instalado.

A continuación, se ha instalado el subsistema de Linux para Windows mediante el enlace con las instrucciones proporcionado en el Módulo 2. Se ha elegido Ubuntu 16.04 LTS como subsistema. La instalación se ha realizado correctamente.

Se ha abierto en el Visual Studio Code un terminal con Ubuntu mediante `bash` y se ha instalado la versión 13.x de node.js y npm mediante el comando:

```
curl -sL https://deb.nodesource.com/setup_13.x | sudo -E bash -
```

```
sudo apt-get install -y nodejs
```

Al comprobar si la instalación había sido correcta con npm -v, daba un error de sintaxis, pero reiniciando el terminal se ha solucionado el problema. Se ha instalado correctamente.

Se ha ejecutado el comando `npm install -g parcel-bundler` pero daba errores de acceso por lo que se ha instalado de forma local en el proyecto mediante el comando `npm install parcel-bundler --save-dev`. Mediante sudo también se podría haber instalado, pero puede dar errores en un futuro, por lo que se ha decidido instalar de forma local.

Mediante el comando npm init -y se ha creado el archivo package.json que se utilizará en el proyecto. También se ha instalado rimraf con `npm install --save-dev rimraf npm-run-all` para poder limpiar el proyecto.

## 1.2 Órdenes para desarrollo y producción

Para configurar las órdenes de desarrollo y producción se ha modificado el package y se han añadido las siguientes sentencias en el apartado “scripts”:

```
"dev": "npm-run-all clean parcel:dev",
```

```
"build": "npm-run-all clean parcel:build",
```

```
"parcel:dev": "parcel index.html",
```

```
"parcel:build": "parcel build index.html",
```

```
"clean": "rimraf dist .cache .cache-loader"
```

### 1.3 Gestión de dependencias: preprocesadores para css y javascript

Con la instalación de parcel ya se incorpora una configuración básica de Babel por defecto.

Se ha añadido a la raíz del proyecto el archivo .browserslistrc con la lista de navegadores que debe ser compatible el código que se procesará.

Seguidamente se ha instalado Autoprefixer a partir del comando `npm install --save-dev autoprefixer`. También se ha configurado PostCSS con Parcel mediante el archivo de configuración .postcssrc.

### 1.4 Repositorio git y sincronización con netlify

Una vez hecha la configuración básica del proyecto mediante GitHub Desktop se ha creado un repositorio y se ha enlazado con la carpeta local que contiene el código. De esta forma es mucho más intuitivo que a través del terminal. Se ha hecho un commit con el máster para guardar la configuración del proyecto y se ha publicado en GitHub.

Finalmente se ha enlazado el repositorio GitHub con Netlify, de forma que se ha generado un vínculo con el que se podrá acceder a la página web. La página se irá actualizando automáticamente a medida que se vayan publicando los cambios de código en GitHub.

El enlace al repositorio de GitHub es el siguiente:

https://github.com/nuri1995/PEC1

El siguiente vínculo enlaza a la página web publicada desde Netlify:

https://desarrollo-front-end-ncs.netlify.com/

## 2 EXPLICACIÓN DE LAS ENTIDADES HTML UTILIZADAS

Para el desarrollo HTML se han programado 6 páginas HTML distintas. La primera es la página de inicio, la cual se encuentra en el directorio raíz, mientras que las otras páginas están dentro de una subcarpeta llamada “pages”. Además, las diferentes imágenes que utiliza el sitio web, se han guardado dentro de la subcarpeta “img”.

En todos los archivos html el head, el header y el footer son comunes. En la cabecera del documento html se ha dado el título de cada página, el nombre del autor, se ha definido el viewport de forma que cuando se cambie de dispositivo la página web sea responsive. También se ha incluido el logo de la página web, se ha referenciado la hoja de estilos css y se han importado las fuentes de google Font que van a utilizarse. En este caso se ha escogido “Montserrat” como fuente para los títulos y “Raleway” para la fuente de texto normal. Finalmente se ha incluido la referencia al recurso de Font Awesome para poder utilizar sus iconos.

La cabecera de cada página, excepto la del índice (en esta la cabecera solo tendrá el logo), consta de un navegador, que será en línea o en bloque según el tamaño de la pantalla. Este navegador ha sido creado mediante una lista <ul> y las etiquetas <a> haciendo referencia a los distintos apartados que forman el sitio web. La categoría de preprocesadores se ha definido con la clase "construccion", ya que se aprovechará para hacer un pequeño desarrollo en javascript y no se definirá su página de detalle.

El pie de página está formado por 3 . En el primero hay tres enlaces, en el segundo hay los links a las redes sociales, donde también se hace uso de los iconos proporcionados por Font Awesome. Finalmente en el tercero hay información sobre el propósito de la página web. En este último , se informarán las fuentes de los contenidos de cada página, ya que han sido sacados textualmente de otros sitios web.

Al final de cada body, se ha añadido la referencia al script de JavaScript.

## 2.1 index.html, lenguajes.html y module.html

Estas tres páginas contienen diferentes menús. La página index.html contiene el enlace a 3 categorías (lenguajes del front-end, module-bundlers y preprocesadores), mientras que las páginas de lenguajes.html y module.html son las páginas de categoría que contienen el menú con los enlaces a las páginas de detalle.

En todas ellas, dentro del main se ha definido un  con la clase "desarrollo", ya que todas las páginas en las que el main contiene un menú, se han definido con esta clase, debido a que tienen un padding y un margen distinto a cuando contienen texto. Este  tendrá un título <h1> y otro . Este último  se ha definido con la clase "cuadricula_menu" y contiene 3 <figure>. Estos <figure> están formados por el enlace a las diferentes categorías y páginas de detalles, junto a un elemento <i> donde se llama al icono deseado de Font Awesome.

## 2.2 lenguaje_html.html y lenguaje_javascript.html

Estas dos páginas contienen el detalle de la subcategoría HTML y Javascript, de la categoría Lenguajes del Front-end. Las dos están formadas por un , y dentro de este  un título <h1> y otro . Este último  pertenece a la clase "content" debido a que para el formato Desktop tendrá unas medidas diferentes que para el formato móvil. Finalmente, en este apartado se ha introducido información de la subcategoría mediante la etiqueta <p>, junto a otro  con un <figure> que contiene un vídeo de Youtube.

## 2.3 module_webpack.html

Esta página contiene el detalle de la subcategoría Webpack, de la categoría Module Bundlers. Esta programada prácticamente igual que las dos páginas anteriores. La única diferencia es que el <figure> no contiene un video de youtube, sino que contiene otro elemento <img> incluyendo una imagen.

# 3 EXPLICACIÓN DE LAS ENTIDADES CSS UTILIZADAS

Se ha seguido el modelo mobile first, por lo que la primera parte del css es todo el diseño para móvil. Una vez se termina esta parte, mediante media queries se ha definido el estilo que tiene que tener para pantallas superiores a 768px. Por lo tanto, de 0 a 768px tendremos un diseño, y cuando sean pantallas superiores a 768px tendrá otro.

En la primera parte primero hemos definido los colores dentro de la :root. Estos colores se utilizaran durante toda la hoja de estilos. También se ha definido el tamaño principal de letra, el cual son 16px para la versión móvil y el color negro para los textos. El tamaño de las otras fuentes estará definido en rem. Las fuentes que se han definido son Realway para los textos y Montserrat para los títulos.

Para el menú de la cabecera se le ha dado estilo para que quedase en bloque en la versión móvil, y en línea en la versión de escritorio. Para la versión de escritorio se hace mediante un grid, donde la columna del logo será más pequeña que las columnas de las categorías. También se ha definido un color distinto para el enlace que corresponde a la misma página en la que se está.

En la versión móvil al logo se le ha dado un tamaño de width del 30% mientras que en la de escritorio es de un 80%.

En el header y el footer se ha definido el fondo para que aparezca con un degradado de color y para que los textos aparezcan con la fuente Montserrat. En la versión desktop el footer y el header se han creado como un grid.

En el apartado de la clase “nosotros” del footer, se ha definido como un block para la versión móvil. En la versión desktop se ha definido en línea.

Las imágenes ocuparán un 90% de donde estén contenidas y los vídeos de youtube también se han hecho que sean responsive, de forma que según el tamaño de la pantalla sean mayor o menor.

Los menús de las categorías del formato móvil están formados por un grid de 1 columna y 3 filas, excepto para la categoría del “module”, que está formado por 2 filas. En la versión escritorio esto cambia, y pasa a estar formado por 3 columnas y una fila, y en el caso del “module” por 2 filas.

# 4 DESARROLLO JAVASCRIPT

El script index.js contiene el pequeño desarrollo javascript que se ha hecho en esta página.

Se ha creado la función popUp(), la cual se encarga de mostrar una pantalla informando de que el enlace al que intenta acceder aun está en construcción. Esta función se llamará cada vez que se intente entrar a la categoría de “Preprocesadores” o a las subcategorías de “CSS” o “Parcel”.

Para ello, se obtienen todos los ítems de la clase “construccion” que haya en esa página HTML y se les hace un loop para definirle a todos los elementos un eventListener, de manera que cuando se clique a un elemento de esa clase, se llamará a la función popUp().