

Índice

1. Preparación del entorno.....	2
2. Decisiones de diseño	3
3. Metodología elegida y/o guía de estilo	3
4. SASS	4
5. Stylelint.....	5
6. Repositorio y publicación	5

El presente documento es una guía orientativa que resume los puntos que componen la PEC1. No es exactamente una solución a la práctica, sino que en ella se proporcionan ejemplos sobre cómo se podrían abordar los distintos apartados.

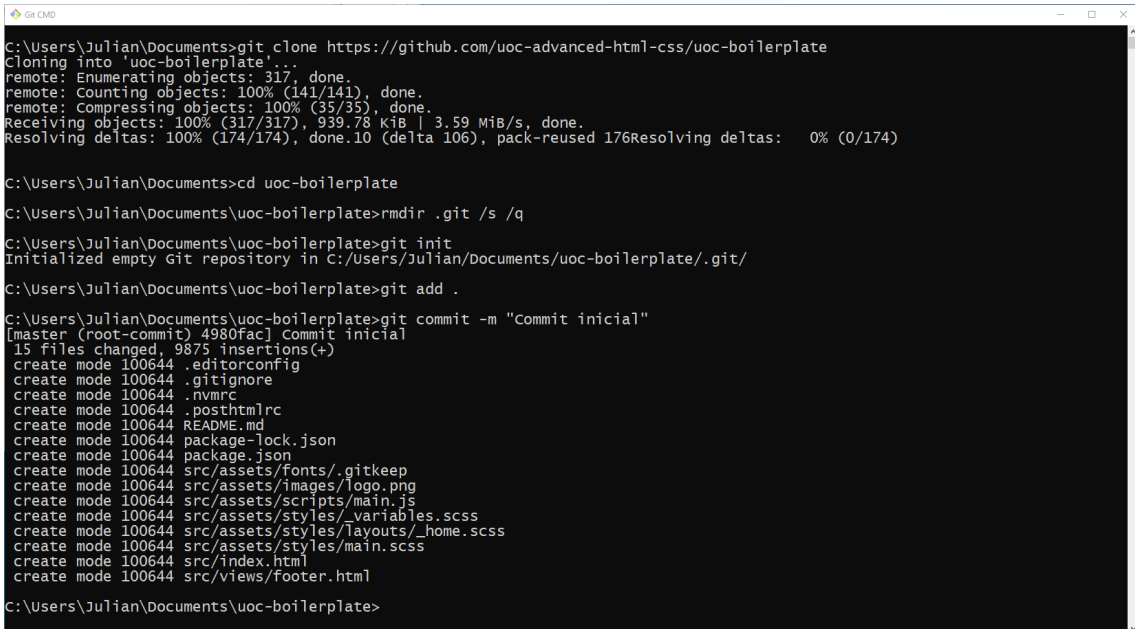
1. Preparación del entorno

En este caso, el entorno de desarrollo está compuesto por:

- Sistema operativo: Windows 10 Pro versión 21H2
- Node 18.16.0 x64 y npm 9.5.1. URL: <https://nodejs.org/en/download>
- Visual Studio Code 1.77.3 x64. URL: <https://code.visualstudio.com/Download>

Clonado de UOC boilerplate e inicialización del repositorio:

- *git clone https://github.com/uoc-advanced-html-css/uoc-boilerplate*
- *cd uoc-boilerplate*
- *rmdir .git /s /q*
- *git init*
- *git add .*
- *git commit -m "Commit inicial"*



```
C:\Users\Julian\Documents>git clone https://github.com/uoc-advanced-html-css/uoc-boilerplate
Cloning into 'uoc-boilerplate'...
remote: Enumerating objects: 317, done.
remote: Counting objects: 100% (141/141), done.
remote: Compressing objects: 100% (35/35), done.
Receiving objects: 100% (317/317), 939.78 KiB | 3.59 MiB/s, done.
Resolving deltas: 100% (174/174), done.10 (delta 106), pack-reused 176Resolving deltas: 0% (0/174)

C:\Users\Julian\Documents>cd uoc-boilerplate
C:\Users\Julian\Documents\uoc-boilerplate>rmdir .git /s /q
C:\Users\Julian\Documents\uoc-boilerplate>git init
Initialized empty Git repository in C:/Users/Julian/Documents/uoc-boilerplate/.git/
C:\Users\Julian\Documents\uoc-boilerplate>git add .
C:\Users\Julian\Documents\uoc-boilerplate>git commit -m "Commit inicial"
[master (root-commit) 4980fac] commit inicial
15 files changed, 9875 insertions(+)
create mode 100644 .editorconfig
create mode 100644 .gitignore
create mode 100644 .npmrc
create mode 100644 .posthtmlrc
create mode 100644 README.md
create mode 100644 package-lock.json
create mode 100644 package.json
create mode 100644 src/assets/fonts/.gitkeep
create mode 100644 src/assets/images/logo.png
create mode 100644 src/assets/scripts/main.js
create mode 100644 src/assets/styles/_variables.scss
create mode 100644 src/assets/styles/layouts/_home.scss
create mode 100644 src/assets/styles/main.scss
create mode 100644 src/index.html
create mode 100644 src/views/footer.html
C:\Users\Julian\Documents\uoc-boilerplate>
```

Instalación de dependencias:

Para este trabajo se ha utilizado **fontawesome** y **croppie**:

- *npm install croppie*
- *npm install --save @fortawesome/fontawesome-free*

También se utiliza **Stylelint**, que se instalará mediante el siguiente comando:

- *npm install --save-dev stylelint*

```
"devDependencies": {
  "@parcel/transformer-sass": "^2.8.3",
  "autoprefixer": "^10.4.13",
  "npm-run-all": "^4.1.5",
  "parcel": "^2.8.3",
  "postcss-preset-env": "^7.8.3",
  "posthtml-include": "^1.7.4",
  "rimraf": "^3.0.2",
  "sharp": "0.31.1",
  "stylelint": "^15.4.0"
},
"dependencies": {
  "@fortawesome/fontawesome-free": "^6.4.0",
  "croppie": "^2.6.5"
}
```

Por su parte Stylelint se ha añadido a las rutinas de UOC boilerplate de la siguiente manera:

- En el apartado "scripts" de package.json se ha añadido lo siguiente: "stylelint": "stylelint src/**/*.scss". De esta manera Stylelint comprobará los ficheros scss dentro de src.
- Además, se ha modificado la rutina build para que ésta pase la validación de Stylelint al compilar para producción. La rutina build ha quedado de la siguiente manera: "build": "npm-run-all clean stylelint parcel:build",

Prácticamente en todas las entregas habéis incluido fontawesome, aunque algunos de vosotros habéis utilizado otras dependencias, como por ejemplo: AOS, lazyframe, splide, animate.css, isotope, etc.

2. Decisiones de diseño

En el caso que nos ocupa se ha decidido implementar un enfoque **mobile first**., aunque ambos enfoques son perfectamente válidos e igual de funcionales para la presente práctica.

También se ha mantenido la nomenclatura utilizada en UOC boilerplate para los ficheros.

Se han utilizado ficheros HTML parciales mediante posthtml-include, al igual que sucedía en UOC boilerplate con el fichero footer.html. En este caso, además del fichero que ya existía, se han creado os ficheros más, header.html y main.html.

En este caso, las dependencias utilizadas para el desarrollo de la práctica han sido:

- Croppie: Herramienta para recortar imágenes.
- Fontawesome: Librería de iconos.
- Stylelint: Herramienta para comprobar las hojas de estilo.

3. Metodología elegida y/o guía de estilo

Pese a ser un trabajo de una sola página, se ha elegido la metodología BEM para su realización. Dicha metodología es muy útil en proyectos de mayor tamaño, de hecho, en proyectos de pequeño tamaño puede llegar a ser difícil de implementar.

Su objetivo es la división de la interfaz del usuario en bloques independientes. De esta forma se pueden crear componentes escalables y reutilizables. BEM se centra en los bloques, elementos y modificadores.

El siguiente ejemplo muestra el HTML del menú de la página en metodología BEM:

```
<nav class="nav">
  <ul class="nav__menu">
    <li class="nav__menu-item--seleccionado">
      <a class="nav__menu-item-enlace" href="#">Inicio</a>
    </li>
    <li class="nav__menu-item">
      <a class="nav__menu-item-enlace" href="#">Historia</a>
    </li>
    <li class="nav__menu-item">
      <a class="nav__menu-item-enlace" href="#">Enlaces</a>
    </li>
  </ul>
</nav>
```

También se han aplicado algunas reglas de la Code Guide de Mark Otto, disponible en: <https://codeguide.co/>. En este caso no se han aplicado todas, ya que algunas de ellas se tendrían que adaptar para combinarlas con BEM.

En la mayoría de las entregas habéis utilizado BEM. En algunos casos se han utilizado otras, como por ejemplo SMACSS, ITCSS o BEMIT (ITCSS + BEM).

4. SASS

En lo que respecta a SASS, se han utilizado variables, nesting, funciones, parciales e importación. Algunas de ellas como por ejemplo variables y parciales ya se utilizaban en UOC boilerplate.

- Variables: Se mantiene su definición en el fichero `_variables.scss`, al igual que se encuentra en UOC boilerplate.
- Nesting: Se ha utilizado el anidado del siguiente modo

```
.menu{
  &__item{
    font-weight: bold;
  }
}
```

- Parciales: Ya existían en UOC boilerplate. Se han creado parciales nuevos como por ejemplo `"_header.scss"` y `"_main.scss"`.
- Importación: Los ficheros parciales se han importado en `"main.scss"` mediante `@import`, como por ejemplo `@import "variables";`
- Funciones: Se han creado algunas funciones para calcular el tamaño de algunos elementos.

```
@function adjustmenu($width, $ratio){
  $new : $width * $ratio;
  @return $new;
}
```

En casi todas las entregas habéis utilizado estas características de SASS, sin embargo, en algunos trabajos habéis incluido también mixins.

5. Stylelint

Como se ha explicado anteriormente en el apartado de configuración del entorno, se ha realizado la instalación de Stylelint y su configuración para validar el código de los ficheros scss automáticamente al ejecutar la rutina build.

Para utilizar Stylelint con SASS, instalaremos stylelint-scss y la configuración recomendada stylelint-config-recommended-scss utilizando la siguiente instrucción:

- `npm install --save-dev stylelint-scss stylelint-config-recommended-scss`

Además, para comprobar la nomenclatura de BEM, añadimos la siguiente regla:

- `"selector-class-pattern": "^[a-z]([a-z0-9-]+)?(__([a-z0-9-]+-?)?(-([a-z0-9-]+-?)?){0,2}$"`,

También podemos instalar stylelint-config-recess-order para comprobar que seguimos el orden de las declaraciones de la Code Guide de Mark Otto:

- `npm install --save-dev stylelint-config-recess-order`

Además, como habéis hecho en algunos de los trabajos, podemos añadir otras reglas personalizadas, como por ejemplo las siguientes reglas que nos permiten cumplir con algunas de las recomendaciones de la Code Guide de Mark Otto:

- `"declaration-block-no-redundant-longhand-properties": true,`
- `"shorthand-property-no-redundant-values": true,`
- `"color-function-notation": modern,`
- `Etc...`

Un ejemplo de fichero ".stylelintrc" podría ser el siguiente:

```
{
  "extends": [
    "stylelint-config-recommended-scss",
    "stylelint-config-recess-order"
  ],
  "rules": {
    "selector-class-pattern": "^[a-z]([a-z0-9-]+)?(__([a-z0-9-]+-?)?(-([a-z0-9-]+-?)?){0,2}$",
    "declaration-block-no-redundant-longhand-properties": true,
    "shorthand-property-no-redundant-values": true,
    "color-function-notation": "modern"
  }
}
```

6. Repositorio y publicación

Repositorio en GitHub

Creemos un repositorio vacío en GitHub, asignándole un nombre, en este caso "prova".

Posteriormente subimos el repositorio local:

- `git remote add origin https://github.com/julianalarte/prova`
- `git push -u origin main`

En este caso, la URL del repositorio sería: <https://github.com/julianalarte/prova>

Publicación en Netlify

- En Netlify, hacemos clic en el botón “Add new site”.
- Posteriormente, “Import an existing Project” y lo elegimos de GitHub. Si no nos aparece, hacemos clic en “Configure the Netlify app on GitHub”.
- Con los parámetros por defecto, hacemos clic en “Deploy site”.

Site settings for julianalarte/prova

Get more control over how Netlify builds and deploys your site with these settings.

Owner

Julián ▼

Branch to deploy

main ▼

Basic build settings

If you're using a static site generator or build tool, we'll need these settings to build your site.

[Learn more in the docs](#) ↗

Base directory ⓘ

Build command ⓘ

npm run build

Publish directory ⓘ

dist

Show advanced

Deploying...

- Una vez hecho, dispondremos de la URL de Netlify.
- En “Site settings” comprobamos que tenemos la web en “Continuous deployment”.