Andorra la Vella!

P1

Adrián Galán Pacheco

UOC

Herramientas html y css II

Contenido

[1 Instalación del proyecto base 1](#_Toc211682578)

[2 Definición de entornos de producción y desarrollo 2](#_Toc211682579)

[2.1 Entorno de desarrollo 2](#_Toc211682580)

[2.2 Entorno de producción 2](#_Toc211682581)

[3 Soporte a navegadores antiguos 3](#_Toc211682582)

[3.1 Transpilación de JavaScript con Babel 3](#_Toc211682583)

[3.2 Procesamiento de CSS con PostCSS y Autoprefixer 3](#_Toc211682584)

[3.3 browsers 3](#_Toc211682585)

[4 Utilización de pre/postprocesadores 4](#_Toc211682586)

[4.1 Postprocesador PostHTML 4](#_Toc211682587)

[4.2 Preprocesador SASS 4](#_Toc211682588)

[4.3 Transpilación con Babel 4](#_Toc211682589)

[5 Dependencia externa 5](#_Toc211682590)

[6 Metodología, guía de estilo y decisiones de diseño 6](#_Toc211682591)

[7 Semántica y accesibilidad 7](#_Toc211682592)

[7.1 Estructura semántica 7](#_Toc211682593)

[7.2 Accesibilidad 7](#_Toc211682594)

[8 Creación y publicación a Git y GitHub 8](#_Toc211682595)

[9 Publicación a internet 8](#_Toc211682596)

# Instalación del proyecto base

El proyecto base utilizado fue UOC Boilerplate, descargado de su repositorio de GitHub mediante el siguiente comando:

git clone https://github.com/uoc-advanced-html-css/uoc-boilerplate.git

Seguidamente, se instalaron sus dependencias usando npm install.

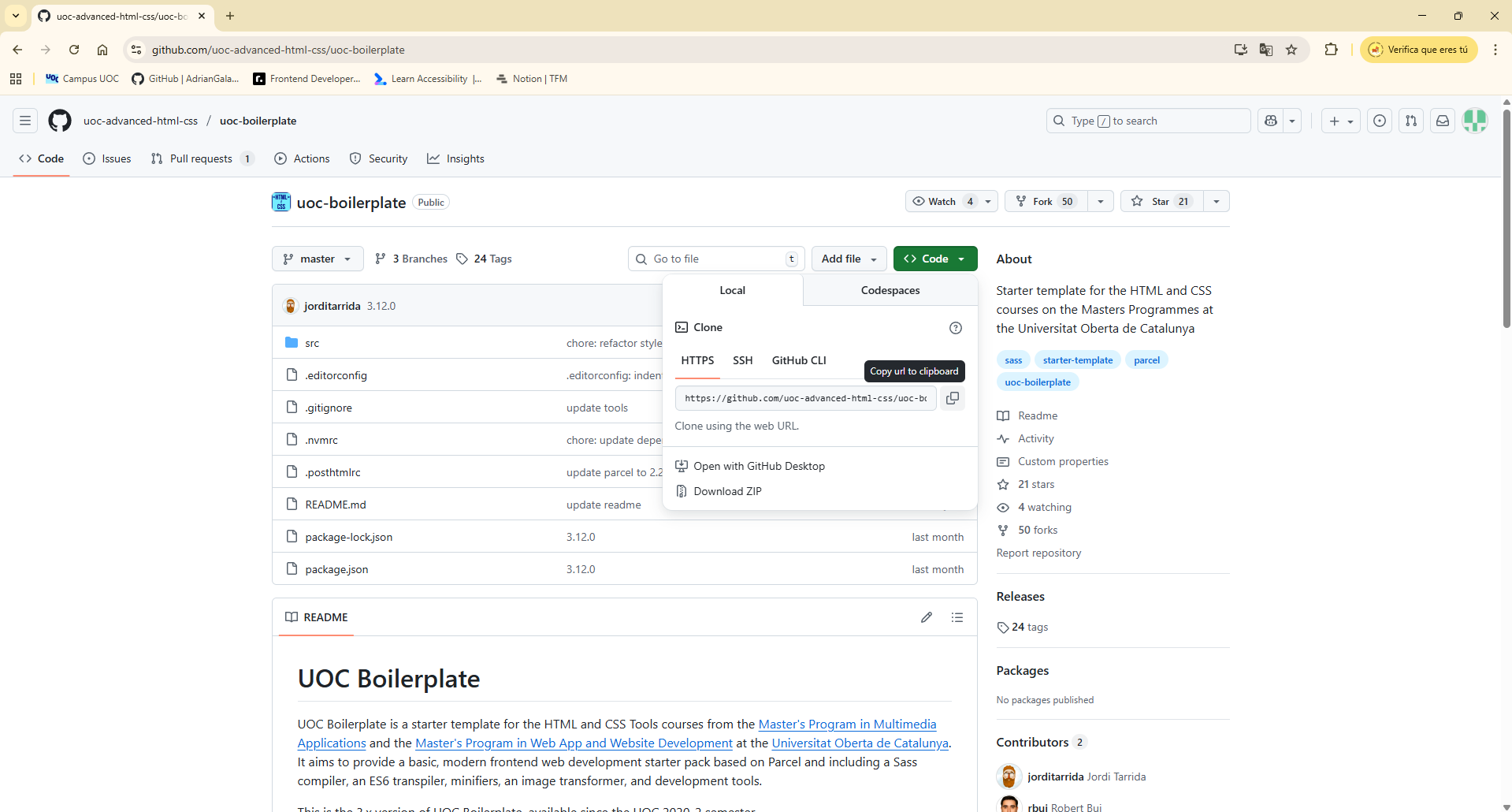


Ilustración 1. Página de GitHub del proyecto base.

# Definición de entornos de producción y desarrollo

## Entorno de desarrollo

El entorno de desarrollo es el espacio de trabajo local, diseñado para la optimizar la productividad y facilitar la depuración durante la fase de construcción del sitio web.

npm run dev

Este comando inicia el servidor local de Parcel, monitorea continuamente los cambios realizados y actualiza el navegador inmediatamente.

## Entorno de producción

El entorno de producción representa la versión final, optimizada y lista para desplegar en un servidor público. El objetivo de este entorno es garantizar el máximo rendimiento, velocidad de carga y compatibilidad para el usuario final.

npm run build

Este comando minifica, transpila y optimiza los *assets*.

El propio proyecto base incluye las dependencias rimraf y npm-run-all para limpiar el directorio dist antes de cada nueva generación y poder ejecutar múltiples scripts a la vez respectivamente.

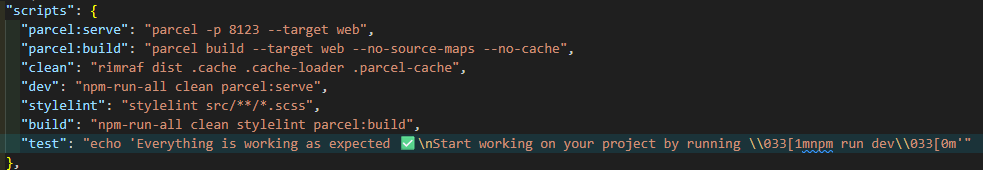


Ilustración 2. scripts de package.json.

# Soporte a navegadores antiguos

Para garantizar que el sitio web funcione correctamente en varios navegadores, incluidos aquellos que no son compatibles con las últimas versiones de CSS y JavaScript, el proyecto utiliza las capacidades de optimización automática de Parcel.

## Transpilación de JavaScript con Babel

El código de JavaScript del proyecto está escrito con la sintaxis de ES6+. Durante el proceso de compilación para producción, Parcel utiliza Babel de forma integrada para transpilar este código a su equivalente en ES5. Esta versión es compatible con navegadores más antiguos, asegurando que funcione sin errores.

## Procesamiento de CSS con PostCSS y Autoprefixer

Parcel procesa los ficheros de estilos con PostCSS y el plugin Autoprefixer. Esta herramienta analiza el código CSS y añade automáticamente prefijos necesarios para que ciertas propiedades de CSS3 funcionen en diferentes navegadores. Este proceso automatizado garantiza la compatibilidad del diseño sin necesidad de gestionar los prefijos manualmente.

## browsers

La configuración específica de los navegadores a los que se dará soporte se define en el archivo package.json mediante el parámetro browsers.

Esta configuración establece en herramientas como Babel y Autoprefixer qué versiones de navegador deben tener prioridad de compatibilidad y optimización. En este caso, navegadores que mantengan más del 0.5% de la cuota de mercado global, además de las dos últimas versiones de los principales navegadores.

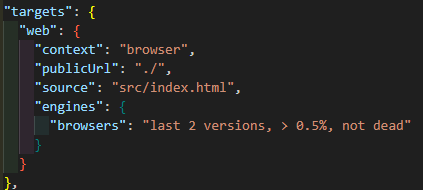


Ilustración 3. browsers de package.json.

# Utilización de pre/postprocesadores

## Postprocesador PostHTML

El proyecto utiliza una combinación de pre y postprocesadores para mejorar la modularidad, la organización y la compatibilidad con el código fuente.

Para evitar la repetición de código y crear una estructura HTML mantenible,el proyecto incluye el postprocesador PostHTML con el plugin posthtml-include.

Su configuración se realiza mediante el archivo .posthtmlrc en la raíz del proyecto para activarla.

## Preprocesador SASS

Para escribir hojas de estilo de manera organizada, modular y escalable, el proyecto incluye el preprocesador SASS. Esta herramienta permite usar características avanzadas como anidamiento, variables, mixins y parciales. Parcel gestiona la compilación a CSS de forma nativa.

## Transpilación con Babel

Aunque Babel es técnicamente un transpilador, actúa como un preprocesador al transformar el código fuente moderno antes de que se use en el navegador.

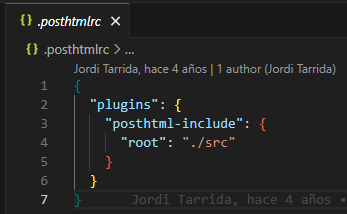


Ilustración 4. Archivo .posthtmlrc.

# Dependencia externa

El proyecto utiliza varias dependencias externas, instaladas a través de npm, para garantizar la accesbilidad y la incorporación de funcionalidades avanzadas.

Para añadir iconos se ha implementado:

* @fortawesome/fontawesome-free: Añade la biblioteca de iconos vectoriales Font Awesome para la iconografía del sitio.

Se instaló con npm install --save @fortawesome/fontawesome-free y se importa en el archivo de estilos de dependencias \_dependencies.scss.

La dependencia externa para aplicar funcionalidades avanzadas ha sido:

* leaflet: Se utiliza para aplicar mapas interactivos.

Se instaló con npm i leaflet y su funcionalidad se activa a través del *script* principal main.js.

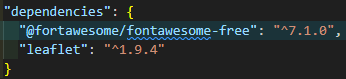


Ilustración 5. Dependencias de package.json.

# Metodología, guía de estilo y decisiones de diseño

El proyecto se construyó siguiendo la metodología BEM (*Block*-*Element*-*Modifier*) para estructurar tanto el HTML como el SASS, ya que permite aplicar modularidad, escalabilidad y previsibilidad. Al ser un sitio web de una sola página, BEM permitió tratar cada sección como un bloque, evitando colisiones entre clases. Además, BEM permite la capacidad de anidamiento de SASS utilizando la sintaxis &\_\_element y &--modifier.

La guía de estilos se implementó mediante la modularidad de SASS. Se separó el diseño en diferentes variables: color, tipografía y *breakpoints*. Esta práctica garantiza que el código sea mantenible y que todos los componentes accedan a una fuente para el diseño. Adicionalmente, se implementó la función opacity-lighten para realizar cálculos de color.

Para asegurar la aplicación de la metodología BEM en el proyecto, se configuró la herramienta Stylelint. En el archivo .stylelintrc se utilizó la regla selector-class-pattern con una expresión regular que refuerza a que todas las clases sigan el formato BEM. Esta configuración automatizada validó que el código cumpliera con la guía de estilo escogida, asegurando la consistencia del CSS.

Respecto a la toma de decisiones de diseño, se implementó la aproximación *Mobile First*. Los estilos se definieron sin *media queries* para garantizar la funcionalidad en dispositivos móviles. Después, el diseño se adaptó de forma ascendente con *media queries* para tablet y escritorio.

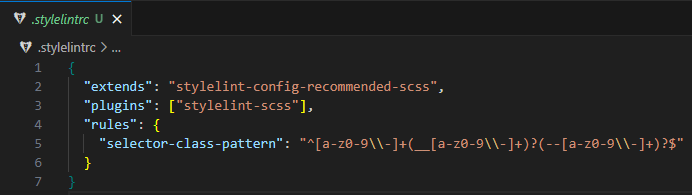


Ilustración 6. Archivo .stylelintrc.

# Semántica y accesibilidad

## Estructura semántica

Se ha construido el sitio web siguiendo la semántica HTML5. Se han utilizado los elementos adecuados para definir las secciones del contenido, garantizando que la web sea legible por motores de búsqueda y otras tecnologías:

Se utilizan etiquetas <header>, <main> y <footer> para delimitar las áreas principales de cada página y se utiliza la etiqueta <nav> para el menú de navegación.

Toda la información del sitio se organizó utilizando secciones semánticas. Cada tema es una etiqueta <section> con un id asociado que facilita la navegación interna.

En la sección de actividades, cada punto de interés se envolvió con una etiqueta <article> para indicar que es contenido independiente.

En la galería se utilizó la etiqueta <figure> para agrupar las imágenes con su respectivo <figcaption>.

Se respeta la jerarquía de encabezados (<h1> a <h6>), usando <h1> para títulos únicos de cada página y <h2> para los títulos de secciones importantes.

## Accesibilidad

Se han seguido los principios de WCAG para garantizar la accesibilidad en el sitio web.

Todas las imágenes contienen atributos alt descriptivos y concisos, lo que permite ayudar a usuarios con discapacidad visual y para SEO.

El *header* se ha diseñado para ser totalmente operativo sin JavaScript, utilizando la técnica del *checkbox* y el CSS.

# Creación y publicación a Git y GitHub

# Publicación a internet