# CONFIGURACIÓN DE LOS ENTORNOS

## 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.

## 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"

## 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.

## REPOSITORIO GIT

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.