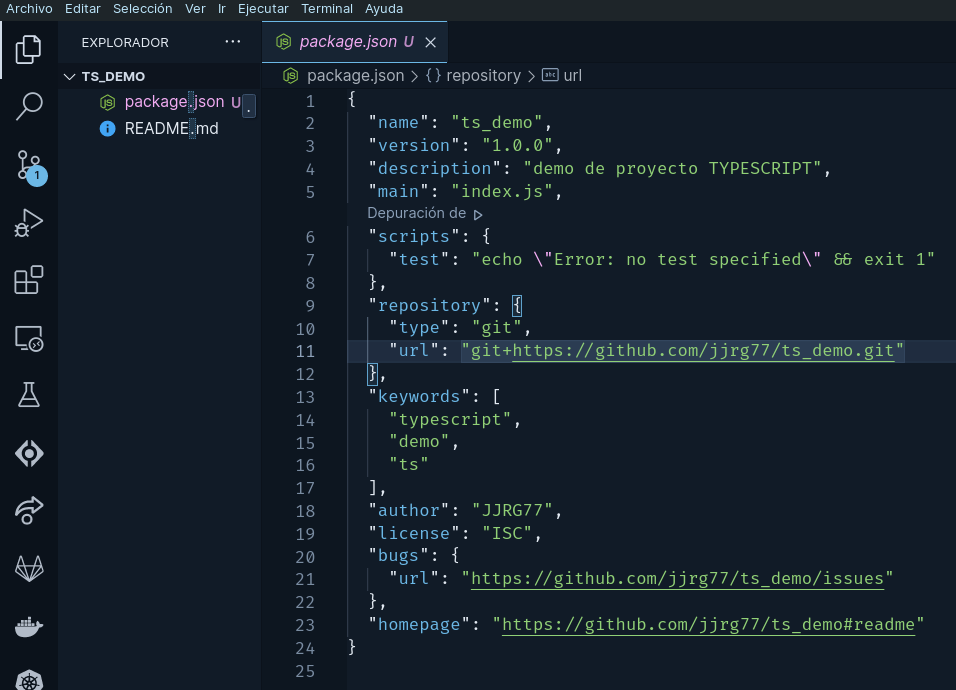
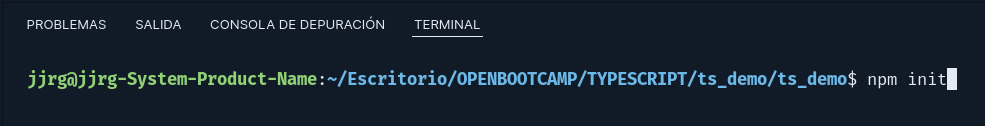
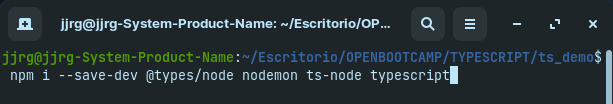
Para crear un proyecto de **TYPESCRIPT**, crearemos previamente, una carpeta en nuestro equipo donde alojar el proyecto (en nuestro caso será **ts\_demo**). Abriremos un terminal dentro de dicha carpeta o bien trabajaremos sobre el terminal que nos ofrece visual-studio-code.

1 – Tecleamos en la terminal **npm init:** este comando funciona como una herramienta para crear el archivo **package.json.**

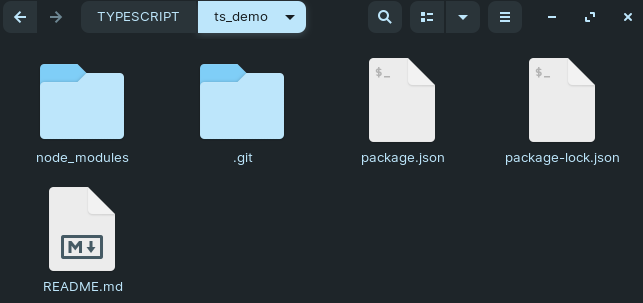
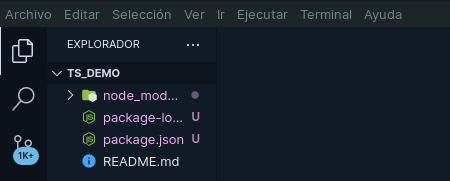
2 – A continuación, tendremos que contestar a las preguntas que nos plantee el sistema, el cual nos dará una opción entre paréntesis. Si fuera válida dicha opción, pulsaremos INTRO para que el sistema la acepte. Una vez aceptadas, los datos se guardarán en el archivo **package.json** que el sistema nos crea:



Creado este archivo, instalamos **HERRAMIENTAS DE DESARROLLO** mediante **npm.** En la terminal escribimos el siguiente comando:



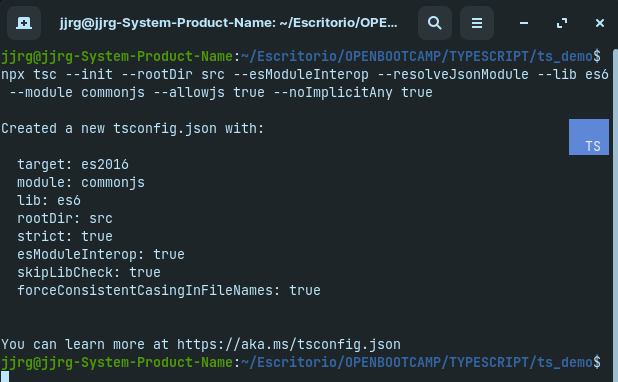
Con esta instalación, se nos generará en el proyecto la carpeta **node-modules**, donde se guardarán todas las dependencias instaladas. También se nos genera el archivo **package-lock.json**, el cual nos da información de las dependencias instaladas. Lo podemos comprobar, tanto en el editor de código como en nuestro equipo:



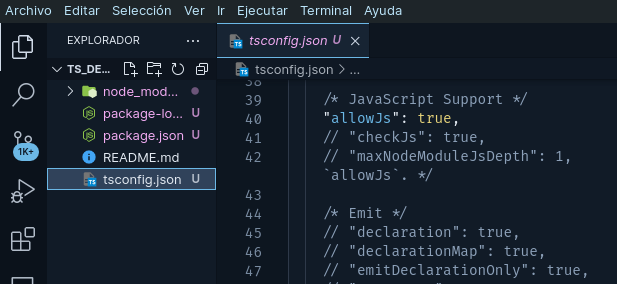
Analicemos los paquetes que hemos instalado:

* **@types/node:** este paquete contiene definiciones tipo para Node.js.
* **nodemon**: es una herramienta que refrescará que refrescará la aplicación cada vez que detecte algún cambio.
* **ts-node**: herramienta para ejecutar archivos **TypeScript,** de esta forma no necesitamos transpilar el código.
* **typescript:** instalamos **TypeScript**.

El siguiente paso es ejecutar **TypeScript**, para ello, debemos generar un archivo de configuración (**tsconfig**); para generarlo, escribiremos el siguiente comando en la consola:



Como nos informa la consola, se nos ha generado el archivo tsconfig.json, desde podremos cambiar alguna configuración del proyecto si fuera necesario:

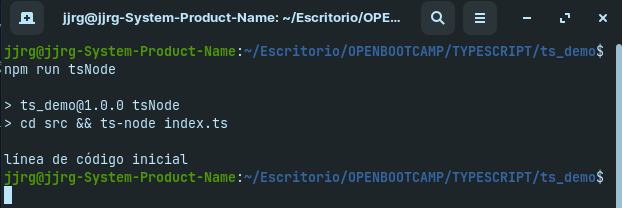
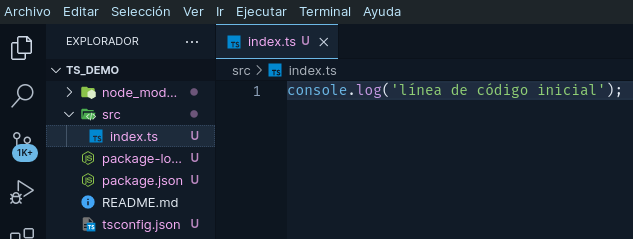
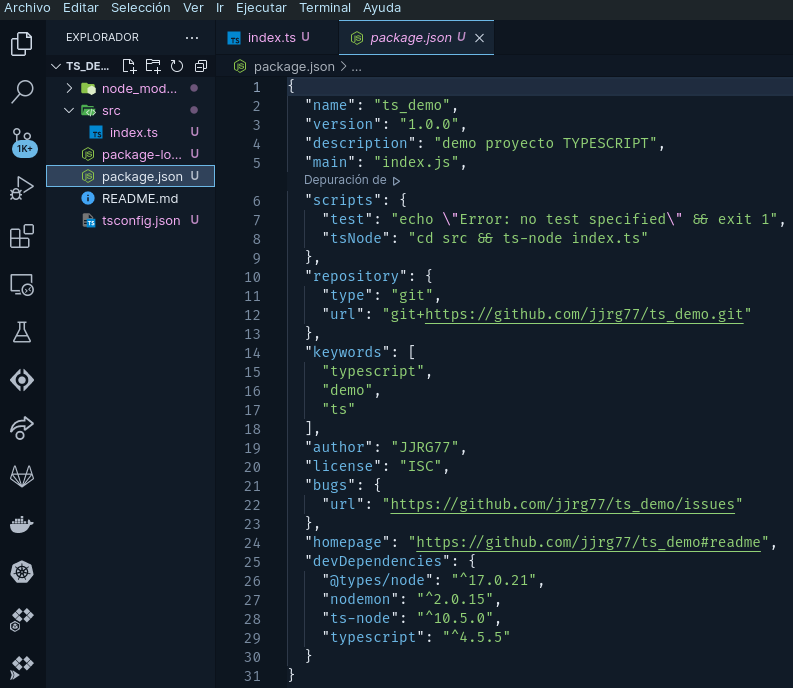


Analicemos ahora, cada elemento de la línea de comando que acabamos de escribir por consola:

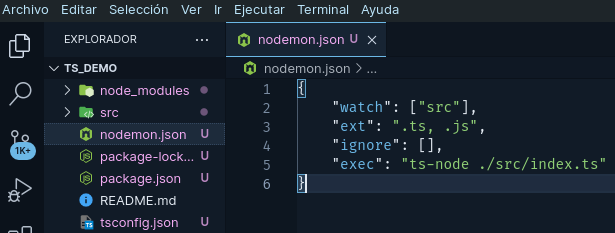
* **npx**: herramienta de **cli** que nos permite ejecutar paquetes de **npm** de forma local.
* **tsc**: compilador que se encarga de transpilar los archivos de **TypeScript** a **JavaScript**.
* **—init**: genera un archivo de configuración. Si no escribiésemos más comandos, nos lo generaría por defecto y tendríamos que modificarlo, por lo que añadimos el resto de comandos para que se instale con la configuración que queremos.
* **—rootDir**: especificamos la carpeta raíz dentro de los archivos de origen.
* **—esModuleInterop**: con este comando, podremos utilizar en otro archivo del proyecto mediante **import/export** cuando creemos una clase, función, módulo... Esta opción tiene que estar en **true** (al escribirlo sin especificarle si es **true** o **false**, lo interpreta como **true** si en la línea de comando s hubiésemos escrito **—esModuleInterop false**, sería **false**).
* **—resolveJsonModule**: así indicamos que trabajaremos con archivos **.json** ya que el archivo de configuración que se nos crea en el proyecto es **.json** (**tsconfig.json**).
* **—lib es6**: indicamos qué versión de **JavaScript** vamos a utilizar en el proyecto (en este caso, **ECMASCRIPT6**).
* **—module commonsjs**: al trabajar con una librería de JavaScript moderna, con este comando indicamos que se pueda transpilar el código en todos los navegadores modernos.
* **—allowjs true**: permitiremos que en esta aplicación conviva JavaScript con TypeScript.
* **—noImplicitAny true**: eliminamos poder utilizar el tipo de dato **any** que tiene **JavaScript**; así, evitaremos que nuestras variables trabajen con distintos tipos de datos (como ocurre en JavaScript).

Realizado este **script** en consola, se nos genera un archivo tsconfig.json en el proyecto con todas las configuraciones escritas.

Ahora crearemos el primer archivo **.ts** en el proyecto. Para ello, crearemos primero una carpeta **src** (carpeta raíz) en la cual tendremos el código principal del proyecto. En dicha carpeta crearemos un archivo **index.ts**. Dentro de este archivo, escribiremos un **console.log.** Para ejecutarlo, crearemos un script en **package.json**; por último, en la consola ejecutaremos el archivo **index.ts** con el script creado:

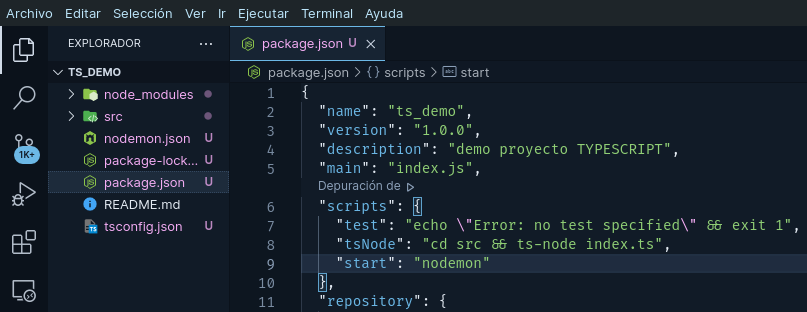


Esta forma de ejecución, nos obliga a escribirla en consola cada vez que haya una modificación. Para que se refresque automáticamente la aplicación cuando haya una modificación, crearemos en la raíz del proyecto un archivo un archivo **nodemon.json** con los siguientes comandos:

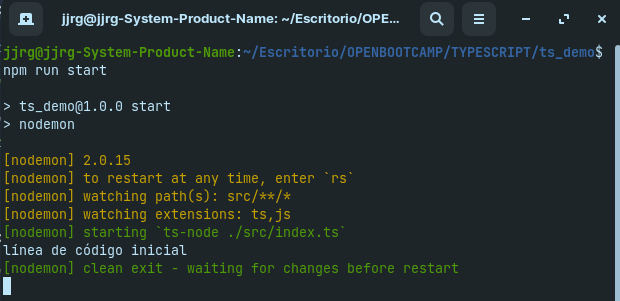


* **“watch”: [“src”]**: escucha la carpeta **src**
* **“ext”: “.ts, .js”**: para los archivos **.ts** y **.js**
* **“ignore”: []**: ignorar, de momento lo dejamos vacío, dentro pondremos más adelante lo que debe ignorar
* **“exect”: “ts-node ./src/index.ts”**: ejecuta el archivo **index.ts** que se encuentra dentro de **src**

Creado **nodemon.js**, nos vamos al archivo **package.json** y creamos el siguiente script:

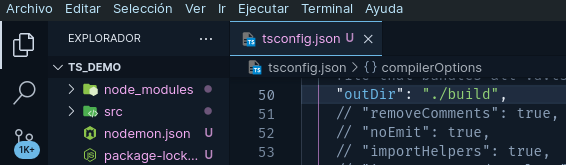


Así, al escribir por consola **npm run start**, la aplicación se refrescará cada vez que haya una modificación:



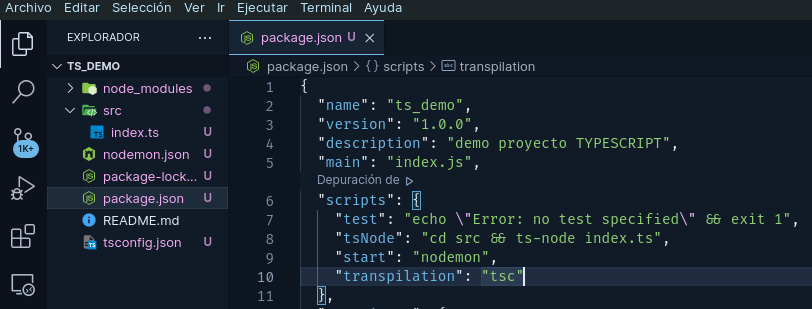
De esta forma, la consola se queda “escuchando” por si hay algún cambio, y si lo hubiere, lo escribiría por consola automáticamente. Para salir de esta escucha, debemos presionar, en consola la tecla **CONTROL** + **C**.

Anteriormente, en la creación del archivo **tsconfig.json**, no indicamos el parámetro **outDir**, cuya función es especificar una carpeta de salida para todos los archivos emitidos. Ésto lo necesitamos para transpilar el código transformando dicho código **TypeScript** a **JavaScript**; genera una carpeta **build** y un archivo **index.js** que contendrá el código transpilado de **index.ts**.

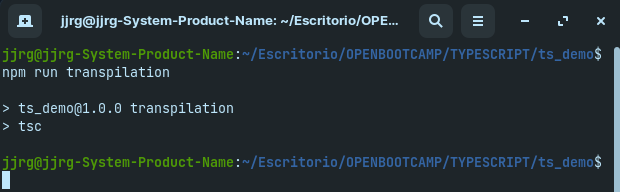


Así, nos vamos a tsconfig.json para definir **outDir**:

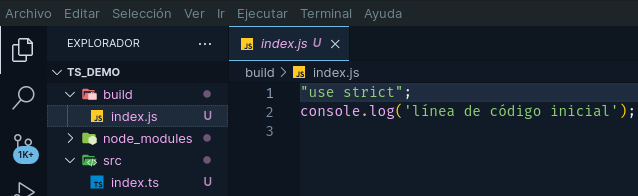
A continuación, en el archivo **package.json** escribimos el siguiente **script**:



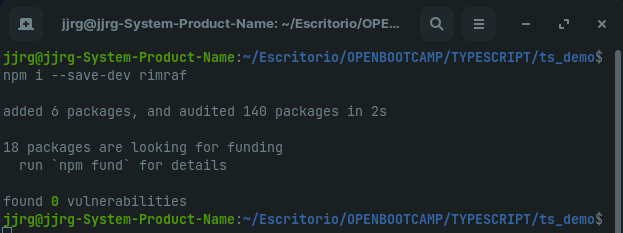
Ahora, escribimos por consola **npm run transpilation**; se nos creará la carpeta **build** y el archivo **index.js** y de esta forma hemos transpilado el código para que sea leído en el navegador:



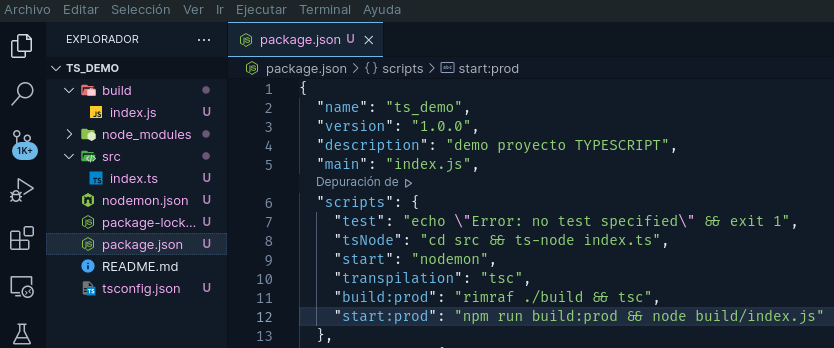
Como podemos observar, se ha creado la carpeta y dentro del archivo **index.js** ha metido el código (transformado a **JavaScript**) que había en **index.ts**:



Por último, vamos a instalar una herramienta (**rimraf**) que nos va a permitir crear un **build** más eficiente para una página web. En la consola escribimos:



Una vez instalada, nos vamos a **package.json** y creamos dos **script** para poder utilizarla. Con el primer **script** le indicamos que creamos una **build** y después lo **transpilamos**. El segundo **script** lo utilizamos para ejecutar la herramienta por consola:



Ahora lo podemos ejecutar. Escribimos en consola:

