1º Proyecto Typescript

1º Introducción

2º Descarga e instalación

3º Práctica

4º Conclusiones

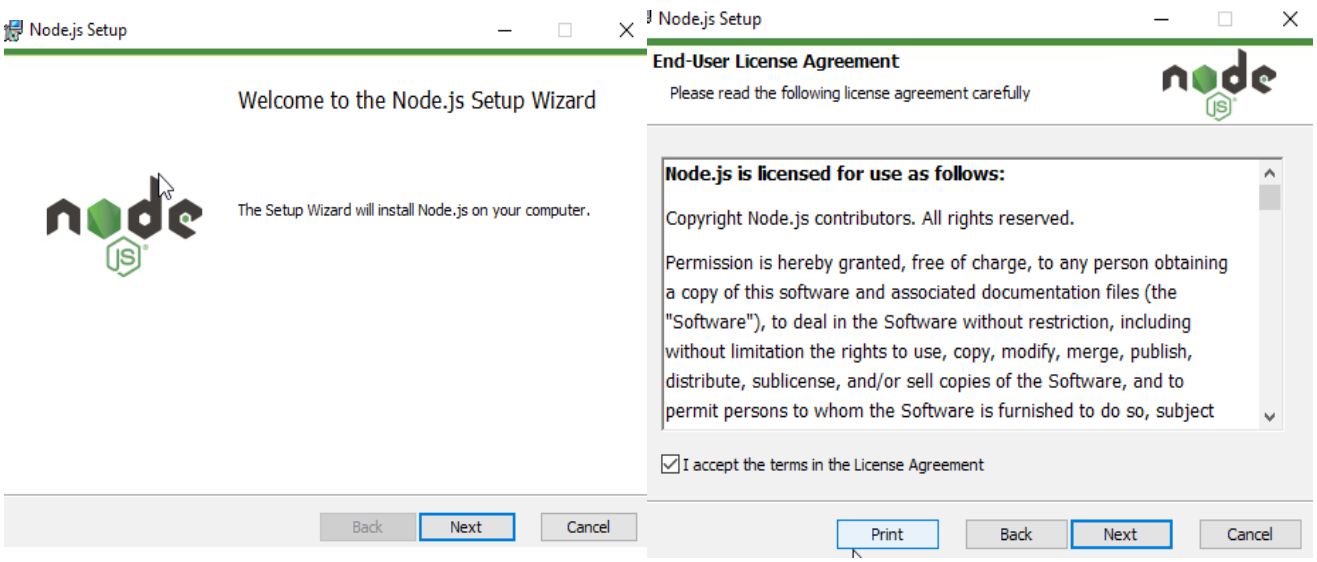
1º Introducción

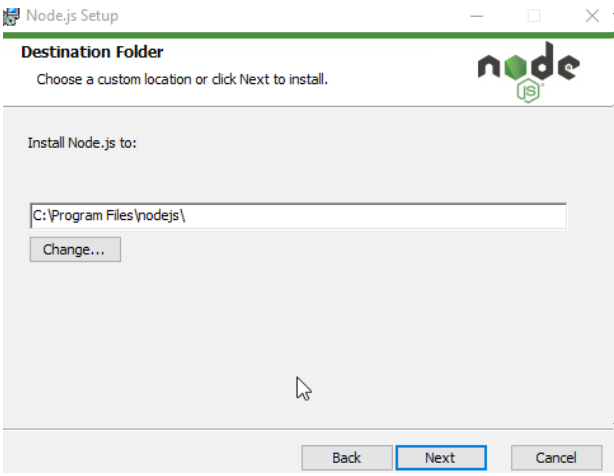
En este trabajo vamos a aprender donde y como instalar typescript y node.js para poder trabajar con ellos en nuestros futuras prácticas, despues de la explicación se mostrará un ejercicio de typescript.

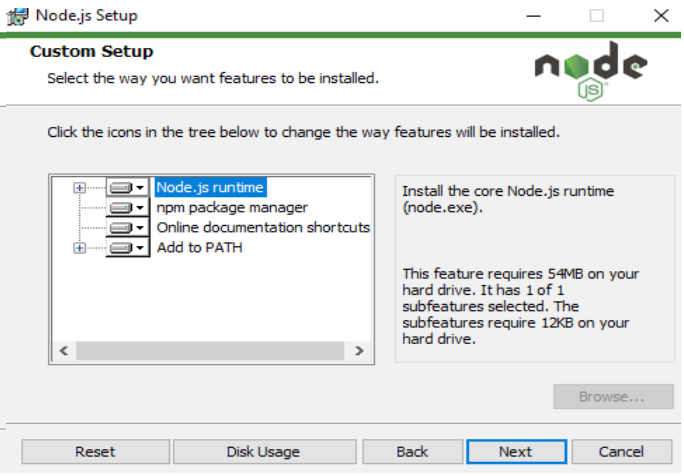
2º Descarga e instalación

Para poder descargar accedemos al siguiente enlace https://nodejs.org/es/ y

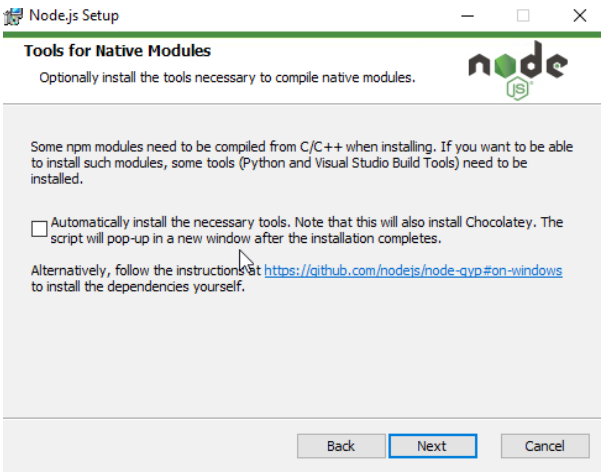
Y comenzamos con la instalación

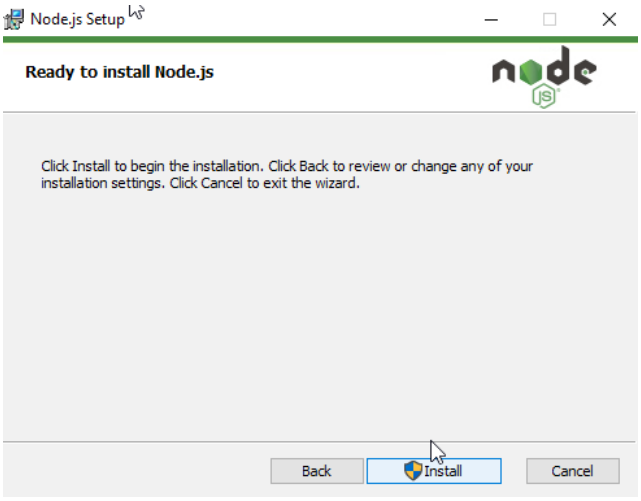




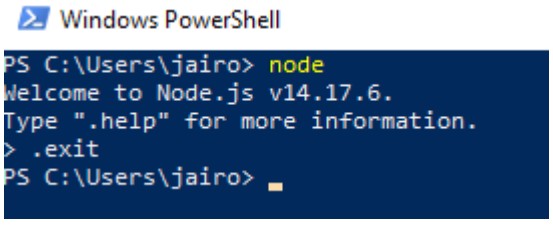


Y le damos a instalar

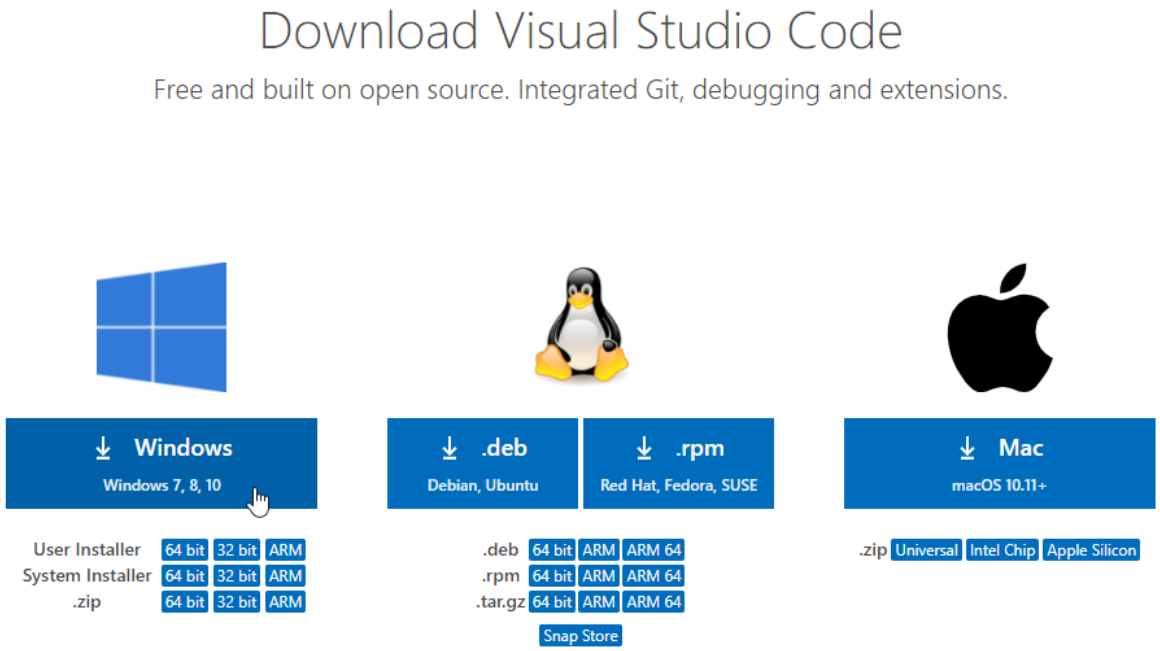




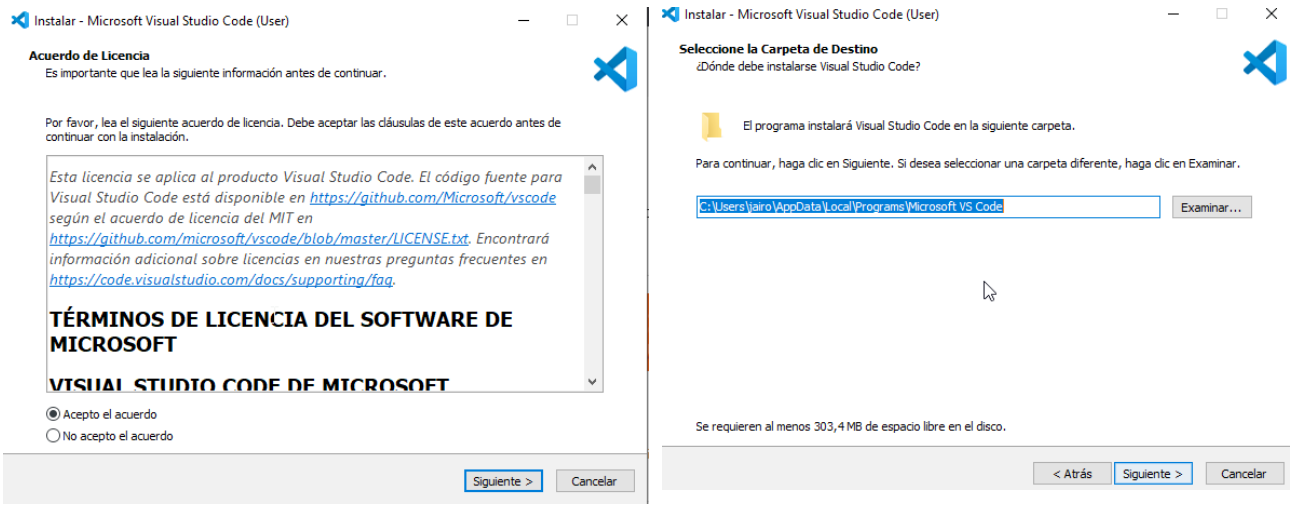
Si accedemos a un terminal (en este caso PowerShell de windows) y escribimos “node” nos muestra que podemos usar el comando .help, lo cual indica que dicho comando ha sido reconocido y por lo tanto node.js está instalado.

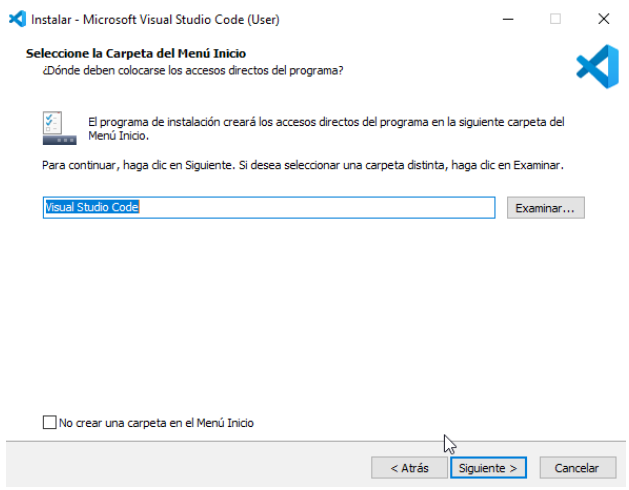


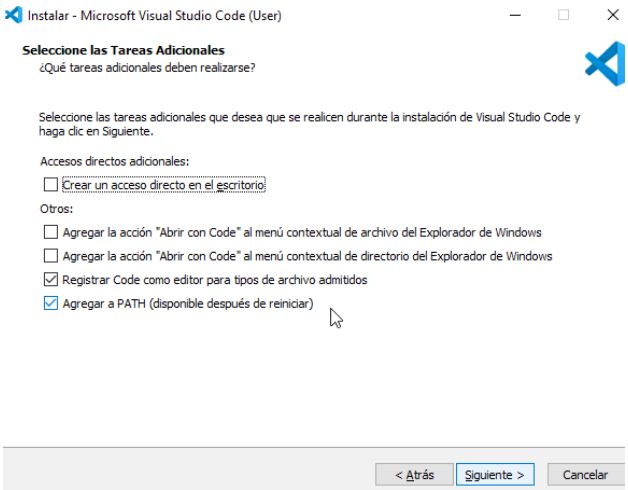
Ahora vamos a instalar “Visual Studio Code” que es el programa que utilizaremos para trabajar con nuestro código y en futuros proyectos, se puede descargar en la página https://code.visualstudio.com/download

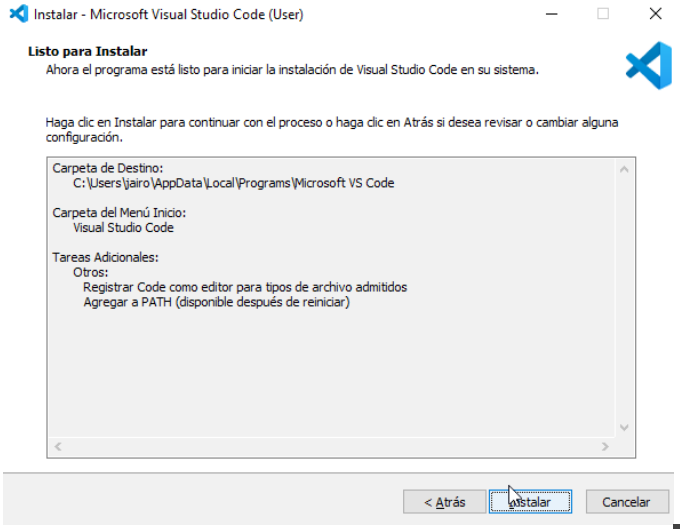


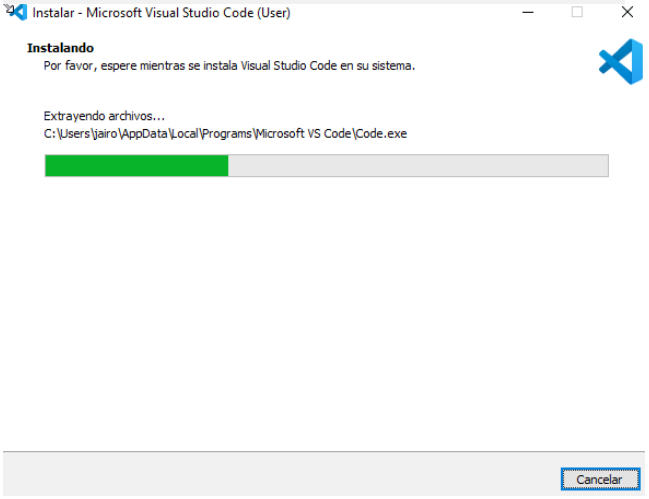
Ahora vamos a ejecutar el instalador y mostraremos las opciones que hay que ir marcando.

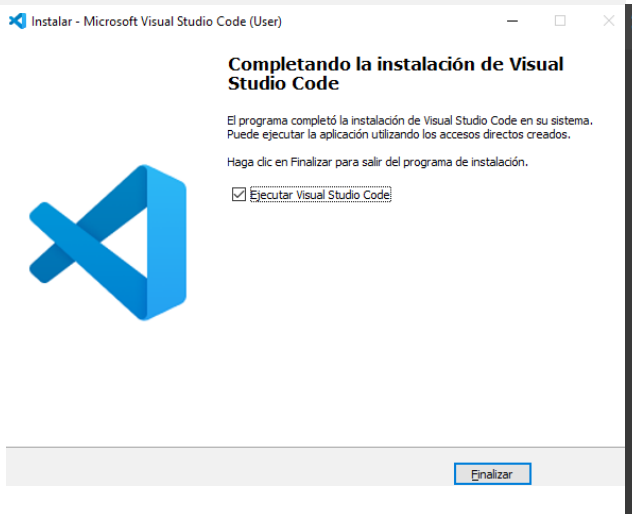


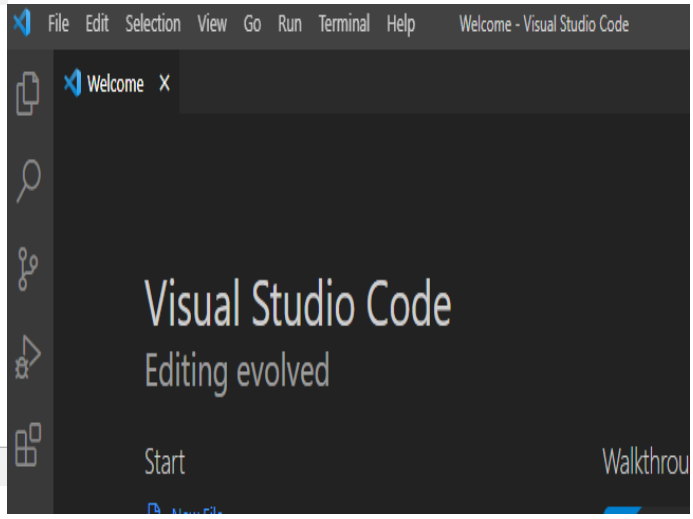






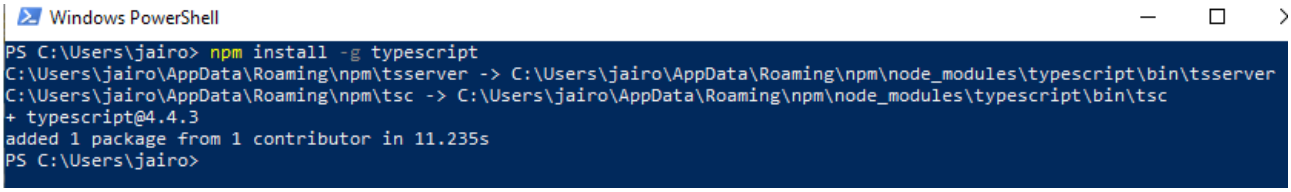






Ahora que ya se ha instalado y podemos abrirlo,vamos a proseguir instalando más programas necesarios.

Para instalar typescript de manera global se usa el comando “npm install -g typescript” de esta forma no tienes que instalarlo en cada proyecto que quieres usarlo.

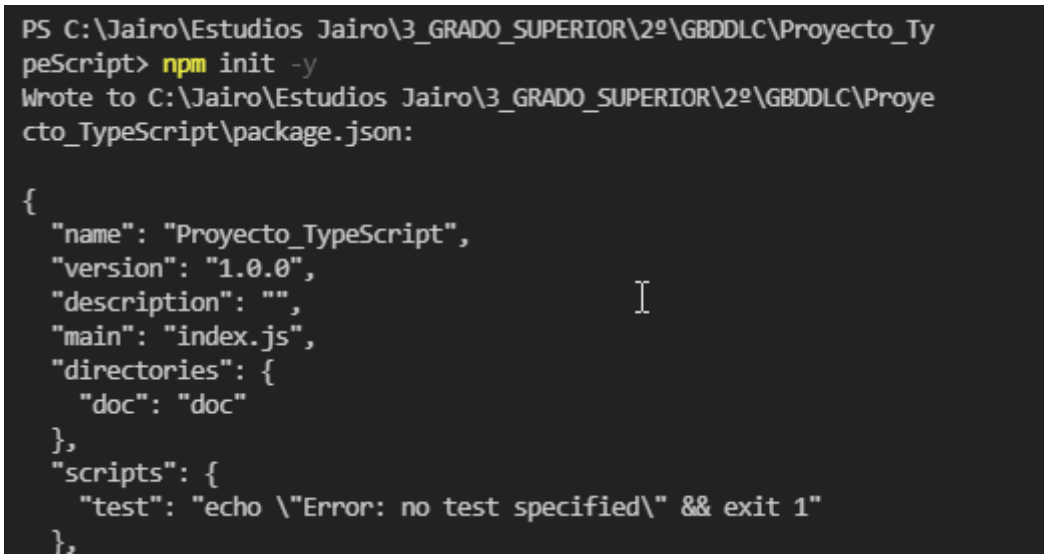


3º Práctica

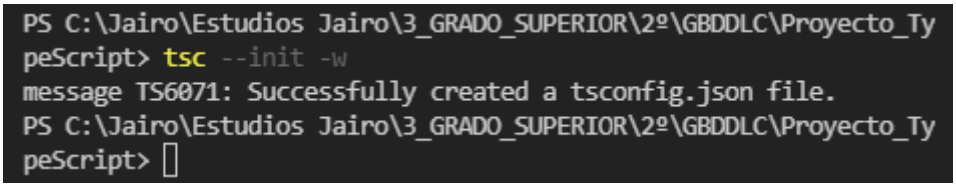
Ahora que ya tenemos node.js instalado vamos a abrir “Visual Studio Code” para trabajar en nuestra práctica.

Como todos nuestros proyectos nuestro archivo tendrá la estructura básica de tres archivos iniciales: Src, Doc y Readme.md, ademas de los que necesitemos añadir para poder seguir con la práctica, que instalaremos a continuación.

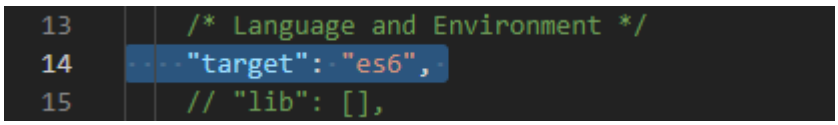
Primero vamos a instalar el “packagejson” donde quedará registrado los paquetes y modulos que vallamos instalando.



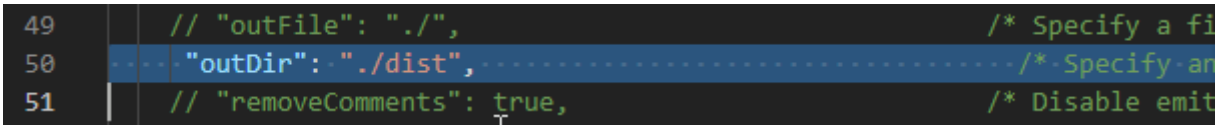
Ahora usamos el comando tsc –init -w, creará el archivo tsconfig.json, en el cual a continuación haremos dos cambios para poder continuar la práctica.



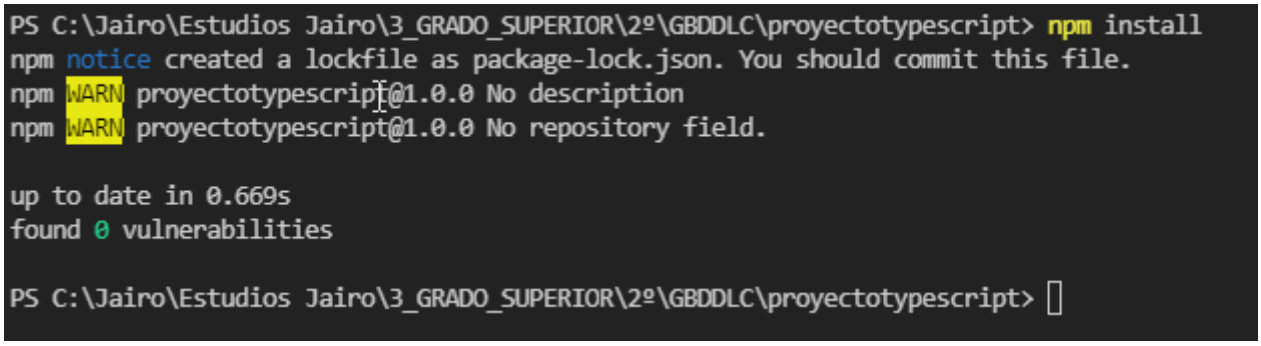
Dentro del archivo tsconfig.json, buscamos la línea 14 en la que se puede ver ‘”target”: “es5”’ y cambiamos el 5 por un 6.



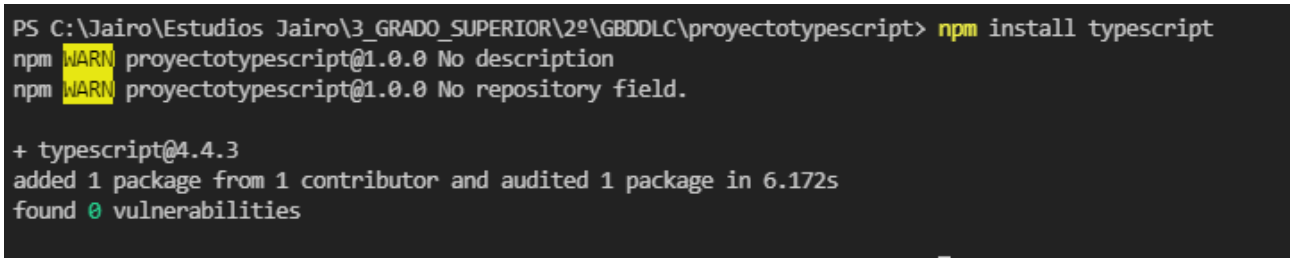
Por otro lado en la línea 50 o en sus alrrededores buscamos ‘//”outDir:” “./”,’ comentada, debeom



Ahora vamos a ejecutar npm install para que se genere el archivo “package-lock.json”



A continuación ejecutamos “npm install typescript” para generar la carpeta node_modules



A continuación crearemos la siguiente estructura:

Se crea dentro de src una carpeta de nombre util que contendrá un archivo de nombre “entradaTeclado.ts” y dentro de src se encontrará el archivo index, con el cual vamos a trabajar más adelante.

Es posible que algunas partes del código del archivo entradaTeclado salga en rojo, sin embargo lo solucionaremos instalando un modulo.

En tal caso emplearemos el comando “npm i –save-dev @types /node”

```
PS C:\Jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\proyectotypescript> npm i --save-dev @types
/node
```

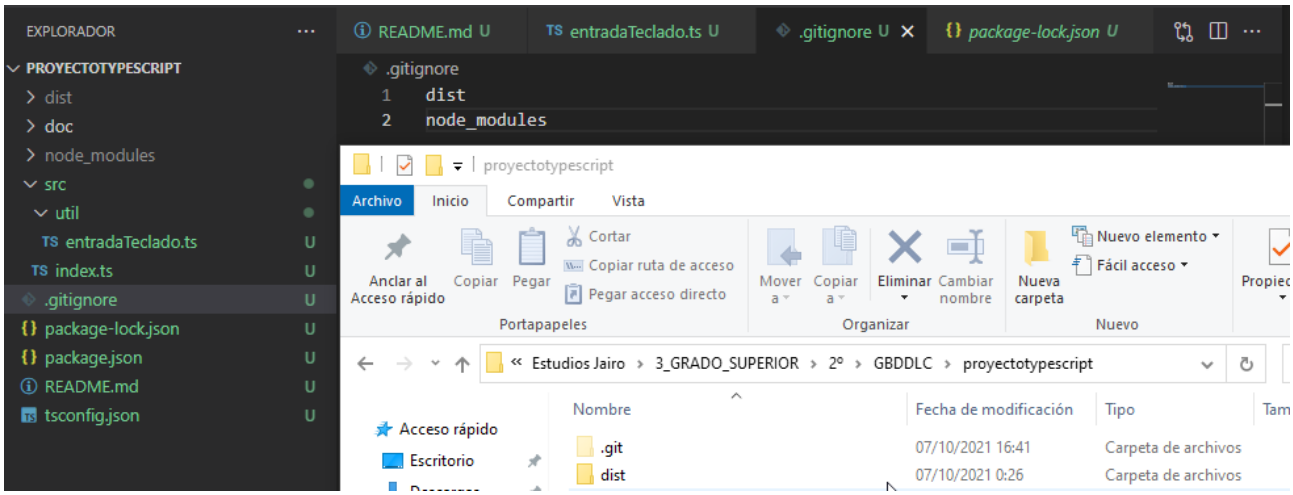
El contenido del archivo “entradaTeclado” es :

```
// https://nodejs.org/dist./v4.8.3/docs/doc/api/readline.html
import readline from 'readline'
let readlineI: readline.Interface

let leelinea = (prompt: string) => {
    readlineI = readline.createInterface({
        input: process.stdin,
        output: process.stdout,
    })
    return new Promise<string>( (resuelta: any, rechazada: any) =>{
        readlineI.question(`${prompt}:`, (cadenaEntrada: string) => {
            resuelta (cadenaEntrada)
        }
    )

})}

export let leerTeclado = async (prompt: string) => {
    let valor:string
    valor = await leelinea(prompt)
    readlineI.close()
    return valor
}
```

Mientras que el archivo index antes de empezar a construir el código que queremos emplear, usaremos el import de la siguiente manera.

```
1   import {leerTeclado} from './util/entradaTeclado'
2   //Creamos la variable función main que llamamos main
```

Ahora vamos a convertir la carpeta del proyecto en un repositorio que subiremos posteriormente a GITHUB para su posterior entrega, así que vamos a usar el comando “git init”, con lo cual crearemos en la carpeta una carpeta llamada .git que transforma nuestro directorio en repositorio, ahora añadimos un archivo de nombre “.gitignore” en el cual pondremos el nombre de los directorios y archivos que queramos que se omitan (dist y node_modules) a la hora de subir el contenido del repositorio.



Una vez echo todo esto, vamos a comenzar con el ejercicio, primero vamos a generar un archivo llamado “menu.ts”, el cual sera precisamente un menú con el que el usuario tendrá que interactuar.

Como vemos en la imagen importamos la entrada por teclado y exportamos el menu, para que en el archivo index que veremos a continuación podamos importar el menu y así no tenerlo en el mismo archivo.

src > util > TS menu.ts

```ts
import { leerTeclado } from '../util/entradaTeclado'

export const menu = async () => {
    let m: number
    console.log('\m')
    console.log('1.- Números Primos')
    console.log('2.- Comparar tres números')
    console.log('3.- Comprobar si un número es par o impar')
    console.log('4.- Formato de Fecha')
    console.log('5.- Salir')
    m = parseInt( await leerTeclado('opción: ') )
    return m
}
```

Ahora vamos a ver los diferentes puntos que se han desarrollado en el index, primero vemos la importación del menú y la entrada por teclado, ademas vemos una serie de variables y la constante main que hará que funcione este archivo.

```
src > TS index.ts > main
1   import { menu } from './util/menu'
2   import {leerTeclado} from './util/entradaTeclado'
3
4   const main = async () => {
5       let m:number
6       let primo: number
7       let n: number
8       let n1: number
9       let n2: number
10      let n3: number
11      let parimpar: number
12      let formato: number
13      let fecha: Date
14      let mes: string
```

Vamos a usar “do while” y un switch para las opciones del menú metiendo en do todas las opciones con los “case del switch” y en while la opción 5 de salir

```
do {
    m = await menu()
    switch(m){
        case 1:
```

```
}while (m !=5)
console.log("Adios hasta pronto")
```

Opción 1: Comprobar si un número dado es o no primo:

```
case 1:
    console.log("1.- Números Primos")
    primo = parseInt(await leerTeclado('Dame un número'))
    n=2
    while (primo % n !==0 && n < primo / 2){
        n++;
    }
    if (primo % n !==0){
        console.log(`El número ${primo} SI es primo`)
    } else {
        console.log(`El número ${primo} NO es primo`)
    }
    break
```

Opción 2: Pedimos tres números y el programa nos dice cual de ellos es el mayor

```
case 2:
    console.log("Necesitamos que escriba tres números que serán comparados y se le dirá cual es el mayor de ellos")
    n1 =  parseInt( await leerTeclado('Dame el primer número: '))
    n2 =  parseInt( await leerTeclado('Dame el segundo número: '))
    n3 =  parseInt( await leerTeclado('Dame un tercer número: '))
    if (n1 > n2 && n1 > n3) {
        console.log(`${n1} es el mayor de los tres`)
    } else if(n2 > n1 && n2 > n3) {
        console.log(`${n2} es el mayor de los tres`)
    } else {
        console.log(`${n3} es el mayor de los tres`)
    }
    break
```

Opción 3: Determinar si un número dado es par o impar

```
case 3:
    console.log("Porfavor escriba un número para determinar si es par o impar")
    parimpar = parseInt( await leerTeclado('Dame un número'))
    if (parimpar % 2 == 0) {
        console.log(`${parimpar} es un número PAR`)
    } else {
        console.log(`${parimpar} es un número IMPAR`)
    }
    break
```

Opción 4: Mostrar la fecha del día actual, eso sí dando a elegir entre dos opciones de formato

```
case 4:
    console.log("Le vamos a solicitar su día y mes de nacimiento")
    console.log("¿En que formato quiere la fecha de hoy?: 1- dd-mm-yyyy 2- mm/dd/yyyy")
    formato = parseInt(await leerTeclado('¿Que opción prefiere?'))
    if (formato == 1) {
        fecha= new Date()
        console.log(`La fecha de hoy es ${fecha.getDate()}-${fecha.getMonth()}-${fecha.getFullYear()}`)
```

```
    } else if(formato == 2) {
        fecha= new Date()
        mes = ""
        switch (fecha.getMonth()){
            case 0:
                mes = "Enero"
                break
            case 1:
                mes = "Febrero"
                break
            case 2:
                mes = "Marzo"
                break
            case 3:
                mes = "Abril"
                break
```

```
            case 11:
                mes = "Diciembre"
                break
        }
        console.log(`La fecha de hoy es ${mes}/${fecha.getDate()}/${fecha.getFullYear()}`)
    } else {
        console.log("El número que ha escrito no corresponde a ninguna opción")
    }
```

Ahora vamos a instalar un proyecto externo en nuestro repositorio local, para ello usaremos una ruta que acaba en una carpeta de nombre “para_instalar”

Primero vamos a convertir el directorio en un repositorio con git init.

```
PS C:\Jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\para_instalar> git init
Initialized empty Git repository in C:/Jairo/Estudios Jairo/3_GRADO_SUPERIOR/2º/GBDDLC/para_instalar/.git/
PS C:\Jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\para_instalar>
```

Usamos git clone, o descargamos de github cualquier proyecto o se lo pedimos a un compañero

```
PS C:\Jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\para_instalar> git clone https://github.com/ItziarEspino/ClubHipico.git
Cloning into 'ClubHipico'...
remote: Enumerating objects: 68, done.
remote: Counting objects: 100% (68/68), done.
remote: Compressing objects: 100% (48/48), done.
remote: Total 68 (delta 15), reused 68 (delta 15), pack-reused 0 eceiving objects:  44% (30/68)
Receiving objects: 100% (68/68), 647.37 KiB | 1.44 MiB/s, done.
Resolving deltas: 100% (15/15), done.
PS C:\Jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\para_instalar>
```

Por último instalamos el proyecto, usando npm install y ya podremos trabajar con el usando los módulos y elementos que el creador del proyecto haya usado en el.

```
PS C:\Jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\para_instalar> npm install
npm WARN saveError ENOENT: no such file or directory, open 'C:\Jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\para_instalar\package.json'
npm notice created a lockfile as package-lock.json. You should commit this file.
npm WARN enoent ENOENT: no such file or directory, open 'C:\Jairo\Estudios Jairo\3_GRADO_SUPERIOR\2º\GBDDLC\para_instalar\package.json'
npm WARN para_instalar No description
npm WARN para_instalar No repository field.
npm WARN para_instalar No README data
npm WARN para_instalar No license field.

up to date in 1.404s
found 0 vulnerabilities
```

4º Conclusiones

Hemos podido ver como instalar y poder crear una api con typescript y node.js desde el principio a fin, con ejemplos prácticos que iremos mejorando y aumentando la dificultad conforme aprendamos más en el curso.