UNIVERSIDAD SANTO TOMAS
PRIMER CLAUSTRO UNIVERSITARIO DE COLOMBIA
T U N J A

TÍTULO PROYECTO

Guías de conocimiento tecnológico para docentes: Serie Frontend Angular

PROPONENTE(S)
Juan Sebastián Ulloa Parra
C.C.1049656808
COD. 22511556

DIRECTOR
Carlos Andrés Guerrero Alarcón
Docente facultad de ingeniería de sistemas

Tunja
Diciembre 2021

CONTENIDO

1. FICHA TÉCNICA DEL PROYECTO

Título	Guías de conocimiento tecnológico para docentes: Serie Frontend Angular
Nombre Estudiante	Juan Sebastian Ulloa Parra
Documento estudiante	1049656808
Correo electrónico	Juan.ulloa@usantoto.edu.co
Director	Carlos Andrés Guerrero Alarcón
Lugar de ejecución del proyecto	Tunja- Universidad Santo Tomas Seccional Tunja
Duración	10 meses
Costo	$26’000.000
Palabras claves	Angular, NodeJS, CLI, Bootstrap, JWT, API, Servicios,

Los abajo firmantes confirman que todos los datos incluidos en la presente propuesta son correctos y verídicos, que no incumplen ninguna ley o norma vigente (incluir nombres y firmas de estudiantes y director).

Firma del autor
Nombre autor:
Juan Sebastian Ulloa Parra

C483438AC02BBBF5331F397DD5C05A8A6DDFF7750C4FDB22E2BE2406582DE9A56CB3D3DB80A936667BC33C1EB7650C40552052F3E73A8D3A0E6C08B60E3778AC.
25-nov-2021

Firma del director
Nombre director del proyecto:
Carlos Andrés Guerrero Alarcón

2. PLANTEAMIENTO DEL PROBLEMA

La evolución constante en las tecnologías de la información y las comunicaciones ha traído consigo la proliferación de cursos y presentación de materiales sin controles de calidad. Es así como se puede encontrar en internet material sin verificación de pares y sin criterios mínimos de calidad para su reproducción, en algunos casos con errores de fondo en su fundamentación teórica. A nivel universitario, el material utilizado exige un alto nivel de calidad, es por esta razón por lo que las fuentes de información comunes como blogs, videos y documentos no son aceptadas como material de referencia para los docentes.

Al incremento de información sobre temas tecnológicos, se suma la creciente demanda de profesionales en el área de desarrollo de software, demanda que a la fecha no está cubierta, el ministerio de las TIC en Colombia espera formar cien mil desarrolladores en un plazo de dos años, un valor alentador sin embargo insuficiente con respecto a la realidad colombiana. Ante esta escasez, los docentes universitarios están migrando de la docencia a la industria, dejando vacíos importantes en la academia en los procesos de formación con calidad. Es común encontrar convocatorias de docentes en el área de sistemas desiertas con perfiles que no cumplen los mínimos requeridos.

Las universidades de alta calidad en Colombia hacen esfuerzos por contratar los mejores docentes para la formación de sus estudiantes, sin embargo, el material de calidad que se utiliza y que está en constante evolución a veces es insuficiente. En el área de sistemas y desarrollo de software, el material en conjunto con la pedagogía del docente conforma el núcleo que trabajo al cual un estudiante puede acceder. Se puede tener un excelente material sin un docente con pedagogía, o un excelente docente con un material deficiente, esa simbiosis es clave en los procesos de formación en tecnologías de desarrollo. Desde el punto de vista pedagógico, aunque existe el problema, no es del resorte de este trabajo de investigación, sin embargo, la calidad de los materiales en un área específica del conocimiento tecnológico puede llegar a ser interesante.

Este trabajo de grado ha identificado una oportunidad latente en el área de desarrollo de software, para los estudiantes de ingeniería de sistemas que tienen como línea de trabajo

el desarrollo a nivel de Frontend y Backend. Ahora bien, el desarrollo Frontend y Backend se puede hacer en muchas tecnologías, cada proveedor de servicios como Microsoft, Google, Facebook, Oracle, entre otras, tienen sus propias tecnologías. ¿Es factible que un solo docente tenga el conocimiento de todas estas tecnologías?, aunque es posible la respuesta generalizada es no, siempre existirá una nueva tecnología que excede nuestra competencia. Entonces, ¿existe el material de calidad adecuado para ser utilizado por docentes al tratar de formar a sus estudiantes en una nueva tecnología? Los proveedores presentan cursos de capacitación, sin embargo, dónde queda la verificación de ese material, dónde queda el proceso de creación de material con calidad. Una posible respuesta es “en los libros”, en el área de informática in libro de más de dos años es obsoleto, por consiguiente, no aplica para el área de desarrollo. En la **¡Error! No se encuentra el origen de la referencia.** se presenta un esquema global del árbol de problemas más ampliado para la presente propuesta.

Imagen 1. árbol de problemas identificado

Fuente: Autor

En términos generales el problema de fondo a cubrir en esta tesis de grado es la creación de material de alta calidad para docentes que deseen formar estudiantes en el área de desarrollo software, la serie que se propone cubrir esta tesis es el desarrollo Frontend con

angular, sin embargo, queda un espacio disponible para el resto de múltiples tecnologías en desarrollo Frontend y backend, las cuales podría ser realizadas por más estudiantes y docentes en formación universitaria.

3. JUSTIFICACIÓN

La ausencia de material de calidad es un hecho en el desarrollo de software, la continua evolución de las tecnologías hace que un material de calidad esté obsoleto en el transcurso de dos años, sin embargo, si este material se convierte en una guía de conocimiento podría ser actualizado por el docente y su grupo de trabajo con una periodicidad semestral, de tal manera que las guías de conocimiento sigan realimentándose en el tiempo, algo complejo para los libros en tan corto tiempo. Este trabajo de grado se justifica desde tres puntos de vista: académico, económico y tecnológico.

Des el punto de vista académico, este trabajo de grado permitirá al estudiante en formación aplicar las competencias adquiridas en la carrera en la construcción de plataformas software, además, podrá aplicar todos estos conocimientos a la creación de material inédito en el área de desarrollo para la tecnología seleccionada. Angular de la empresa Google, es una tecnología en constante evolución, por lo tanto, la generación de guías de conocimiento que permitan ser actualizadas de manera periódica y contengan material actualizado sobre el proceso de desarrollo Frontend, permitirá al estudiante exponer todo lo aprendido en la carrera en pro y beneficio de la mejora continua de la calidad del programa.

Desde el punto de vista económico el proyecto es viable, el estudiante formación debe cumplir con los requerimientos de la universidad para graduarse y, por otro lado, la universidad le proveerá las herramientas y recursos para que el estudiante pueda lograr con éxito su grado. Uno de los recursos que provee la universidad son los docentes, docentes con alto nivel de experiencia en el desarrollo de software frontend y backend.

Analizando la parte tecnológica, el proyecto también es factible, toda vez que las nuevas tecnologías son un reto en formación tanto para estudiantes, docentes como graduados,

pues deben actualizarse de manera continua para ser competitivos en un mercado cada vez más dinámico. Por lo anteriormente expuesto, este trabajo de grado plantea los siguientes objetivos para aportar un grano de arena a la solución del problema presentado.

4. OBJETIVOS

4.1. Objetivo General

Construir un cuerpo de conocimiento para soportar el proceso de enseñanza aprendizaje de la parte del desarrollo web que se presenta al usuario final, utilizando una metodología orientada a talleres que pueda ser utilizada por los docentes de la Facultad de Ingeniería de Sistemas de la Universidad Santo Tomás.

4.2. Objetivos específicos

Tabla 1. Objetivos específicos

Nro.	Objetivo específico
1	Diseñar un instrumento que permita la construcción de guías de conocimiento para la presentación de talleres sobre Angular 12, incluyendo: requerimientos, instalación, presentación de la metodología de las guías, talleres completos con explicación y recursos de apoyo para los docentes.
2	Desarrollar los talleres que harán parte de las guías de conocimiento para apoyar el proceso de formación en Angular 12, utilizando Bootstrap como librería para la construcción del maquetado del frontend y typescript como lenguaje de desarrollo.
3	Desarrollar una aplicación funcional recopile los conceptos presentados en todas las guías de conocimiento para permitirle al docente disponer de un material completo que incluya: Autenticación con consumo servicios REST, manejo de funcionalidades personalizadas para autorización y construcción de dos CRUDS para ejemplificar el manejo de ventanas flotantes y pestañas para organización de información.
4	Verificar las guías de conocimiento, utilizándolas en la asignatura Fullstack como material de apoyo para el docente y los estudiantes. Para el caso de los estudiantes se definirán dos grupos de apoyo, uno trabajará con las guías de conocimiento y apoyo del docente y el otro grupo trabajará solo con la orientación del docente, con el fin de analizar los resultados de utilizar el material construido.

Fuente: Autor

5. METODOLOGÍA

Para el desarrollo del proyecto se propone la metodología de desarrollo prototipado, la cual se reconoce por la construcción de un prototipo donde el cliente está en constante contacto visualizando el producto, para el desarrollo del proyecto el contacto será con los docentes de la facultad de ingeniería de sistemas de la Universidad Santo Tomas, donde se tendrá en cuenta los nuevos requerimientos y sugerencias en cada una de las faces incrementando la calidad del producto.

Fase 1: Construcción de instrumentos.

Construcción de instrumentos	
Entradas	• ejemplos de plantillas o tutoriales en formato PDF
Proceso	Para esta fase se definen las siguientes actividades: definición de la estructura de documento, la calibración de los instrumentos, construcción de formatos, logotipos y sello distintivo.
Resultados	• Documento con formato Word con las secciones para crear las guías de conocimiento

Fase 2: Análisis y selección de versiones actualizadas.

Análisis y selección de versiones actualizadas	
Entradas	• Sitio web nodeJS • Sitio web Angular • Sitio web Bootstrap • Sitio web PostgreSQL
Proceso	Una de las actividades de esta fase es verificar la versión LTS (Long Term Suport) de nodeJS, adicionalmente se comprueban las versiones del framework (Angular), Bootstrap, Express, Type Script y los demás paquetes necesarios en la versión actual. La otra actividad de esta fase es la comparación de los motores de base de datos MySQL con PostgreSQL.
Resultados	• Imágenes adicionadas a guías de conocimiento

Según (López Herrera, 2016) PostgreSQL se define como un sistema de gestión de bases de datos objeto-relacional (ORDBMS) fundamentado en Postgres. Por otro lado, PostgreSQL es un sucesor de código abierto del código original de Berkeley, adicionalmente es compatible con una gran parte de la Estándar SQL.

De acuerdo con (Navajas Manuel & & Pamplona Sonia 2009). Los frameworks se define como estructuras de código genérico y reusable, lo anterior con el objetivo de ayudar a los desarrolladores en tareas de programación y tiene como finalidad aumentar la productividad.

(Cámara Braña,2020) manifiesta que angular se define como un marco de trabajo creado, desarrollado y mantenido por Google. Adicionalmente emplea una arquitectura SPA para el desarrollo de aplicaciones Web. Por otro lado, la filosofía de programación es de tipo declarativo, lo cual le favorece más que el tipo imperativo al tener que interactuar con interfaces de usuario. Por otra parte, entre sus características se encuentra el aislamiento del código entre componentes, la inyección de dependencias y su orientación a servicios.

Fase 3: Construcción de Talleres Frontend

Construcción de Talleres Frontend	
Entradas	• Modelos • Mocks
Proceso	En la construcción de los talleres para la parte de Frontend se tiene propuestos una cantidad de talleres que serán acumulativos estos contendrán los contenidos temáticos para el área de frontend en Angular.
Resultados	• Taller instalación y prerrequisitos es un proyecto generado por la CLI de Angular el cual solo es ejecutado. • Taller de estructura proyecto generado por la CLI de Angular contendrá el uso del componente principal la creación de estilos y maquetado HTML. • Taller múltiples componentes proyecto generado por la CLI de Angular contendrá la creación de componentes que son cargados en el componente principal. • Taller instalación de librerías proyecto generado por la CLI de angular en este taller se hace uso de Bootstrap y Jquery creando dos componentes que usaran maquetado de este framework de diseño. • Taller uso de iconos y asignación de puerto proyecto generado por la CLI de angular en este taller se hará uso de la librería de iconos Fontawesome y se hace el cambio de puerto en el despliegue de los proyectos. • Taller de CRUD uso de modelos y pruebas de front, en este proyecto se genera el primer CRUD con su respectivo modelo y se hacen las pruebas de frontend con un mock.

	• Taller CRUD con imágenes, este proyecto contendrá un CRUD con imágenes donde para manipular las imágenes se hará uso del Base64. • Taller gestión de ruteo es el primer taller que hará uso del ruteo para hacer la navegación de una pequeña aplicación CRUD de dos modelos relacionados. • Taller módulos este proyecto contendrá el manejo de múltiples módulos con su ruteo, uso de ventanas modales, tabs, pipe ordenamiento etc. • Taller uso y consumo de web services en este taller se hará el consumo de una API sencilla en express, contendrá un inicio de sesión, ajuste de perfil de usuario y CRUD'S. • Taller construcción aplicación final es la recopilación de todos los talleres para la creación de un módulo de autorización y autenticación reutilizable RAAM con una Api en express con Middleware

De acuerdo con (Alamilla Hernández, Pérez Romero, et. Al, 2021) Los Web Services (Web Services) permiten enlazar, a través del uso de diferentes protocolos, cualquier tipo de tecnología o servicio en la web, obteniendo que aplicaciones legado se comuniquen con otras aplicaciones de una organización o de cualquier tercero.

Fase 4: Construcción de Talleres backend.

Construcción de Talleres backend.	
Entradas	• Talleres Frontend • Aplicación final frontend
Proceso	En la construcción de los talleres para la parte de Backend se tiene propuesto la creación de taller (express) con apoyo del marco de trabajo Type Script para ejemplos de clase y la creación de API express con middleware para aplicación final.
Resultados	• Talleres ejemplo clase en formato Type Script en estos se encontrará la conexión con la base de datos, se hará uso del patrón de diseño DAO para construcción de controladores, generación de las rutas y un archivo de ejecución al servidor backend. • Api express aplicación final con formato Type Script en esta aplicación estará la conexión a la base de datos con la Promise de PostgreSQL, creación de controladores con patrón de diseño DAO y TAO, creación de rutas, creación de middleware y archivo de ejecución del servidor.

Según (Llerena Ocaña, Fernández Villacres, et. Al, 2021) menciona que Type Script se refiere a un amplio mundo en el desarrollo, tanto en aplicaciones web, como aplicaciones móviles, Por otro lado, no pueden quedar de lado las aplicaciones de escritorio, además en

todas estas hay que destacar que la UI (Interfaz de Usuario) para un correcto uso. Adicionalmente tiene que ser apreciado por el mercado que influye directamente.

Fase 5: Prueba y verificación de los talleres.

Prueba y verificación de los talleres	
Entradas	• Talleres Frontend y Backend • Aplicación final
Proceso	Para la verificación de los talleres se hará por parte de un par externo, como la aplicación en la Asignatura de Fullstack con la colaboración del docente a cargo.
Resultados	• Documento con formato Word con plantilla de verificación

Fase 6: Abstracción de los talleres para la generación de guías.

Abstracción de los talleres para la generación de guías.	
Entradas	Talleres Frontend
Proceso	En esta fase se procede a generar cada una de las guías de aprendizaje, la verificación de estilo y el fondo de estás.
Resultados	• Guía instalación y prerrequisitos documento con formato Word contendrá el contenido de la instalación de nodeJS, Angular, motor de base de datos PostgreSQL y editores de código VSC y Atom. • Guía instalación de librerías con formato Word contendrá la generación de los talleres de estructura y múltiples componentes. • Guía de manejo de componentes con formato Word contendrá la generación del taller de primer CRUD con modelos y pruebas de front. • Guía de CRUD Imágenes con formato Word contendrá el taller de CRUD de imágenes con decodificación Base64. • Guía de gestión de rutas con formato Word contendrá la generación del taller gestión de ruteo. • Guía de Manejo de Módulos con formato Word contendrá la generación del taller de módulos. Guía de Manejo y consumo de servicios con formato Word tiene la generación del taller de uso y consumo de web services.

Fase 7: Preparación de recursos. Las actividades para esta fase son la construcción y uso de repositorios.

Preparación de recursos.	
Entradas	• Talleres Frontend • Talleres Backend • Aplicación final
Proceso	Las actividades para esta fase son la construcción y uso de repositorios.
Resultados	• Repositorios en Git Hub

Fase 8: Verificación de las guías de conocimientos.

Verificación de las guías de conocimientos.	
Entradas	• Guías de conocimiento
Proceso	Para la fase de verificación de las guías en la asignatura Fullstack y con el ciclo de desarrollo del ministerio de los tics.
Resultados	• encuesta de satisfacción en anexo documento

Fase 9: Entrega de la tesis. Para la fase de entrega se tienen en cuenta las actividades de creación del documento final como los posibles imprevistos.

Verificación de las guías de conocimientos.	
Entradas	• propuesta • guías de conocimiento • sugerencias
Proceso	Para la fase de entrega se tienen en cuenta las actividades de creación del documento final como los posibles imprevistos.
Resultados	• documento final con formato Word contiene el contenido y desarrollo de la tesis.

6. CRONOGRAMA DE ACTIVIDADES

Para el desarrollo de la tesis se tiene en cuenta las siguientes actividades que se desarrollarán en el tiempo estimado de 10 meses, para un desarrollo adecuado de cada una de las actividades propuestas, la Tabla 2 presenta el cronograma con las actividades propuestas y los tiempos de ejecución.

Tabla 2. Cronograma de actividades

Nro	Fase	Meses									
		Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov
1	**Construcción de instrumentos**										
1.1	*Definición de la estructura del documento*										
1.2	*Calibración de los instrumentos*										
1.2	*Formatos, logotipos y sello distintivo*										
2	**Análisis y selección de versiones actualizadas**										
2.1	*Verificación de versiones de angular, NodeJS y Express*										
2.2	*Comparación de Postgres y MySQL*										
3	**Construcción de los talleres Frontend**										
3.1	*Talleres de instalación y prerrequisitos*										
3.2	*Talleres para instalación de librerías*										
3.3	*Talleres para manejo de componentes*										
3.4	*Talleres para gestión de rutas*										
3.5	*Talleres para manejo de imágenes*										
3.6	*Talleres para gestión de CRUDS*										
3.7	*Talleres para manejo de Módulos*										
3.8	*Talleres para manejo de servicios*										
3.9	*Talleres para consumo de Servicios*										
3.10	*Construcción de aplicación final funcional*										

Nro	Fase	Meses									
		Feb	Mar	Abr	May	Jun	Jul	Ago	Sep	Oct	Nov
4	**Construcción de los talleres Backend**										
4.1	*Construcción de taller para ejemplos de clase*										
4.2	*Construcción de backend son middleware para Express*										
5	**Prueba y verificación de los talleres**										
5.1	*Verificación por parte de par externo*										
5.2	*Aplicación de los talleres en la asignatura Fullstack*										
6	**Abstracción de los talleres para la generación de guías**										
6.1	*Construcción de las guías*										
6.2	*Verificación de estilo*										
6.3	*Verificación de fondo*										
7	Preparación de recursos										
7.1	*Construcción de repositorios*										
7.2	*Uso de los repositorios*										
8	Verificación de las guías de conocimiento										
8.1	*Verificación en los talleres de la asignatura Fullstack*										
8.2	*Verificación de las Guías en el ciclo desarrollo Web de MINTIC*										
9	**Entrega de la tesis**										
9.1	*Construcción del documento final*										
9.2	*Imprevistos*										

Fuente: Autor

7. DESARROLLO DEL PROYECTO

7.1. Guía 1

7.1.1. Presentación

Angular es un marco de trabajo basado en JavaScript y soportado por Google. Su función es crear aplicaciones del tipo SPA (*Single Page Aplicación*) eficientes y sofisticadas. La estrategia de Angular es administrar el software desarrollado a partir de módulos y componentes, por lo tanto, comprender la estructura de una página en forma de componentes es la clave para entender este marco de trabajo.

Las razones por la cual Angular eleva la productividad de los desarrolladores están soportadas en:

1. AngularCLI, es un conjunto de herramientas en modo consola que permite crear cualquier elemento necesario en Angular.
2. El uso de plantillas, en Angular reutilizar plantillas html5 es una tarea que simplifica el desarrollo Web.
3. Escalabilidad, Angular está diseñado para que sus aplicaciones sean escalables de un proyecto pequeño a uno de nivel empresarial.
4. Desarrollar por componentes independientes hacer del proceso una tarea más eficiente.

Para complementar el proceso de desarrollo, el Frontend requiere de su contra parte en Backend, por lo tanto, es importante introducir el entorno de desarrollo de JavaScript más popular: Node JS.

Node JS es un entorno de ejecución de JavaScript con una estructura basada en eventos para aplicaciones escalables. Node JS se destaca por la comunicación con el protocolo HTTP con una baja latencia entre sus comunicaciones, lo cual lo convierte en un recurso atractivo para librerías o Frameworks web.

Como motor de persistencia, en estas guías de conocimiento se utilizará la base de datos PostgreSQL. Es el más grande sistema de bases de datos relacionales de tipo open source en la actualidad. Desde su lanzamiento, PostgreSQL ha demostrado ser confiable, asegurando la integridad en lo datos, la extensibilidad y su adaptabilidad a volúmenes de información altos con tolerancia a fallos.

7.1.2. Verificación de requerimientos

7.1.2.1 Acabando los mitos

Uno de los mitos más frecuentes en los tutoriales de Angular es la necesidad de utilizar GIT (software de control de versiones) para construir proyectos en Angular. Esto es incorrecto y genera en el aprendiz un poco de confusión. Por lo anterior, en esta guía se van a utilizar solo los elementos claves para construir una aplicación Web en Angular.

Para desarrollar en Angular no es necesario utilizar repositorios como GitHub o GitLab.

Idea clave

7.1.2.2 Características de la máquina a utilizar en desarrollo

La Tabla 3 presenta la información del equipo en donde se realizarán las pruebas de esta guía.

Ítem	Descripción
Procesador	Intel(R) Core (TM) i7-3770 CPU @ 3.40GHz
Memoria física total	16 gigas
Espacio disco disponible	480 gigas SSD
Adaptador de pantalla	AMD Radeon HD 5450
Sistema operativo	Windows 10 versión 21H1

Tabla 3. Características del equipo de desarrollo

7.1.3. ¿Por dónde empezamos?

Bien, a nivel de Front está Angular, el cual es un Framework para desarrollar en JavaScript, a nivel de servidor está Node, el cual es una plataforma que permite ejecutar JavaScript en el Servidor. ¿Cómo se puede comunicar Angular con Node?, existen varias respuestas, aunque la más utilizada hoy en día es Express.

Una estrategia muy acertada para desarrollar en Angular y Node, es utilizar TypeScript. TypeScript es un lenguaje de código abierto basado en JavaScript. La potencia de

TypeScript es su sencillez, el tipado de datos, la orientación a objetos y el cumplimiento de estándares como EcmaScript.

Resumiendo, vamos a utilizar en estas guías de conocimiento: Angular, NodeJS, TypeScript, PostgreSQL y algunos otros Frameworks que harán el proceso de desarrollo mucho más interesante

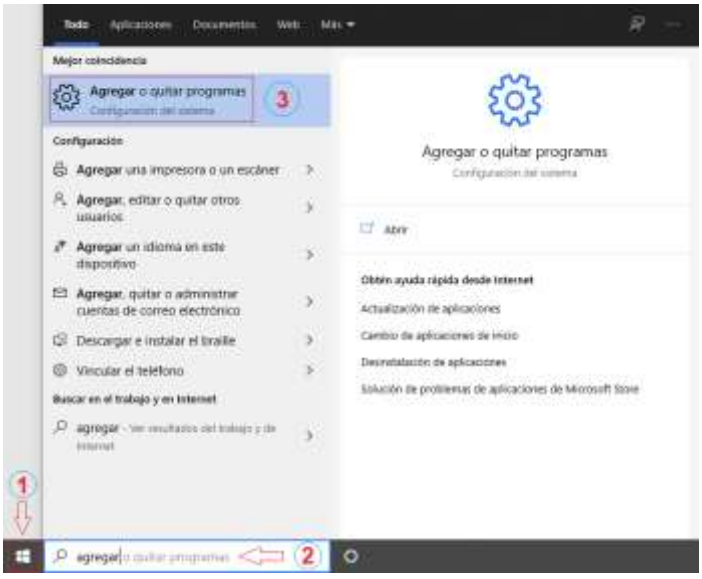
7.1.4. Instalación de herramientas

Si ha instalado versiones inferiores a la 14.17.5 LTS de Node JS, el siguiente apartado puede ser muy útil pues le permitirá remover las versiones antiguas e instalar una versión más actual de Node JS.

7.1.4.1 Remover versiones anteriores Node JS

Para remover instalaciones previas de Node JS se utilizará la herramienta agregar o quitar programas de Windows. La Imagen 2 presenta los pasos a seguir para lanzar la aplicación.

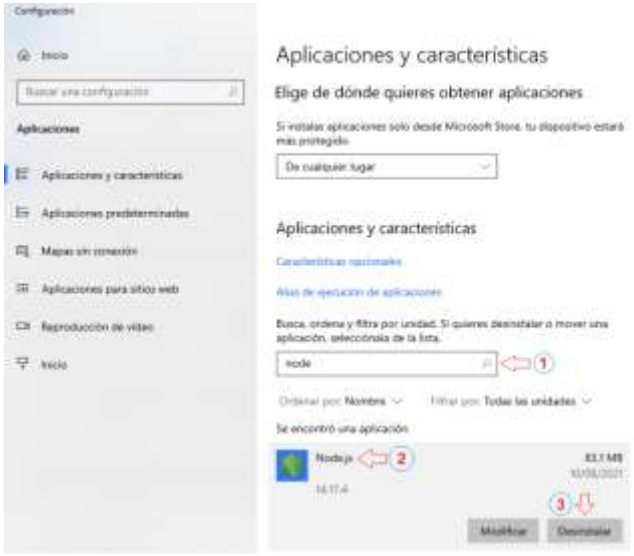
Imagen 2. Herramienta agregar o quitar programas



Fuente: (Autor, 2021)

La herramienta para eliminar programas de Windows presenta una caja de texto en donde se puede escribir el nombre del programa que se desea eliminar, con esto se filtran las demás aplicaciones y se previenen errores. La Imagen 3 presenta la interfaz de la herramienta para eliminar programas.

Imagen 3. Desinstalación de Node JS

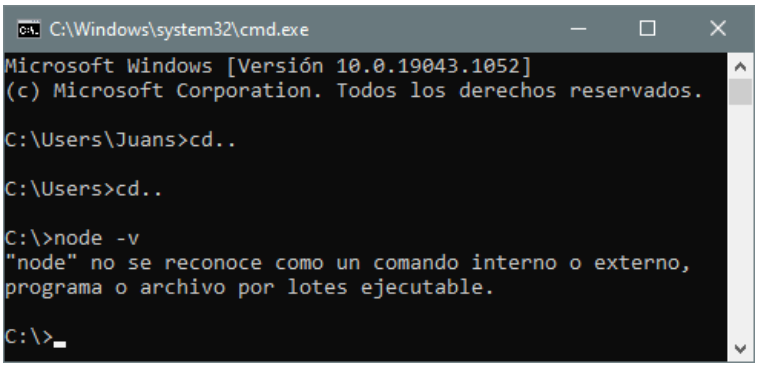


Fuente: (Autor, 2021)

Finalizado el proceso de desinstalación del Node JS, es posible verificar que efectivamente ha sido desinstalado, para esto, se debe abrir una consola de comandos de Windows (CMD). La Imagen 4 presenta la verificación de la desinstalación de Node JS. El comando para verificar la existencia de Node JS en el equipo es:

```
node -v
```

Imagen 4. Consola de comandos verificación desinstalación NodeJS

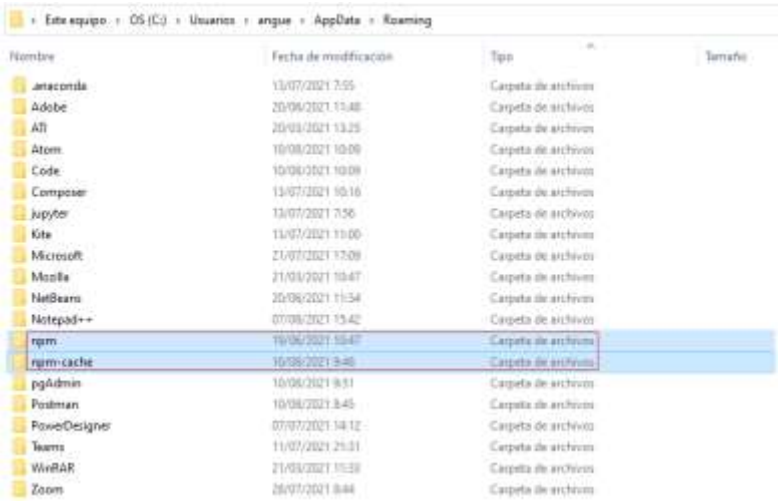


Fuente: (Autor, 2021)

Uno de los problemas frecuentes al desinstalar aplicaciones es la persistencia de archivos adicionales. La información adicional que no es eliminada generalmente corresponde a archivos de configuración o caché, los cuales son almacenados para mejorar la experiencia del usuario ante una posible nueva instalación. Sin embargo, es recomendable eliminar todo rastro de la versión anterior de Node JS y de posibles frameworks adicionales que

hayan sido instalados. La Imagen 5 presenta las carpetas que deben ser eliminadas manualmente para finalizar el proceso de desinstalación de Node JS.

Imagen 5. Explorador de archivos para eliminar carpetas NPM



Fuente: (Autor, 2021)

7.1.4.2 Instalación de Node JS

Node es ambiente para ejecutar código en JavaScript. Si una aplicación necesita o utiliza código en JavaScript es susceptible de utilizar Node para ejecutar código en un servidor a través de Ajax. Básicamente funciona con una arquitectura basada en eventos.

La potencia de Node JS se basa en la capacidad para gestionar la memoria, un sistema tradicional incrementa el uso de memoria a medida que un usuario se conecta, esto implica que la cantidad de usuarios conectados va a depender de la cantidad de RAM del equipo, por ejemplo, si un servidor tiene 32 gigas en RAM, tiene una alta probabilidad de tener capacidad para administrar máximo 16.000 usuarios, lo cual es un número aceptable. No obstante, ¿Qué pasa si los usuarios superan por mucho ese número? Acá es donde entra Node JS pues no utiliza el concepto de hilos, por el contrario, ejecuta en su motor un nuevo evento cuando se ejecuta un nuevo usuario, lo cual implica una optimización extrema del uso de la memoria, hasta el punto de permitir hasta millones de usuarios. Si es desarrollador de sistemas Web tradicionales, es posible que le interese este artículo: https://kinsta.com/es/blog/nginx-vs-apache/

Como se presenta en la Imagen 6, se puede descargar de forma gratuita del sitio: https://nodejs.org/es/

Imagen 6. Sitio Web oficial de NODE JS

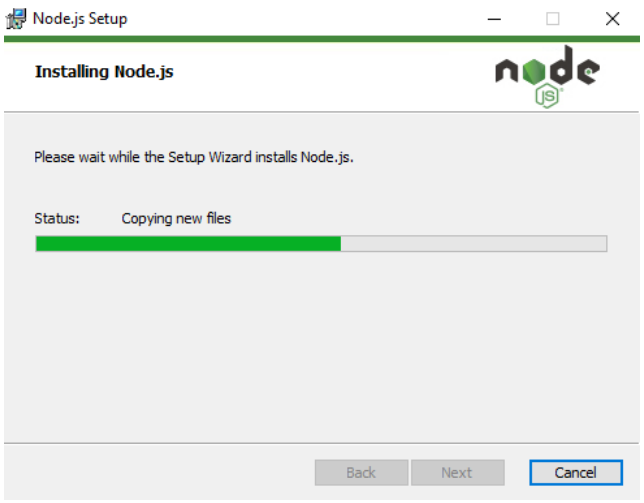
Fuente: (OpenJS Foundation, 2021)

En esta guía se utilizará la versión 14.17.5 LTS para Windows de 64 Bits. Si aún desea utilizar una versión previa de Node JS, asegúrese que sea una versión superior a Node JS 12.0, las versiones anteriores probablemente no funcionarán con este contenido.

¿Por qué usar una versión recomendada en vez de la versión actual? como se puede visualizar en la Imagen 6 la versión 14.17.5 LTS termina con las siglas LTS, esto significa que la versión tiene soporte a largo plazo, garantiza que los errores críticos se solucionaran en un lapso corto, lo que no sucede con la versión actual que no cuenta con el soporte a largo plazo.

Después de descargar Node JS, se debe realizar la instalación. La instalación no requiere conocimientos técnicos, es sencilla y basta con dar clic en siguiente varias veces. Se sugiere no cambiar las opciones por defecto de la instalación. La Imagen 7 presenta el progreso de la instalación de Node JS.

Imagen 7. Progreso de instalación de Node JS



Fuente: (OpenJS Foundation, 2021)

Advertencia

En versiones actualizadas de Windows no es necesario reiniciar el equipo. Sin embargo, no está demás reiniciarlo después de instalar NODE JS.

Para verificar la instalación de Node JS y su gestor de paquetes npm (Node Package Manager) se debe abrir una consola de comando de Windows (CMD) como se aprecia en la Imagen 8 para ejecutar los siguientes comandos:

```
node -v
```

```
npm -v
```

Imagen 8. Verificación de la instalación Node JS y npm

Fuente: (Autor, 2021)

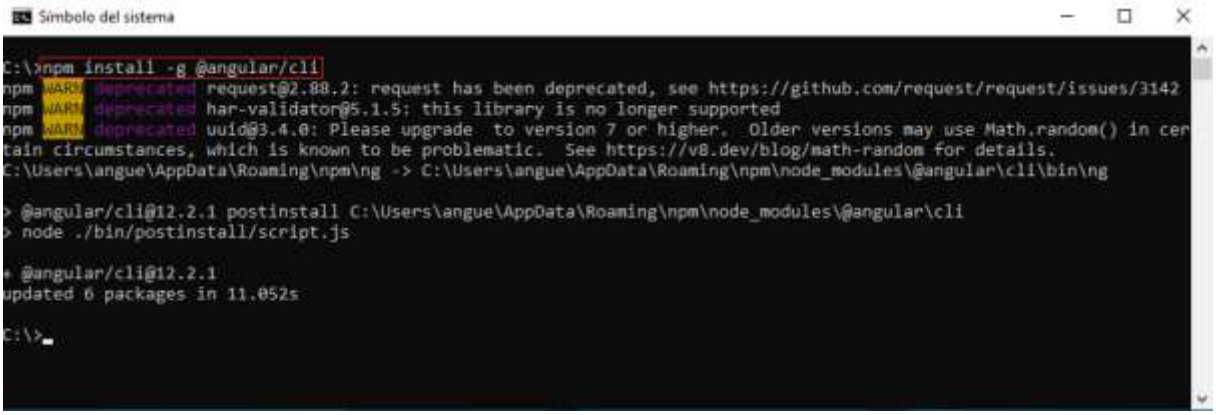
7.1.4.3 Instalación de Angular

Para instalar Angular es necesario tener instalado NodeJS y por ende el gestor de paquetes npm. Angular se puede instalar de forma global, esto quiere se podrá utilizar desde cualquier carpeta del disco e instalará la última versión disponible. EL comando para instalar Angular es:

```
npm install -g @angular/cli
```

La Imagen 9 presenta el resultado de la instalación global de angular en el equipo.

Imagen 9. Utilizando npm para instalar Angular



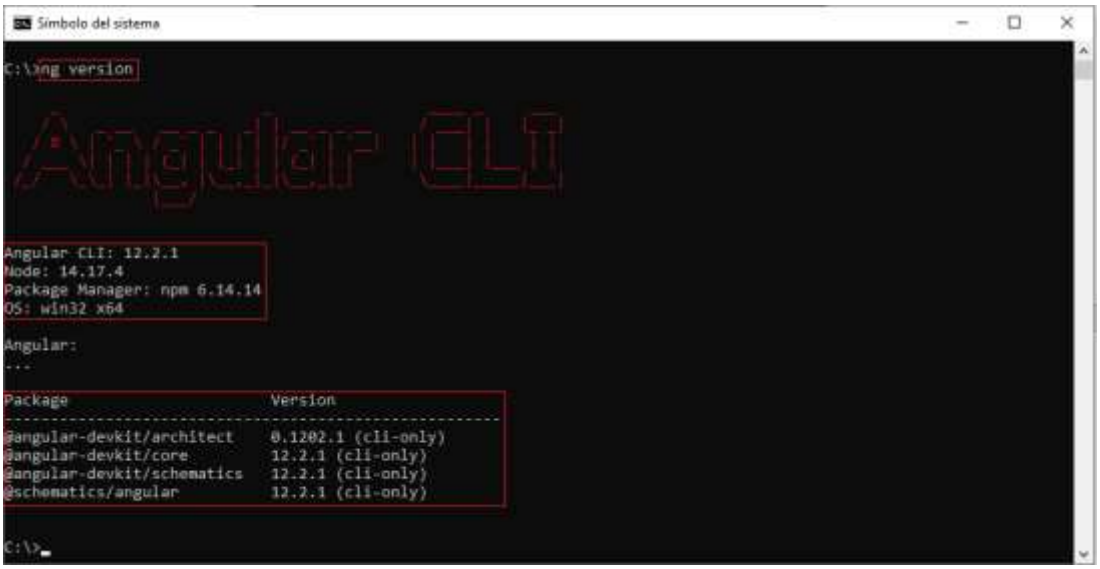
Fuente: (Autor, 2021)

Para verificar la versión de Angular instalada se debe ejecutar el siguiente comando en una consola de Windows:

```
ng version
```

La Imagen 10 presenta el resultado del comando ng version.

Imagen 10. Versión instalada de angular, nodeJS y npm



Fuente: (Autor, 2021)

Idea clave

Si ya tenía instalada una versión de Angular y se desea conservar la versión de Node JS, en posible desinstalar solamente angular de la siguiente manera:

```
npm uninstall -g @angular/cli
npm cache clean
npm cache verify
npm cache clean --force
```

7.1.4.4 Instalación de TypeScript

Este lenguaje para apoyar el desarrollo sobre JavaScript se puede utilizar de dos formas, por un lado, realizando la instalación de manera global, así cualquier proyecto tanto en angular como en Node JS lo podrá utilizar. Por otro lado, instalándolo como un paquete adicional en un proyecto particular. En esta guía vamos a instalar typescript de manera global a través de la ejecución del siguiente comando.

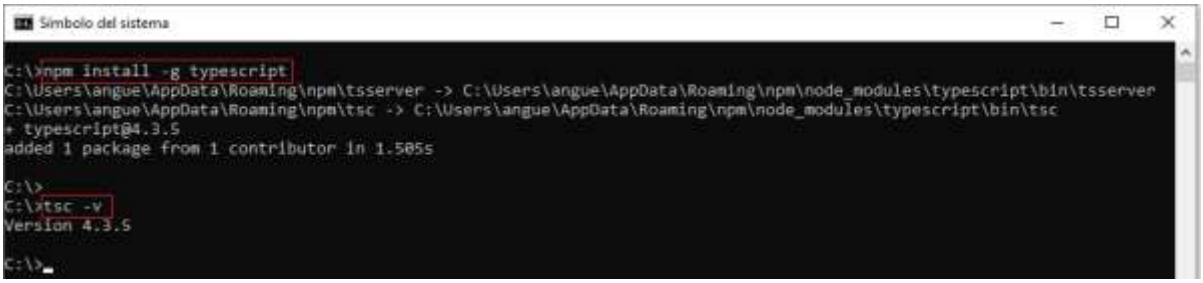
```
npm install -g typescript
```

Para verificar la instalación de typescript se utiliza el siguiente comando:

```
tsc -v
```

La Imagen 11 presenta el resultado de la instalación global de typescript y la versión instalada

Imagen 11. Instalación de typescript y su versión



Fuente: (Autor, 2021)

7.1.4.5 Editores de código

Infortunadamente las mejores IDE's para desarrollo como Eclipse, Netbeans y Visual Studio no tienen componentes optimizados para desarrollar en Angular y Node JS, por lo tanto, optaremos por utilizar dos de los tres editores de código más populares para Angular y Node JS. La Imagen 12 presenta los sitios Web de cada los mejores editores de código.

Imagen 12. Editores de código más populares para Angular

https://www.sublimetext.com https://atom.io/ https://code.visualstudio.com

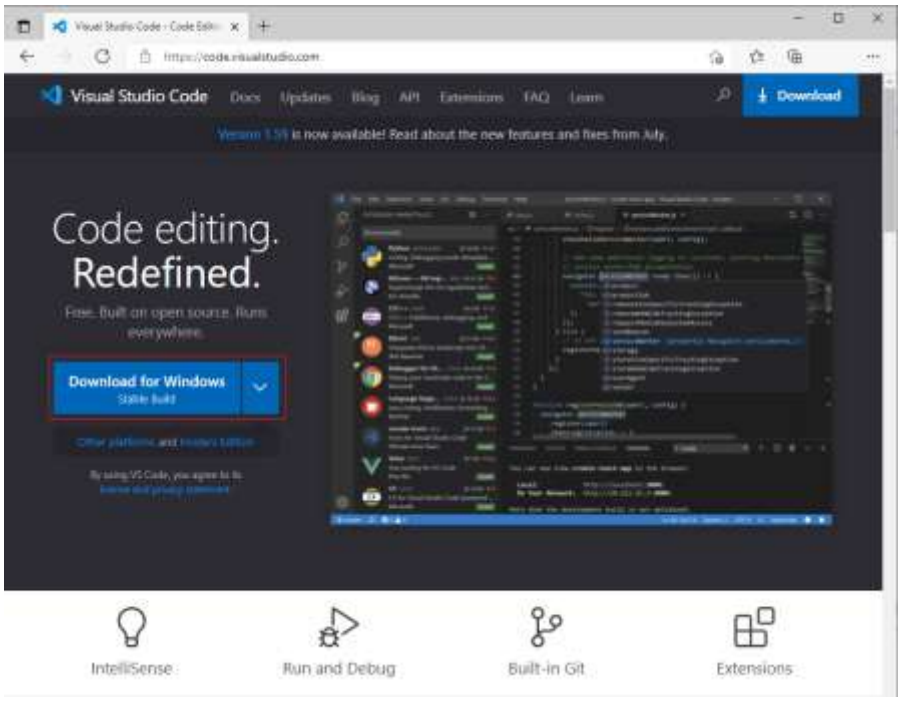
Fuente: (Autor, 2021)

En estas guías se utilizará Visual Studio Code para desarrollar el frontend y Atom para desarrollar el backend.

7.1.4.6 Instalación de Visual Studio Code

La descarga del editor seleccionado se realiza del sitio web https://code.visualstudio.com en la opción "***Download***" en la parte superior derecha. En la Imagen 13 se puede apreciar el sitio web del editor de código seleccionado para el frontend.

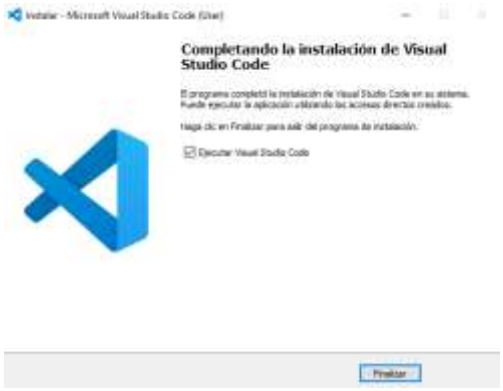
Imagen 13. Sitio web de Visual Studio Code



Fuente: (Microsoft, 2021)

El proceso de instalación es simple, como todo instalador de Windows con varios clics en siguiente, siguiente y finalizar es suficiente. La Imagen 14 presenta la instalación exitosa del editor de código.

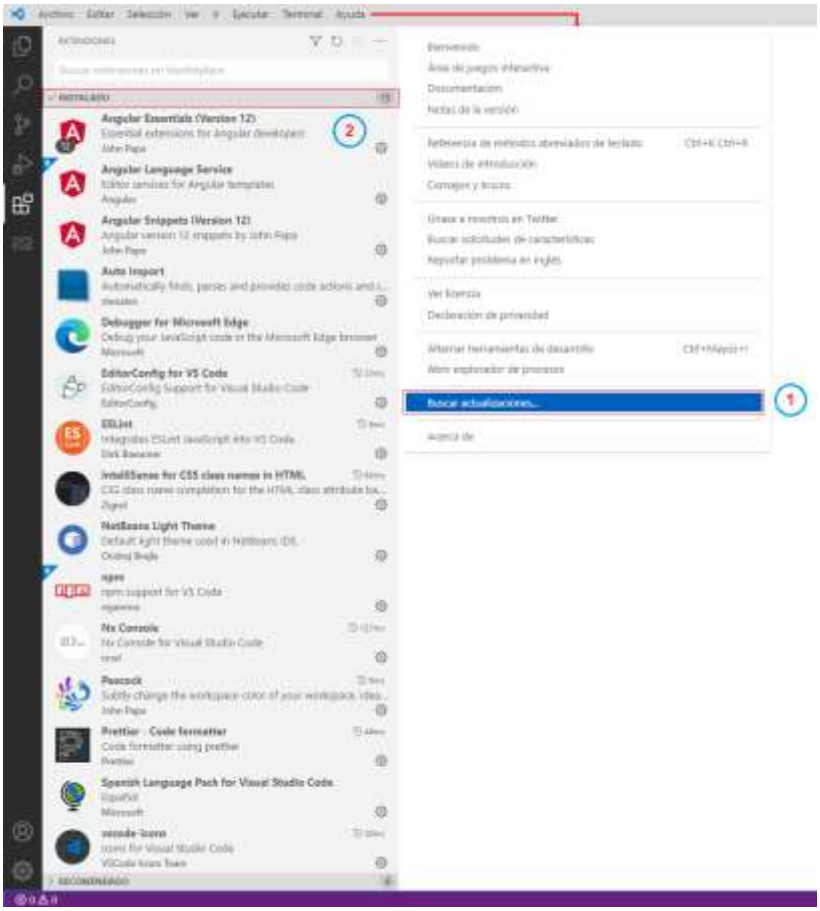
Imagen 14. Instalación correcta de Visual Studio Code



Fuente: (Microsoft, 2021)

Si ya tiene instalado el editor Visual Studio Code, no es necesario desinstalarlo, es suficiente con hacer clic en el menú ayuda (Help) y seleccionar la opción buscar actualizaciones (Check for Updates). La Imagen 15 presenta la forma de realizar la actualización en Visual Studio Code y las extensiones más útiles al momento de desarrollar en Angular.

Imagen 15. Actualización y extensiones sugeridas para el Visual Studio Code

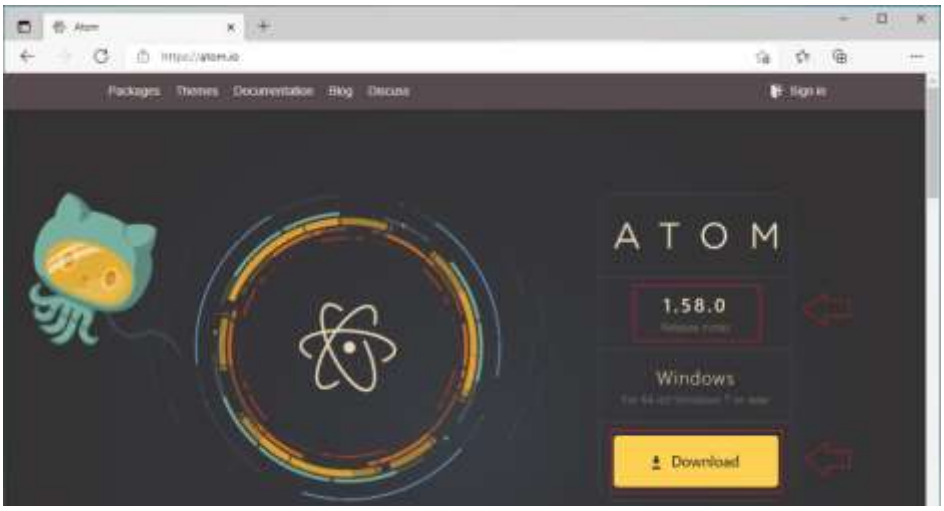


Fuente: (Microsoft, 2021)

7.1.4.7 Instalación del editor de código Atom

El sitio Web del editor de código Atom es https://atom.io la opción de descarga y la versión se presenta en la Imagen 16.

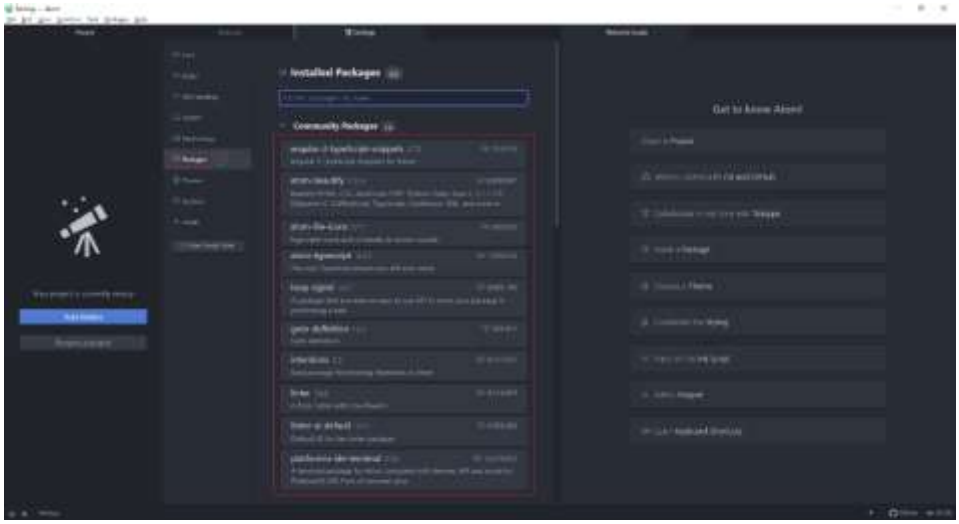
Imagen 16. Sitio oficial y descarga de Atom



Fuente: (@atom, 2021)

El proceso de instalación del editor es sencillo, clic en el instalador y seguir las instrucciones con las opciones por defecto. La Imagen 17 presenta la interfaz de Atom y los principales paquetes adicionales que debe tener el editor de código para desarrollar cómodamente el backend.

Imagen 17. Editor de código Atom

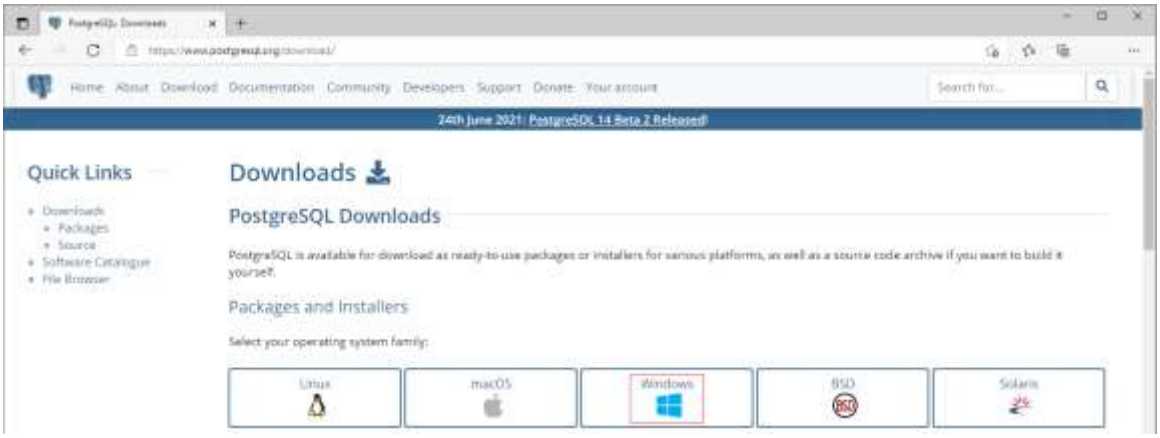


Fuente: (GitHub, 2021)

7.1.4.8 Instalación de PostgreSQL

PostgreSQL es el motor de base de datos relacional de código abierto más potente en la actualidad. Se puede descargar del sitio Web https://www.postgresql.org seleccionando el sistema operativo sobre el cual se desee trabajar y asegurándose de descargar los instaladores binarios. La Imagen 18 presenta el sitio Web para descargar PostgreSQL.

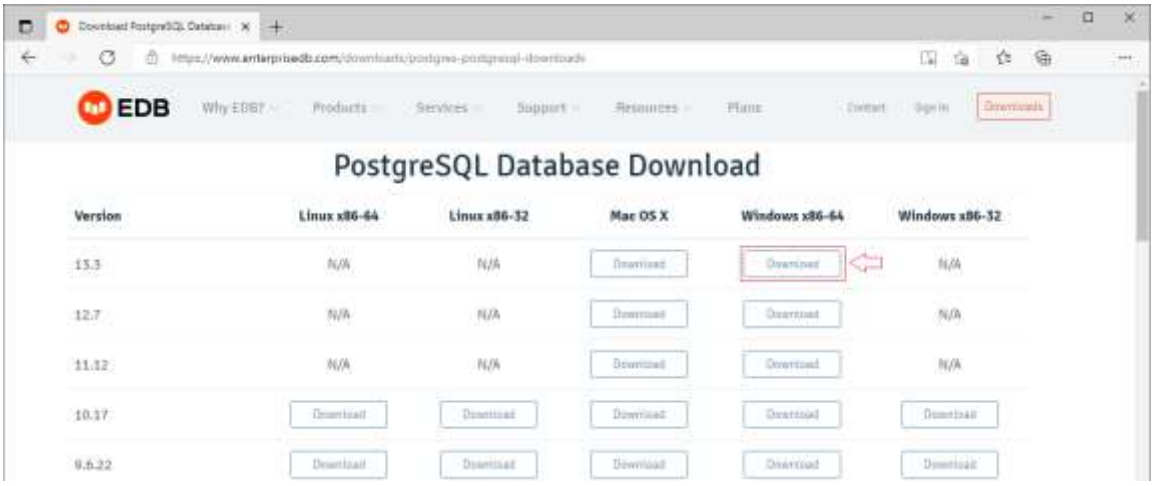
Imagen 18. Sistemas operativos disponibles para descargar PostgreSQL



Fuente: (PostgreSQL, 2021)

El número de la versión y el sistema operativo depende del desarrollador, en esta guía se utilizará la versión 13.3 para Windows de 64 bits. La Imagen 19 presenta la descarga seleccionada en esta guía.

Imagen 19. Versión de PostgreSQL a utilizar

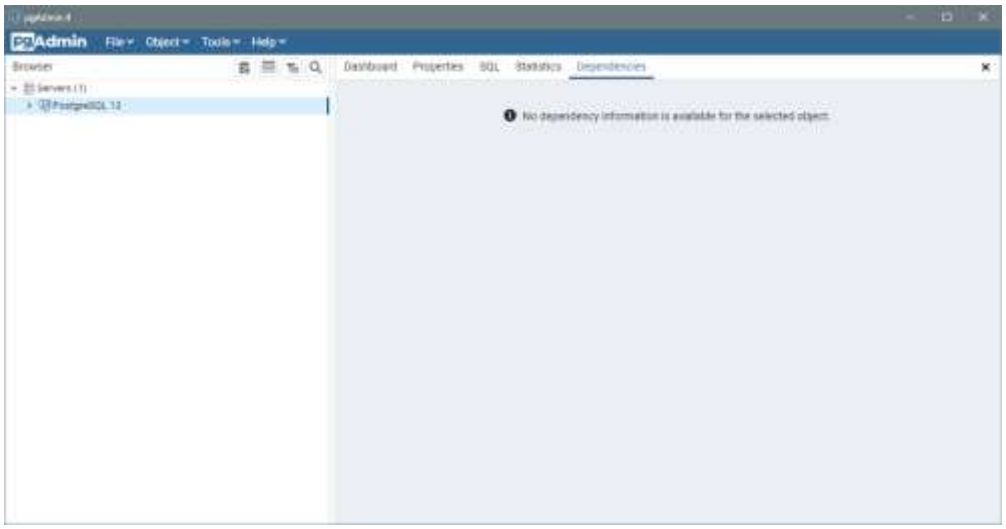


Fuente: (PostgreSQL, 2021)

La instalación de PostgreSQL se puede realizar con todos los valores por defecto, no es necesario cambiar nada. Es importante que el desarrollador recuerde la contraseña que asignará a la base de datos.

La herramienta gráfica para la gestión de bases de datos a utilizar es pgAdmin. Se puede encontrar en el menú de Windows una vez finaliza la instalación. Al ejecutar pgAdmin la herramienta solicitará la contraseña asignada en el proceso de instalación. La Imagen 20 presenta la interfaz gráfica que se utilizará para la gestión de las bases de datos.

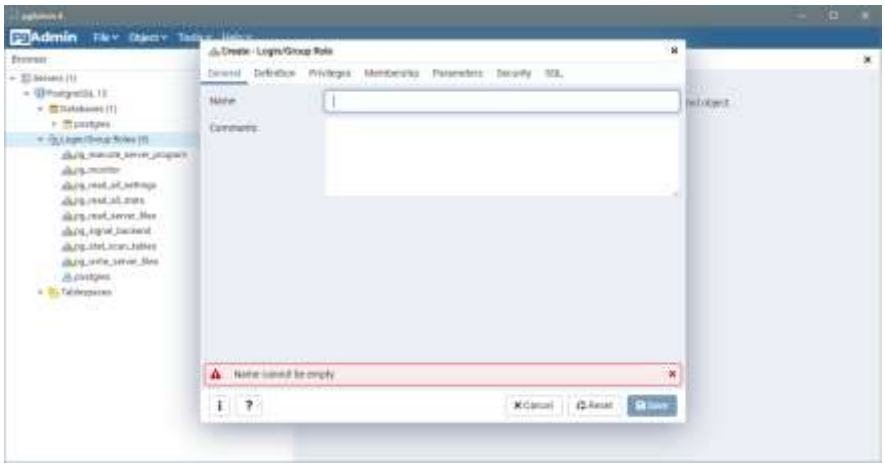
Imagen 20. Herramienta para la administración – pgAdmin



Fuente: (Autor, 2021)

Es de vital importancia la creación de un usuario que permita gestionar las bases de datos que se crearán. La Imagen 21 presenta el árbol de opciones que agrupa contiene la opción Login/group Roles. En la opción Login/group Roles se debe hacer clic derecho para crear un nuevo usuario. La Imagen 21 presenta la interfaz para la creación del nuevo usuario.

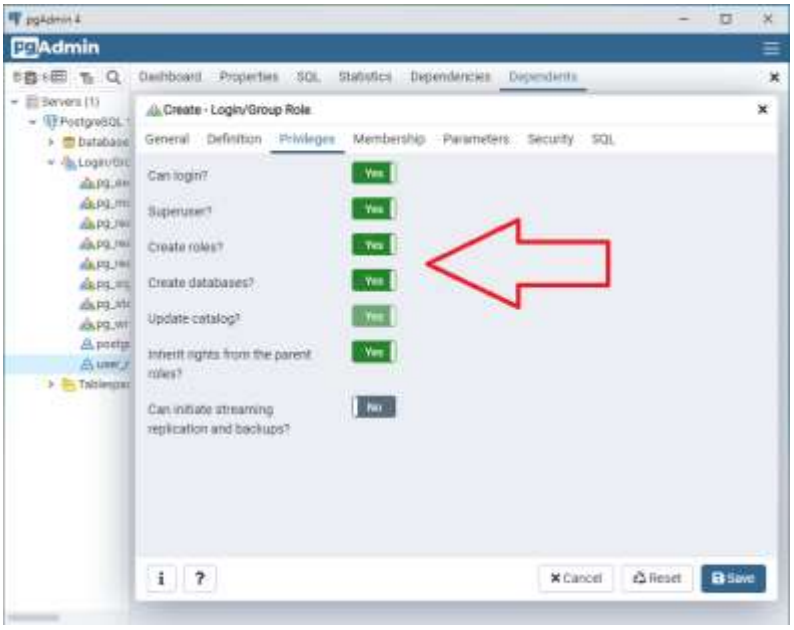
Imagen 21. Ventana crear usuario en el pgAdmin



Fuente: (Autor, 2021)

Las pestañas: general, definición (para asignar la contraseña) y privilegios deben ser diligenciadas de manera completa, de lo contrario el usuario no estará en capacidad de permitir el acceso a las bases de datos que se le asignen. La Imagen 22 presenta los privilegios que se deben asignar para completar la creación del usuario.

Imagen 22. Ventana para la creación de usuario - Privilegios

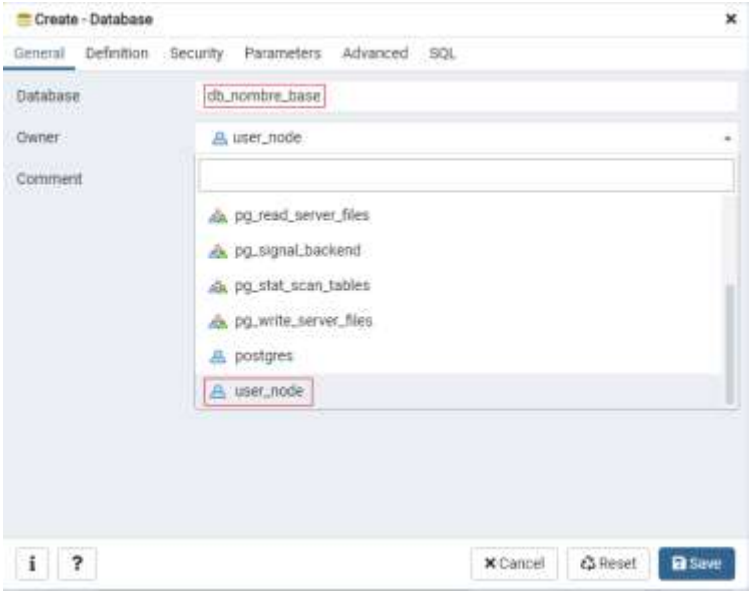


Fuente: (Autor, 2021)

Para finalizar, la creación de una base de datos permitirá verificar todo el proceso de instalación de PostgreSQL y de la creación del usuario para permitir la conexión posterior desde Node JS. En el árbol de la izquierda en la opción bases de datos (Databases) se

debe hacer clic derecho y seleccionar la opción nueva base de datos. La Imagen 23 presenta la ventana modal para la creación de una base de datos.

Imagen 23. Ventana para la creación de usuario - Privilegios



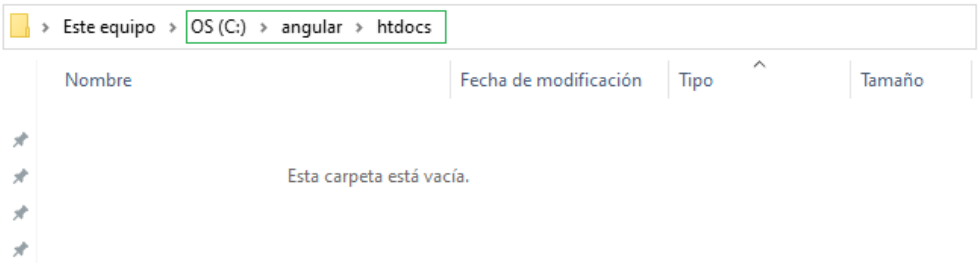
Fuente: (Autor, 2021)

7.1.5. Primer proyecto Angular y prueba del editor de código

7.1.5.1 Estructura de carpetas sugerida

La organización de las carpetas para administrar los proyectos de Angular queda a criterio del desarrollador, sin embargo, a continuación, se propone una estructura de carpetas que permita organizar los proyectos de forma similar a como lo hacen otros lenguajes de programación Web. La Imagen 24 presenta la estructura de carpetas sugerida. La creación de estas carpetas se realiza de manera manual en el explorador de Windows.

Imagen 24. Estructura de carpetas sugerida

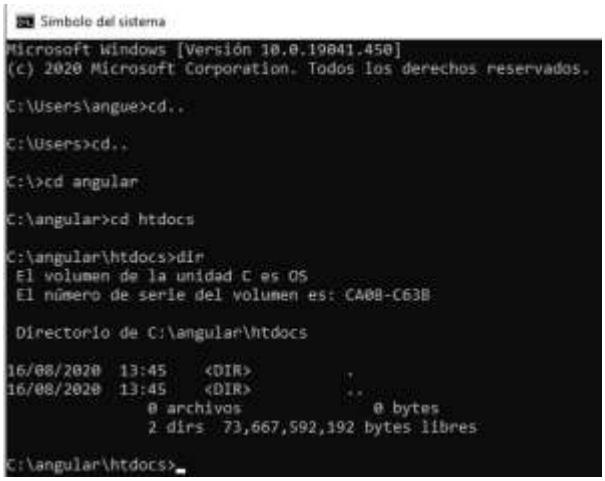


Fuente: (Autor, 2021)

7.1.5.2 Creación de un proyecto en Angular

Para crear el primer proyecto en Angular se debe abrir una consola de Windows y cambiar de directorio hasta la estructura sugerida en el punto anterior. La Imagen 25 presenta los pasos para cambiar de carpeta.

Imagen 25. Consola carpeta sugerida



Fuente: (Autor, 2021)

Los comandos utilizados en la consola de Windows fueron:

```
cd ..
cd ..
cd angular
cd htdocs
```

Una vez ubicada la carpeta sugerida para almacenar los proyectos de Angular, se procede a crear el primer proyecto, ejecutando el siguiente comando:

```
C:\angular\htdocs> ng new taller01
```

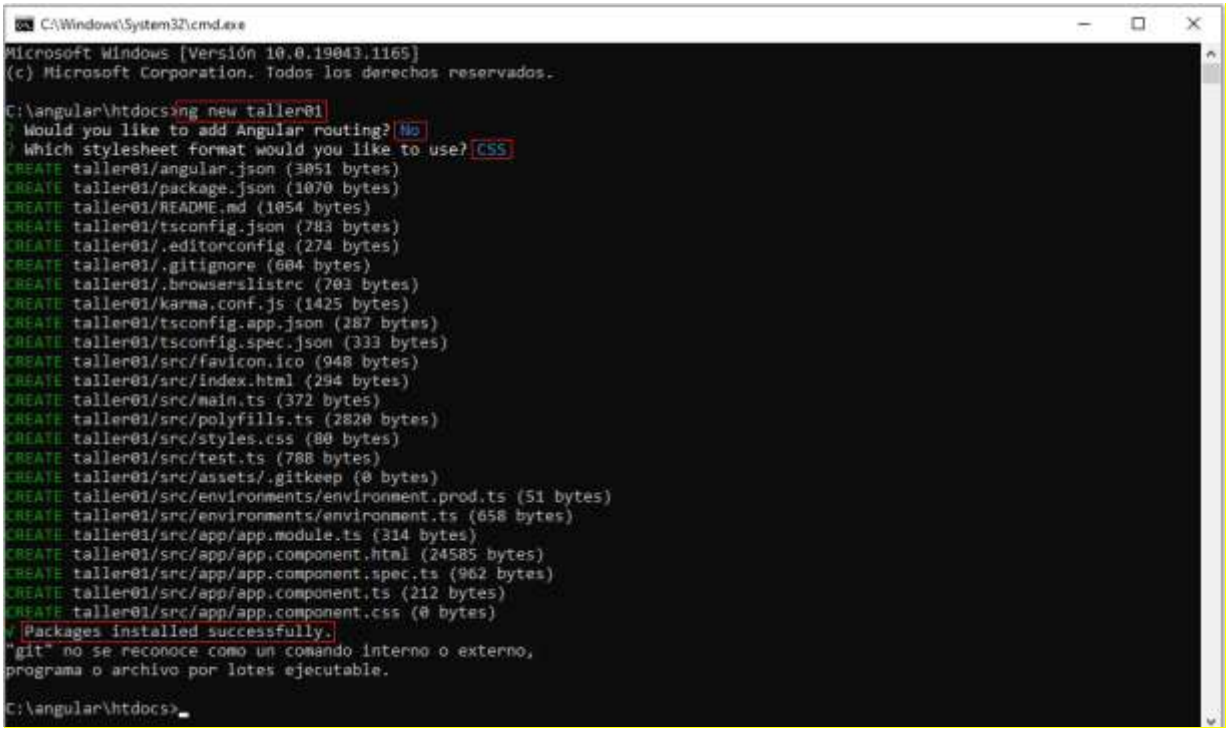
La Imagen 26 presenta la ejecución del comando para crear un proyecto en Angular. Al ejecutar el comando ANGULAR CLI le hará dos preguntas:

1) ¿Desea utilizar rutas?, por ser un primer ejercicio se sugiere **N**

2) ¿Qué tipo de hoja de estilo se desea utilizar?, se sugiere **CSS**,

con [Enter] iniciará el proceso de creación del proyecto. El proceso consiste en la creación de una estructura básica de proyecto la cual es descargada de internet.

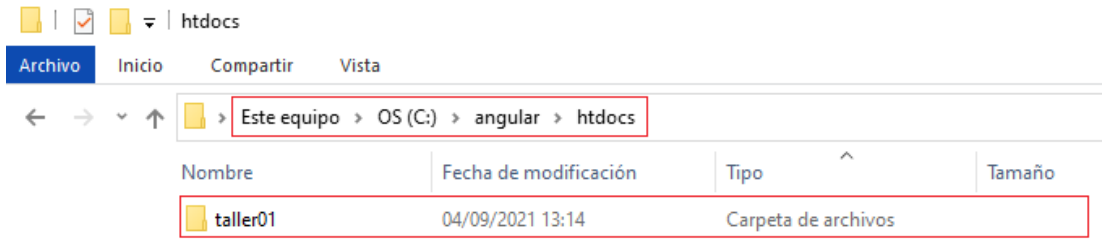
Imagen 26. Creación del taller01 con Angular CLI



Fuente: (Autor, 2021)

Dependiendo de la configuración del equipo, se pueden obtener tres tipos de mensaje. En este caso la creación del proyecto fue exitosa y se generó una advertencia puesto que no se está utilizando GIT. En el explorador de archivos de Windows se puede observar la carpeta que se generó al crear el proyecto como se presenta en la Imagen 27.

Imagen 27. Carpeta generada por Angular CLI



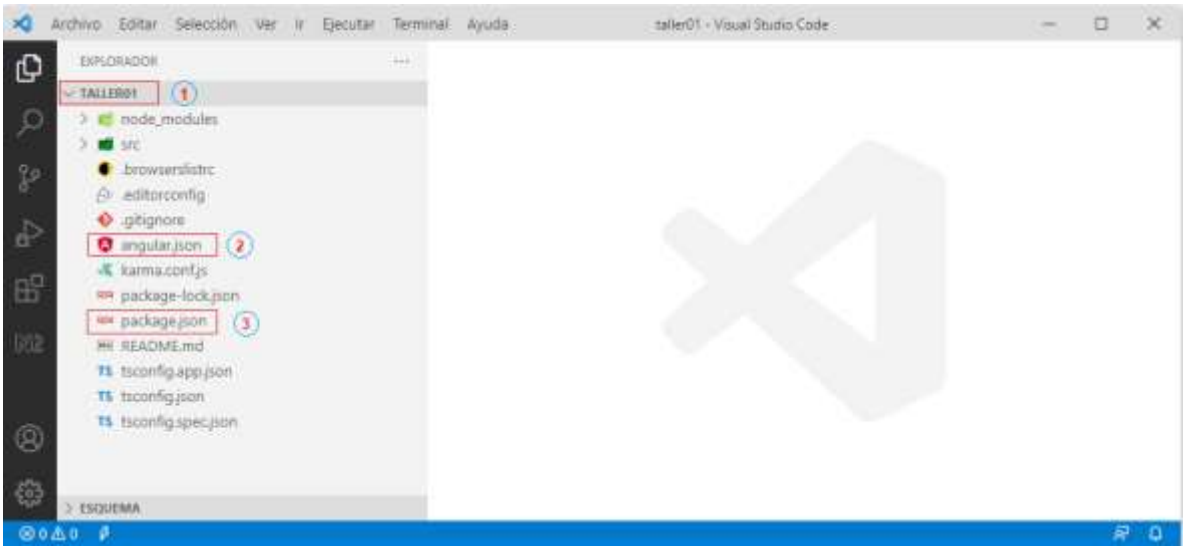
Fuente: (Autor, 2021)

7.1.5.3 Prueba del editor de código

El editor de código seleccionado no maneja el concepto de proyecto, en su defecto utiliza el concepto de carpeta. Esto quiere decir que se pueden ejecutar varias instancias de Visual Studio con diferentes carpetas abiertas. Con Visual Studio Code abierto, se debe

seleccionar el menú “archivo” o “File” y luego la opción “abrir carpeta” o “Open Folder”, se debe buscar la carpeta creada en el paso anterior. La Imagen 28 presenta el proyecto cargado.

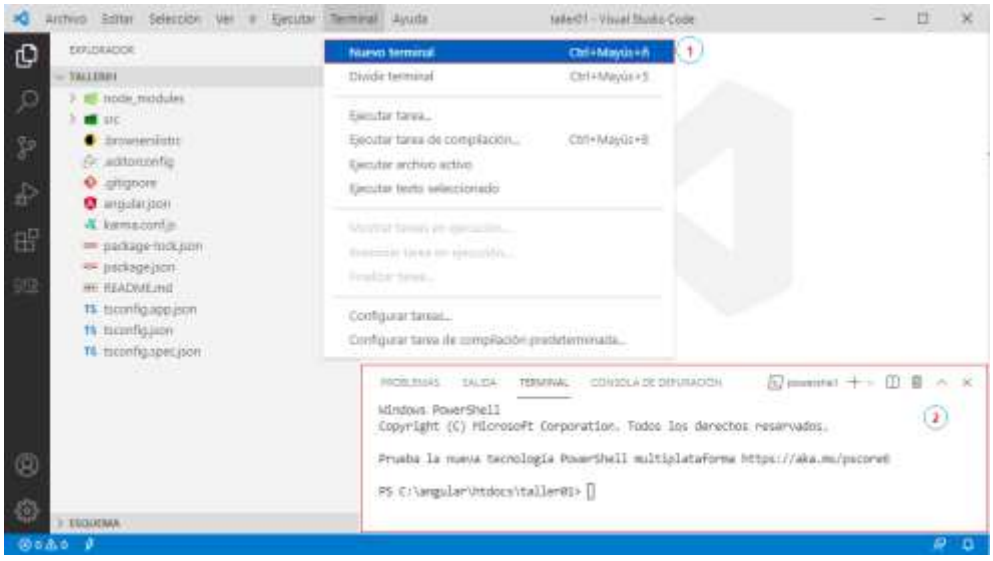
Imagen 28. Inclusión de la carpeta del taller001 en Visual Studio Code



Fuente: (Autor, 2021)

El siguiente paso es ejecutar el proyecto desde el editor, con esto se busca probar el editor de código y ver el resultado de correr el primer proyecto creado con Angular CLI. Para ejecutar el proyecto se procede abrir una terminal desde el editor. En el menú superior se selecciona la opción “Terminal” y luego “New Terminal”. La Imagen 29 presenta el editor con la consola o terminal lista para recibir comandos.

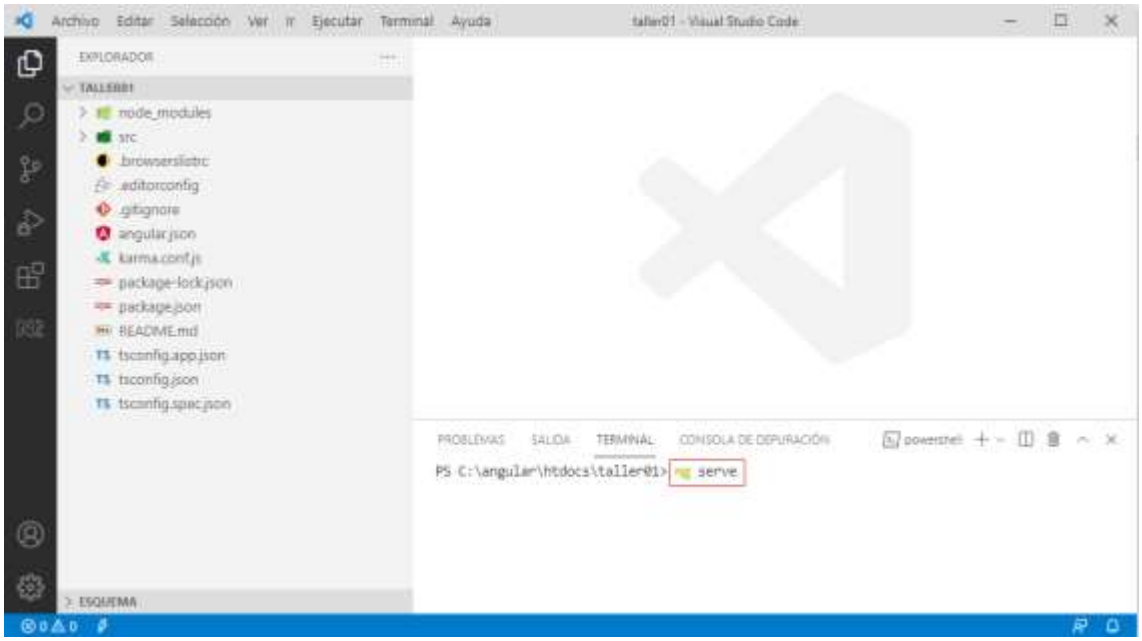
Imagen 29. Abriendo terminal de Visual Studio Code



Fuente: (Autor, 2021)

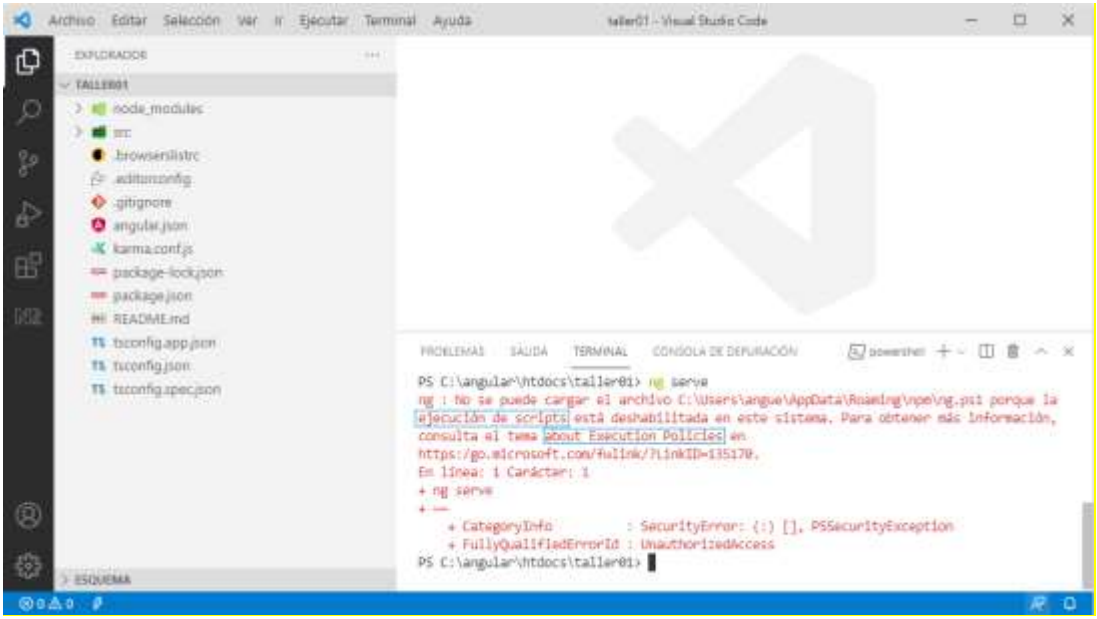
Para correr el proyecto creado con Angular CLI, se debe ejecutar desde la terminal de Visual Estudio Code, el comando "***ng serve***", la Imagen 30 presenta la distribución normal del editor de código seleccionado.

Imagen 30. Inicio del servidor y ejecución del taller01



Al intentar ejecutar el proyecto desde el editor de código, puede que se presente un error. El error se da porque Windows no permite la ejecución de Scripts, esto se debe a las políticas de seguridad de Windows. Dado que los scripts que se van a utilizar serán creados por el desarrollador, es factible eliminar las restricciones para trabajar sin problemas con Visual Studio Code. La Imagen 31 presenta el error que se puede generar.

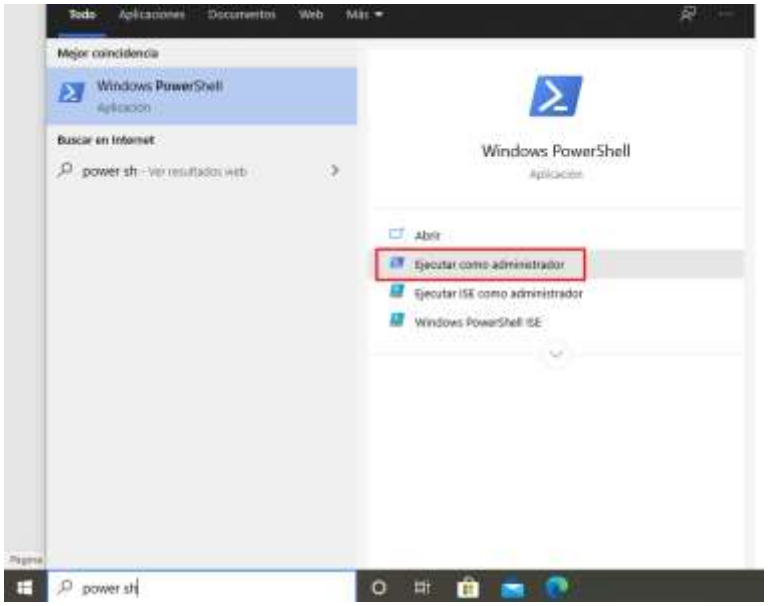
Imagen 31. Error políticas de la PowerShell en la terminal de Visual Studio Code



Fuente: (Autor, 2021)

Para eliminar el error en la ejecución de Scripts se debe abrir una consola de PowerShell en modo Administrador. En la Barra de Windows se debe escribir “PowerShell” y aparecerá una opción como la presentada en la Imagen 32.

Imagen 32. Ejecución como administrador la PowerShell



Fuente: (Autor, 2021)

En la consola “**Azul**” de PowerShell se deben ejecutar los comandos para verificar las políticas en cuanto a ejecución de Scripts:

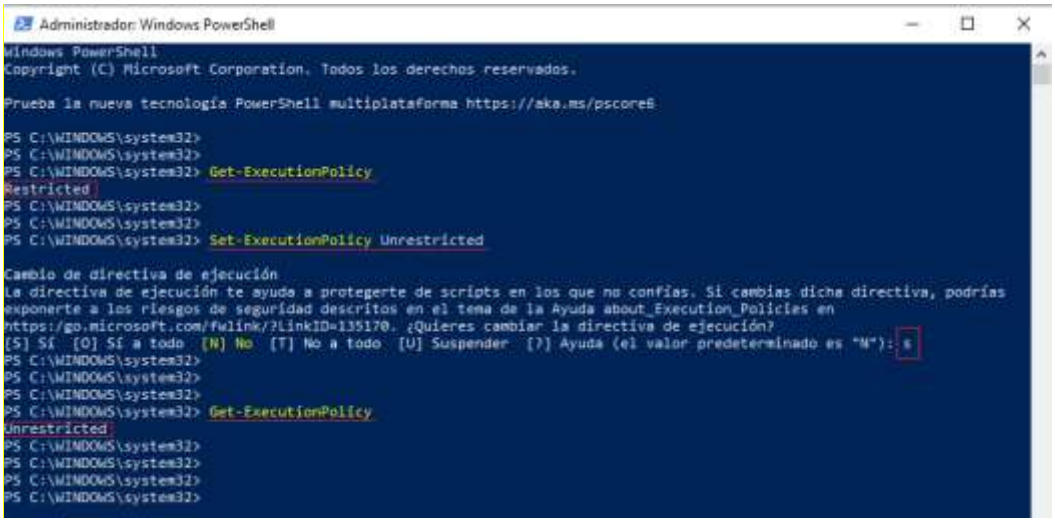
`Get-ExecutionPolicy`

El comando anterior se ejecuta para consultar el estado de las políticas de ejecución de Scripts.

`Set-ExecutionPolicy Unrestricted`

El comando anterior se ejecuta para eliminar las restricciones de ejecución de Scripts. La Imagen 33 presenta la ejecución de los comandos anteriores.

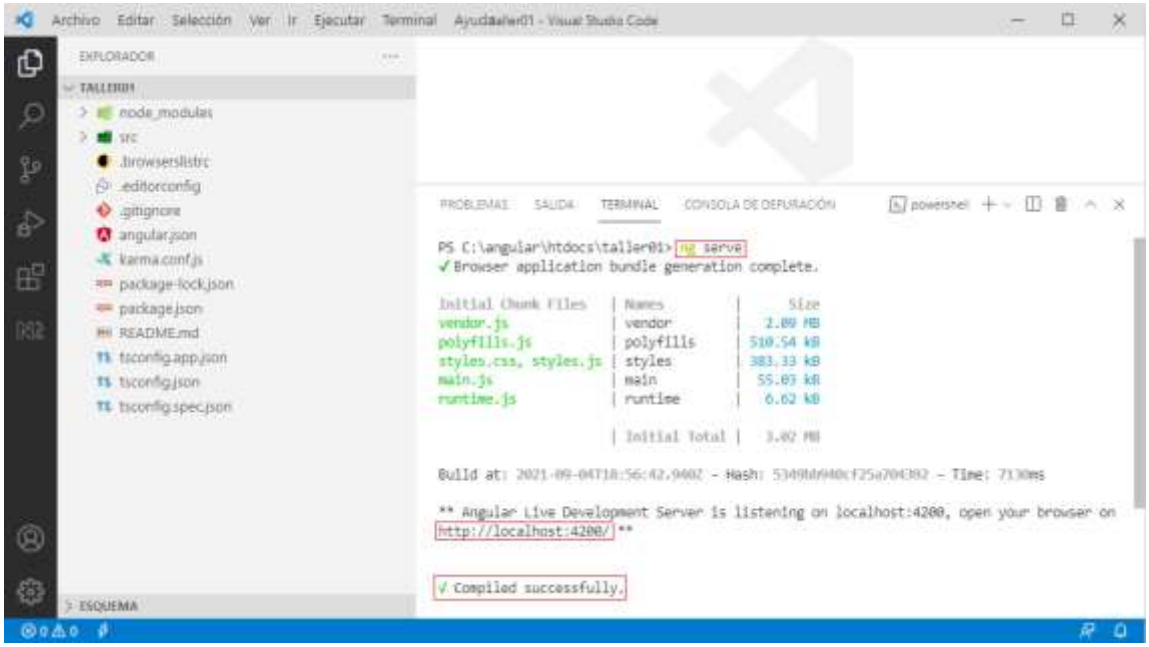
Imagen 33. Terminal de PowerShell para asignar permisos de ejecución de Scripts



Fuente: (Autor, 2021)

Una vez eliminadas las restricciones para la ejecución de Scripts se procede a ejecutar nuevamente el comando "***ng serve***" en la terminal del Visual Studio Code. Al ejecutar el comando, Angular preguntará si se desean compartir datos de forma anónima con Google. Si la data que se está trabajando en el proyecto no es privada, se recomienda compartir los datos. El comando "***ng serve***" realizará un proceso de transpilación del código generado en el proyecto. La Imagen 34 presenta el resultado al ejecutar el servidor y habilitar el proyecto.

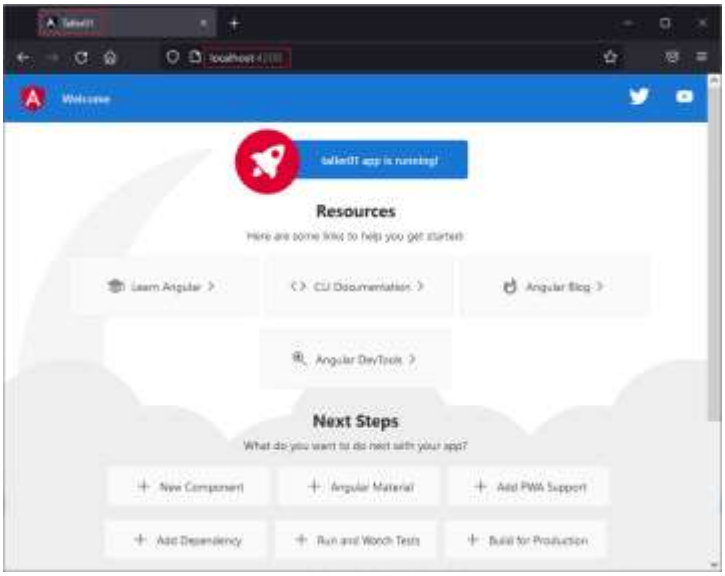
Imagen 34. Ejecutando el proyecto en el editor Visual Studio Code



Fuente: (Autor, 2021)

El resultado de la compilación no arroja errores y permite ver que el proyecto puede ser verificado en un navegador Web en la URL http://localhost:4200. La Imagen 35 presenta el proyecto ejecutándose correctamente el navegador Firefox.

Imagen 35. Taller001 ejecutándose en el navegador



Fuente: (Autor, 2021)

7.2. Guía 2

7.2.1. Taller02, conociendo la estructura

7.2.1.1 Ubicación del proyecto

Como se presenta en la Imagen 36, los talleres se crearán en la carpeta: "***c:\angular\htdocs***",

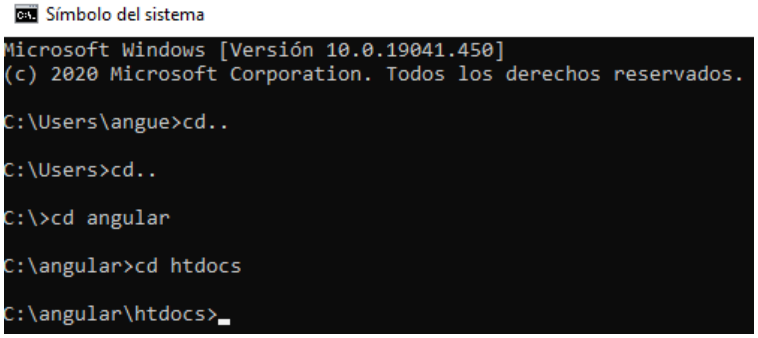
Imagen 36. Ruta para almacenar el taller02

Fuente: (Autor, 2021)

7.2.1.2 Creación del proyecto

Para crear el taller se debe abrir una consola comandos de Windows (CMD), luego se cambia la carpeta, asegurándose que la ruta final sea "c:\angular\htdocs". Imagen 37 presenta la consola de comandos con los pasos.

Imagen 37. Consola de comando para crear el taller02



Fuente: (Autor, 2021)

Idea clave

En estos primeros ejercicios no es necesario habilitar rutas en la creación del proyecto, además, es recomendable escoger el estilo *CSS* para al momento de crear el proyecto

Una vez en la carpeta "c:\angular\htdocs" se debe ejecutar el comando:

```
C:\angular\htdocs> ng new taller02
```

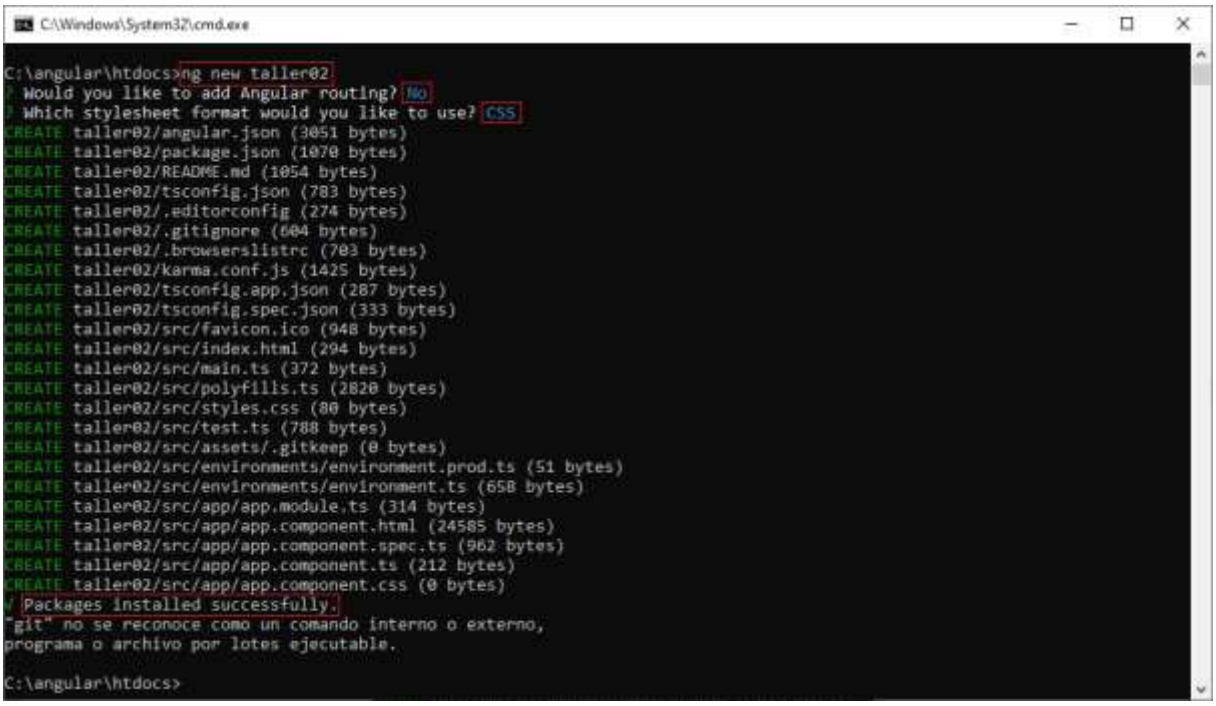
El comando anterior creará la estructura básica de los archivos, conocida como "***Scaffolding***". La Imagen 38 presenta la ejecución del comando anterior. Al ejecutar el comando "***ng new taller02***" aparecerán dos preguntas:

1) ¿Desea utilizar rutas?, por ser un primer ejercicio se sugiere N

2) ¿Qué tipo de hoja de estilo se desea utilizar?, se sugiere CSS

con [Enter] iniciará el proceso de creación del proyecto. El proceso consiste en la creación de una estructura básica de proyecto la cual es descargada de internet.

Imagen 38. Resultado al crear el proyecto taller002



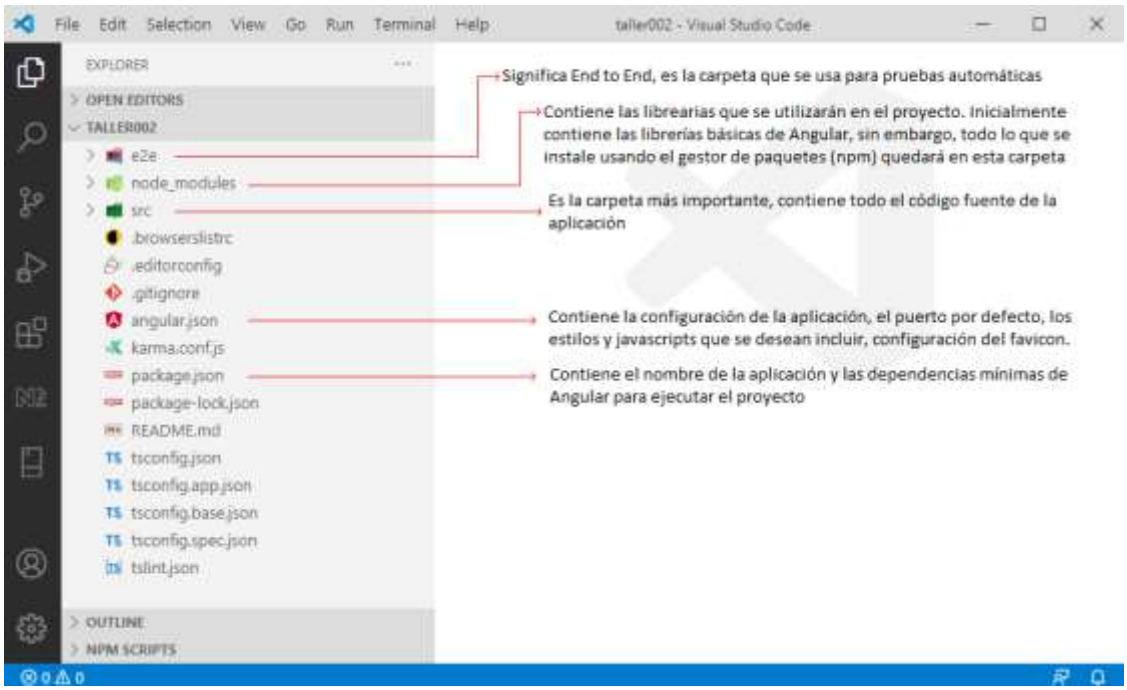
Fuente: (Autor, 2021)

7.2.1.3 Estructura y descripción de las principales carpetas y archivos de Angular

Una vez Angular CLI genera el proyecto, se procede a cargar el proyecto en Visual Studio Code, en el menú "***FILE***" o "***Archivo***", opción "***Open Folder***" o "***Abrir carpeta***".

Si había otro proyecto abierto en el editor de código, sencillamente Visual Studio Code abrirá la nueva carpeta seleccionada y cerrará la anterior. La Imagen 39 describe las principales carpetas y archivos de un proyecto en Angular.

Imagen 39. Principales archivos y carpetas de un proyecto Angular



Fuente: (Autor, 2021)

Toca aplicación en Angular tiene un componente por defecto. Este componente se encuentra en la carpeta "***src\app***" y contiene 5 archivos. La Imagen 40 presenta los archivos que tienen un el componente por defecto.

Imagen 40. Principales archivos de la carpeta app

Fuente: (Autor, 2021)

7.2.1.4 Cambiando el contenido de la plantilla y la hoja de estilos

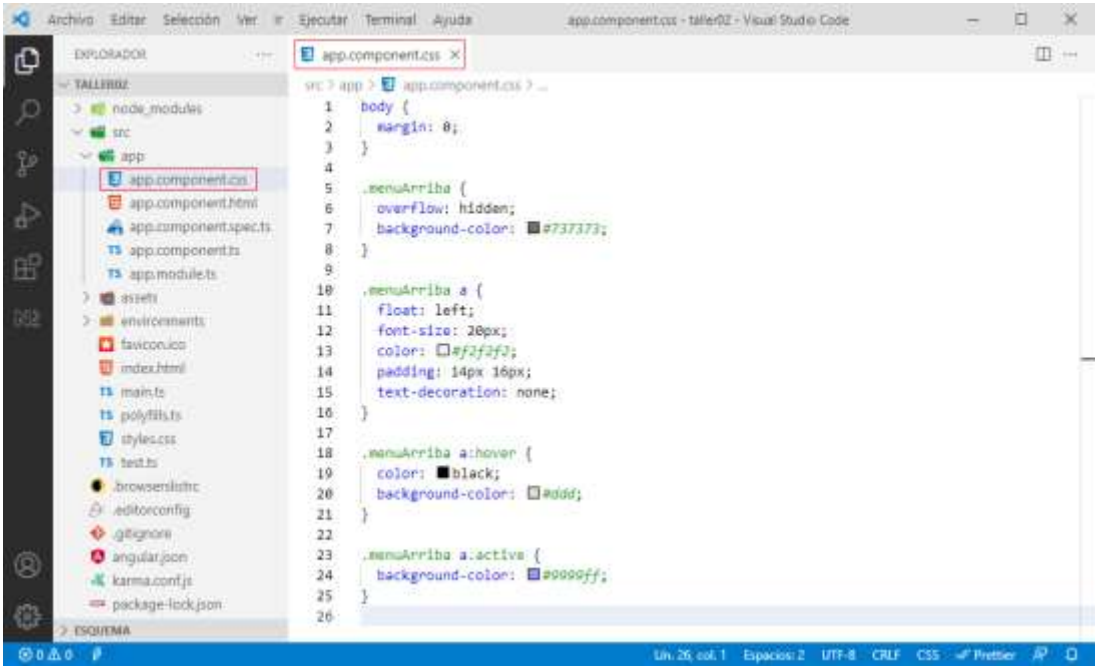
En el archivo "***src\app\app.component.css***", se colocarán los estilos propuestos en la Tabla 3, la nueva información solo tendrá efecto si el archivo está grabado.

```
body {
  margin: 0;
}

.menuArriba {
  overflow: hidden;
  background-color: #737373;
}

.menuArriba a {
  float: left;
  font-size: 20px;
  color: #f2f2f2;
  padding: 14px 16px;
  text-decoration: none;
}

.menuArriba a:hover {
  color: black;
  background-color: #ddd;
}

.menuArriba a.active {
  background-color: #9999ff;
}
```

Tabla 4. Taller02, ejemplo para el archivo src\app\app.component.css

La Imagen 41 presenta los estilos propuestos en el archivo "***src\app\app.component.css***" utilizando Visual Studio Code.

Imagen 41. Taller02, estilos sugeridos para el componente principal



Fuente: (Autor, 2021)

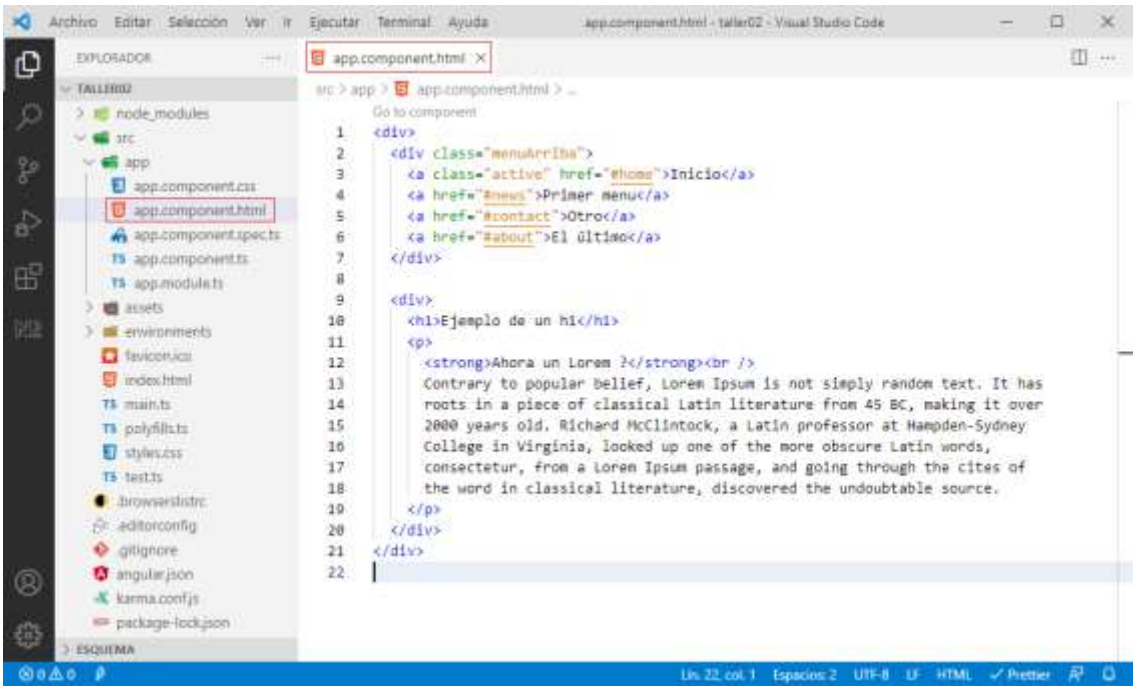
En el archivo "***src\app\app.component.html***" se encuentra la ***plantilla*** del componente por defecto. Este archivo contiene el código HMTL que Angular generó por defecto al crear el proyecto. A manera de ejercicio se sugiere sobrescribir el contenido de la plantilla con el código de la Tabla 5.

```
<div>
  <div class="menuArriba">
    <a class="active" href="#home">Inicio</a>
    <a href="#news">Primer menu</a>
    <a href="#contact">Otro</a>
    <a href="#about">El último</a>
  </div>

  <div>
    <h1>Ejemplo de un h1</h1>
    <p><strong>Ahora un Lorem ?</strong><br />
      Contrary to popular belief, Lorem Ipsum is not simply random text.
      It has roots in a piece of classical Latin literature from 45 BC, making it over 2000 years old.
      Richard McClintock, a Latin professor at Hampden-Sydney College in Virginia,
      looked up one of the more obscure Latin words, consectetur, from a Lorem Ipsum passage,
      and going through the cites of the word in classical literature, discovered the undoubtable source.</p>
  </div>
</div>
```

Tabla 5. Taller02, ejemplo para el archivo src\app\ app.component.html

La Imagen 42 presenta el código HTML propuesto en el archivo "***src\app\app.component. html***" utilizando Visual Studio Code.

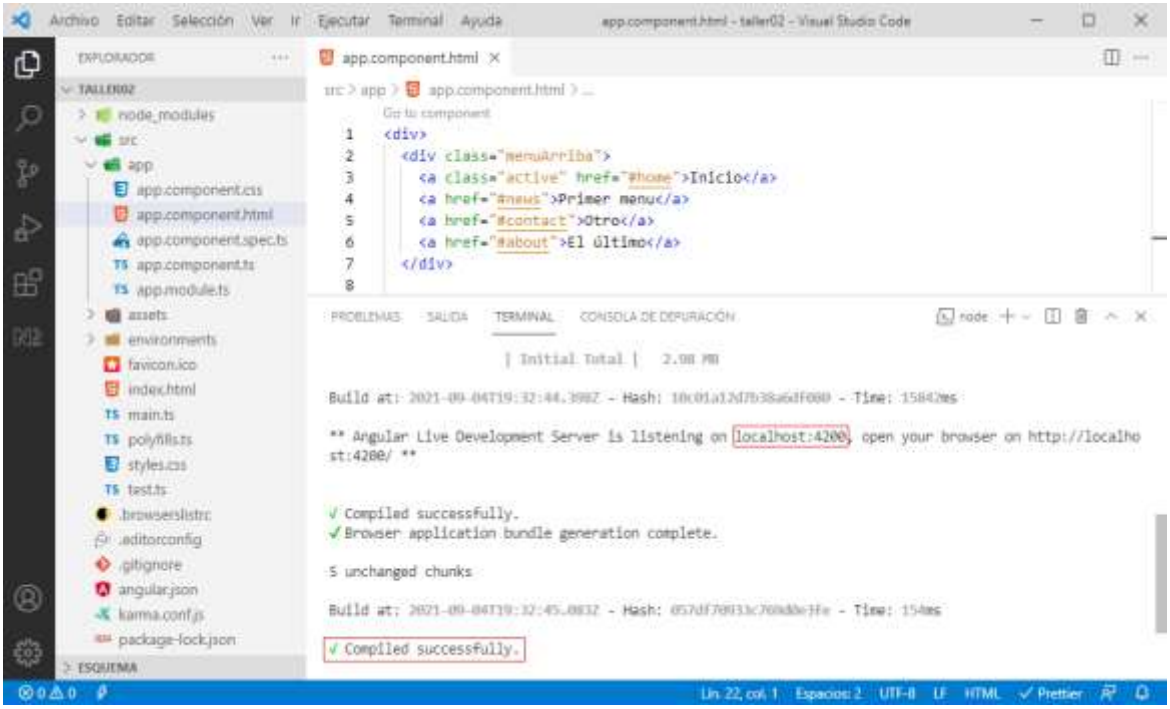
Imagen 42. Taller02, código HTML sugerido para el componente principal



Fuente: (Autor, 2021)

Una vez modificados los dos archivos, se procede a ejecutar el proyecto desde la terminal de Visual Estudio Code. Se selecciona el menú "***Terminal***" y luego la opción "***New Terminal***" o "Nueva terminal". Finalmente, se ejecuta el comando "***ng serve***". La Imagen 43 presenta el resultado de la ejecución del comando "***ng serve***".

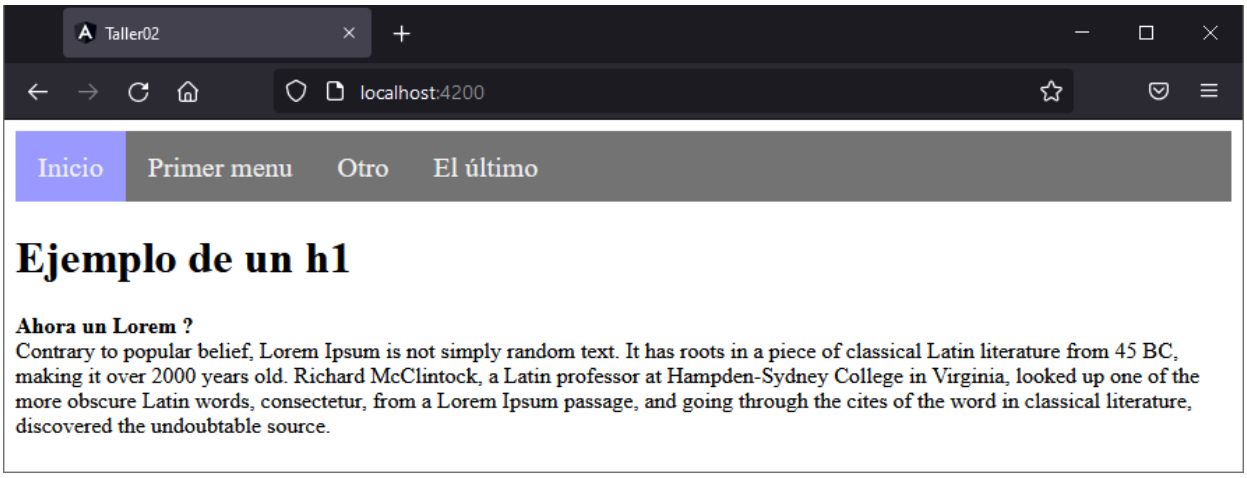
Imagen 43. Taller02, ejecución de ng serve



Fuente: (Autor, 2021)

Si todo es correcto, se puede ejecutar el proyecto en el navegador Firefox y se observará en pantalla una interfaz como la presentada en la Imagen 44.

Imagen 44. Taller02 desplegado en el navegador Firefox



Fuente: (Autor, 2021)

7.2.2. Taller03, creando múltiples componentes

7.2.2.1 Ubicación del proyecto

Como se presenta en la Imagen 45, los talleres se crearán en la carpeta: "***c:\angular\htdocs***".

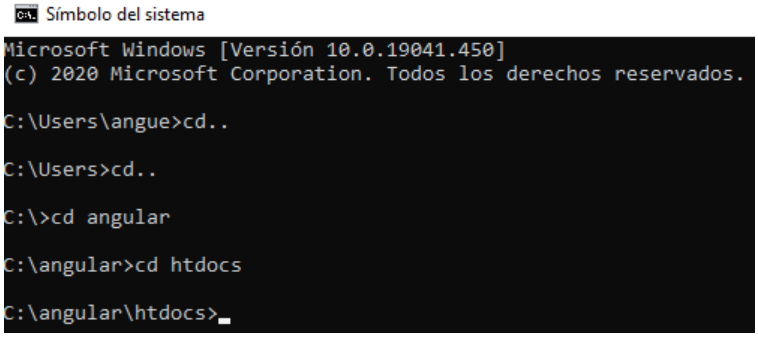
Imagen 45. Ruta para almacenar el taller03

Fuente: (Autor, 2021)

7.2.2.2 Creación del proyecto

Para crear el taller se debe abrir una consola comandos de Windows (CMD), luego se cambia la carpeta, asegurándose que la ruta final sea "c:\angular\htdocs". Imagen 46 presenta la consola de comandos con los pasos.

Imagen 46. Consola de comando para crear el taller03



Fuente: (Autor, 2021)

Idea clave

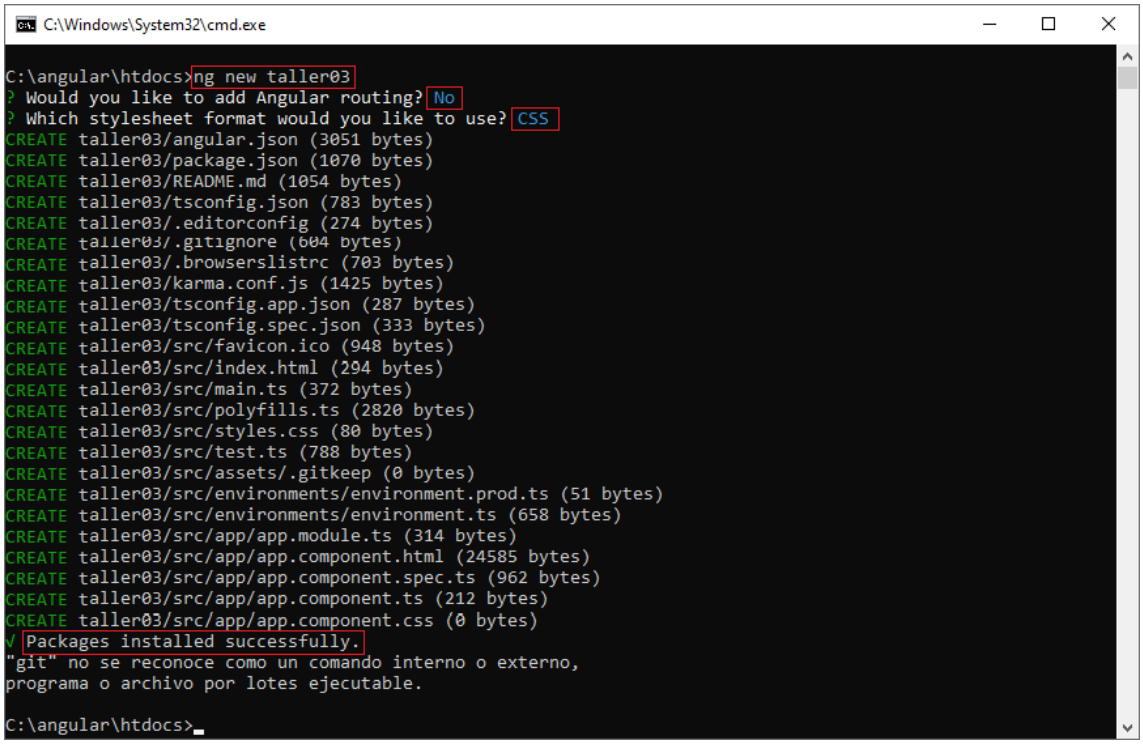
En estos primeros ejercicios no es necesario habilitar rutas en el proyecto, además, es recomendable escoger el estilo *CSS* para crear el proyecto

Una vez en la carpeta "***c:\angular\htdocs***" se debe ejecutar el comando:

```
C:\angular\htdocs> ng new taller03
```

El comando anterior creará la estructura básica de los archivos, conocida como "***Scaffolding***". La Imagen 47 presenta la ejecución del comando anterior.

Imagen 47. Resultado al crear el proyecto taller03



Fuente: (Autor, 2021)

7.2.2.3 Creación de componentes

Antes de crear los componentes se debe abrir el taller03 en el editor Visual Estudio Code.

Los componentes son parte esencial de angular, comprender que un componente puede ser una parte de una página, un menú, un pie de página o el mismo contenido, es algo que el desarrollador podrá entender luego de realizar varios ejercicios. Este taller propone la creación de 4 componentes, los cuales van a permitir comprender el concepto de la

decoración "***@component***" y su propiedad "***selector***". Desde la terminal de Visual Estudio Code se debe ejecutar el comando:

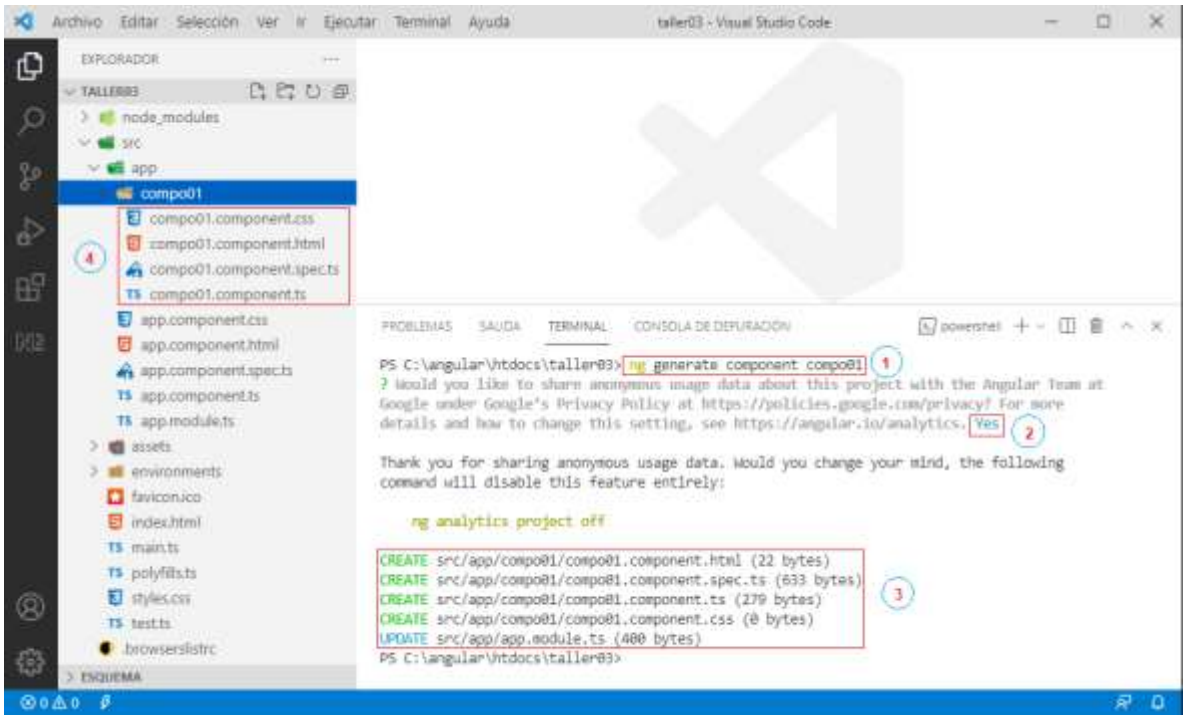
```
ng generate component compo01
```

O su abreviatura

```
ng g c compo01
```

Al ejecutar el comando se creará una carpeta en el proyecto en "***src\app\compo01***". La nueva carpeta tiene 4 archivos. La Imagen 48 presenta el proyecto, el nuevo componente y el resultado de la terminal.

Imagen 48. Resultado al crear un componente



Fuente: (Autor, 2021)

De la misma forma en que se creó el primer componente, se procede a ejecutar los comandos necesarios para crear 3 componentes más:

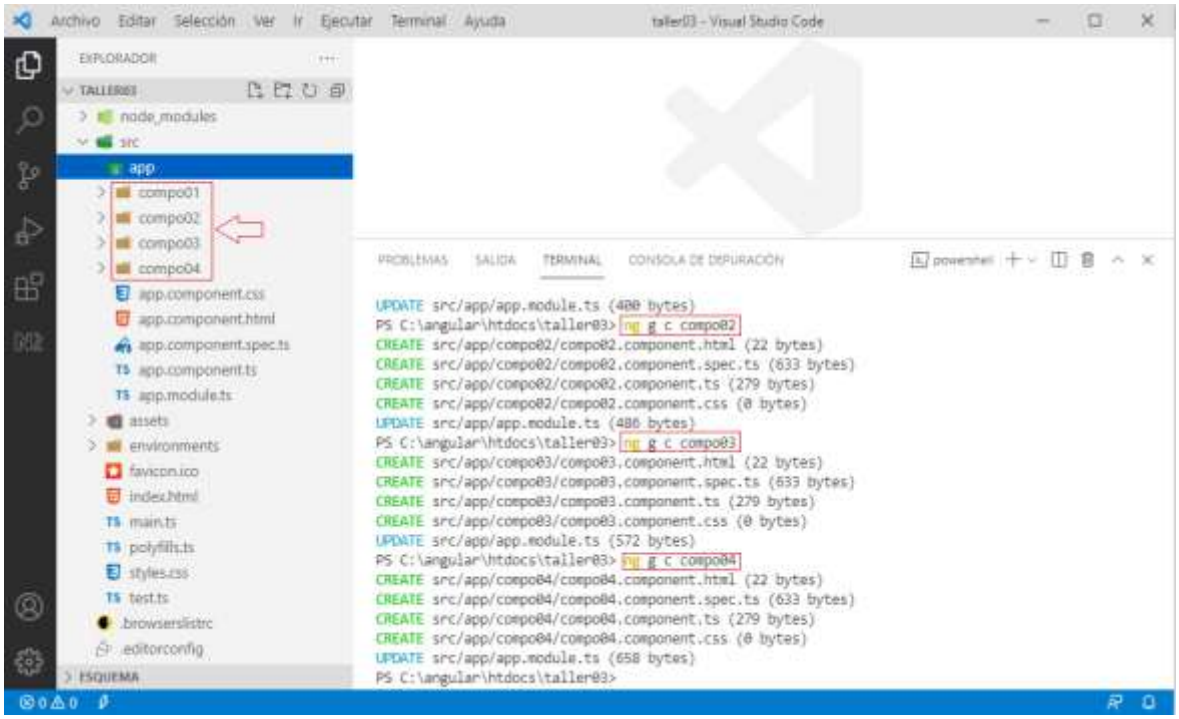
```
ng g c compo02
```

```
ng g c compo03
```

```
ng g c compo04
```

La Imagen 49 presenta la estructura del proyecto con los 4 componentes creados y los resultados de la terminal el Visual Studio Code.

Imagen 49. Taller003 con 4 componentes



Fuente: (Autor, 2021)

7.2.2.4 Configurando la aplicación y los componentes

En el archivo "***src\app\app.component.html***" se encuentra la plantilla principal del proyecto, todo el contenido de este archivo fue generado por angular CLI al crear el proyecto. Procedemos a eliminar todo el contenido del archivo. En su defecto vamos a colocar el siguiente código:

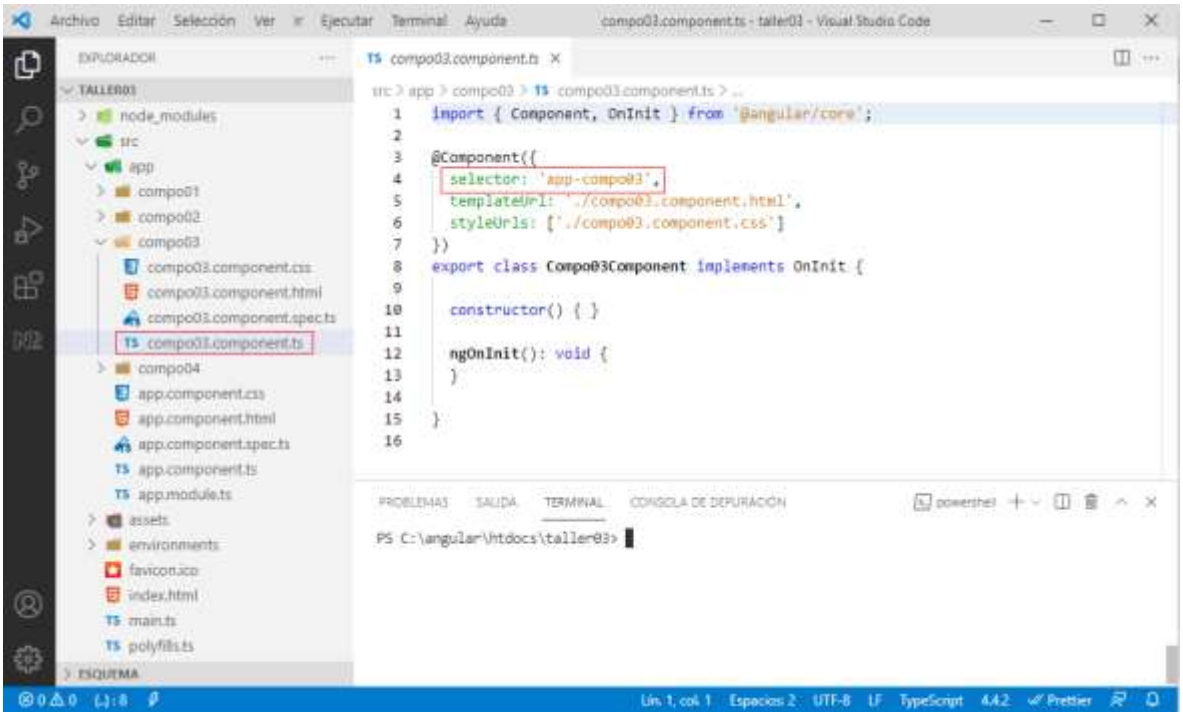
```
<div>
  <app-compo01></app-compo01>
  <app-compo02></app-compo02>
  <app-compo03></app-compo03>
  <app-compo04></app-compo04>
</div>
```

Tabla 6. Taller03, ejemplo para el archivo src\app\ app.component.html

El bloque de código anterior presenta en pantalla el contenido de todas las plantillas de cada componente, los nombres de los selectores están en los archivos de cada componente llamados "***compo0X.component.ts***". La

Imagen 50 presenta un ejemplo de la clase principal de uno de los componentes.

Imagen 50. Ejemplo de la clase principal del componente compo03



Fuente: (Autor, 2021)

7.2.2.5 Cambiando las plantillas de cada componente

En cada uno de los componentes existe una plantilla:

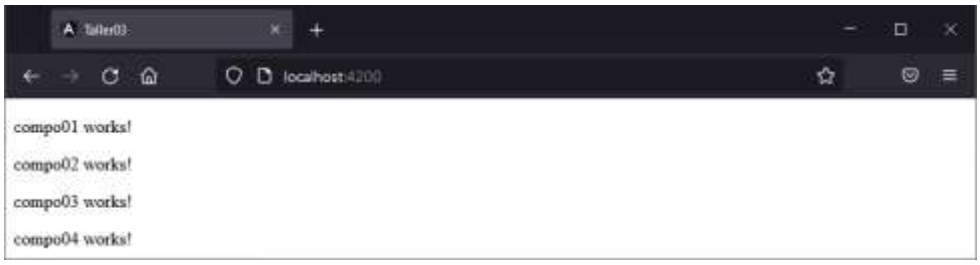
"***src\app\compo01\ compo01.component.html***"

"***src\app\compo02\ compo02.component.html***"
"***src\app\compo03\compo03.component.html***"
"***src\app\compo04\ compo04.component.html***"

Una verificación de cada uno de esos archivos permitirá encontrar un texto sencillo como "*** compo0X works! ".*** Todo el contenido que se coloque en estos archivos se presentará en el navegador de acuerdo con el orden de los selectores que se haya definido en la plantilla principal de la aplicación, es decir en el archivo "***src\app\app.component.html***".

Ejecutando el proyecto en la terminal, con el comando "***ng serve***", se puede visualizar el proyecto en ejecución en el navegador de su preferencia, el resultado se presenta en la Imagen 51.

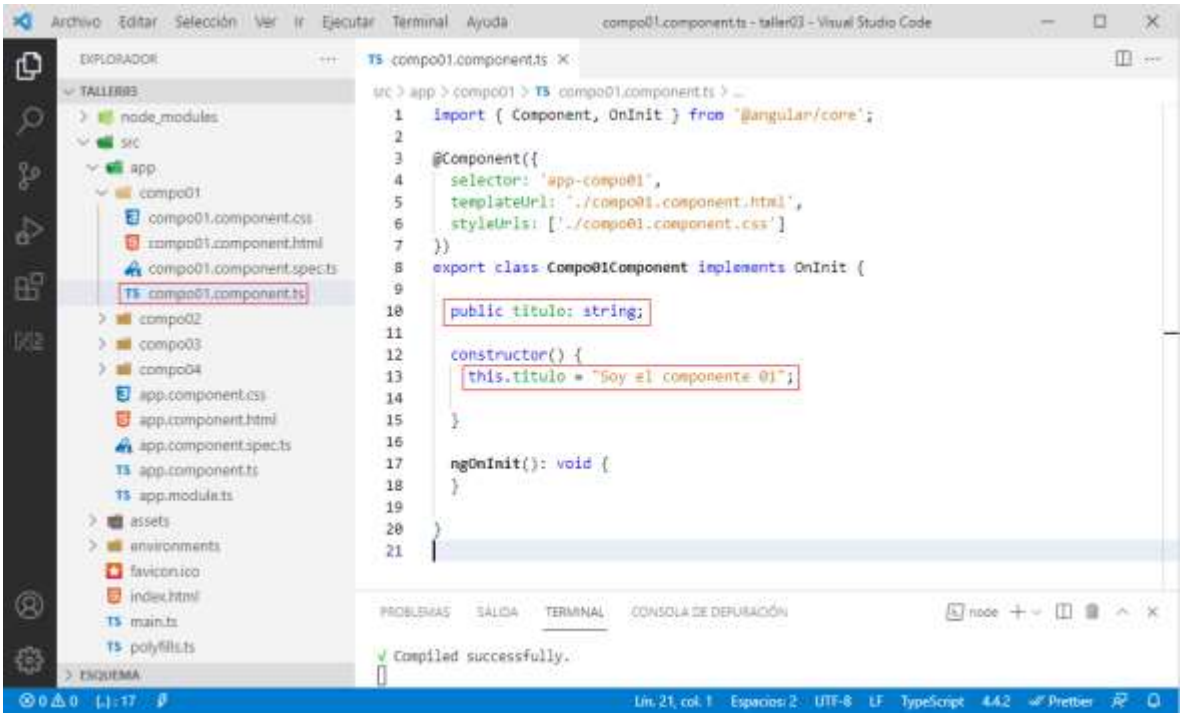
Imagen 51. Taller03 en ejecución



Fuente: (Autor, 2021)

A continuación, se agregará una propiedad o atributo a la clase principal de cada componente, de esta manera, se puede incluir código en las plantillas de cada componente por medio de la "***Interpolación***". Para realizar este cambio, se tomará como referencia la clase principal del primer componente, es decir "***src\app\ compo01\ compo01.component.ts***". La Imagen 52 presenta la propiedad que se debe adicionar en cada uno de los componentes para realizar el ajuste.

Imagen 52. Taller03, adición de propiedad en la clase de cada componente

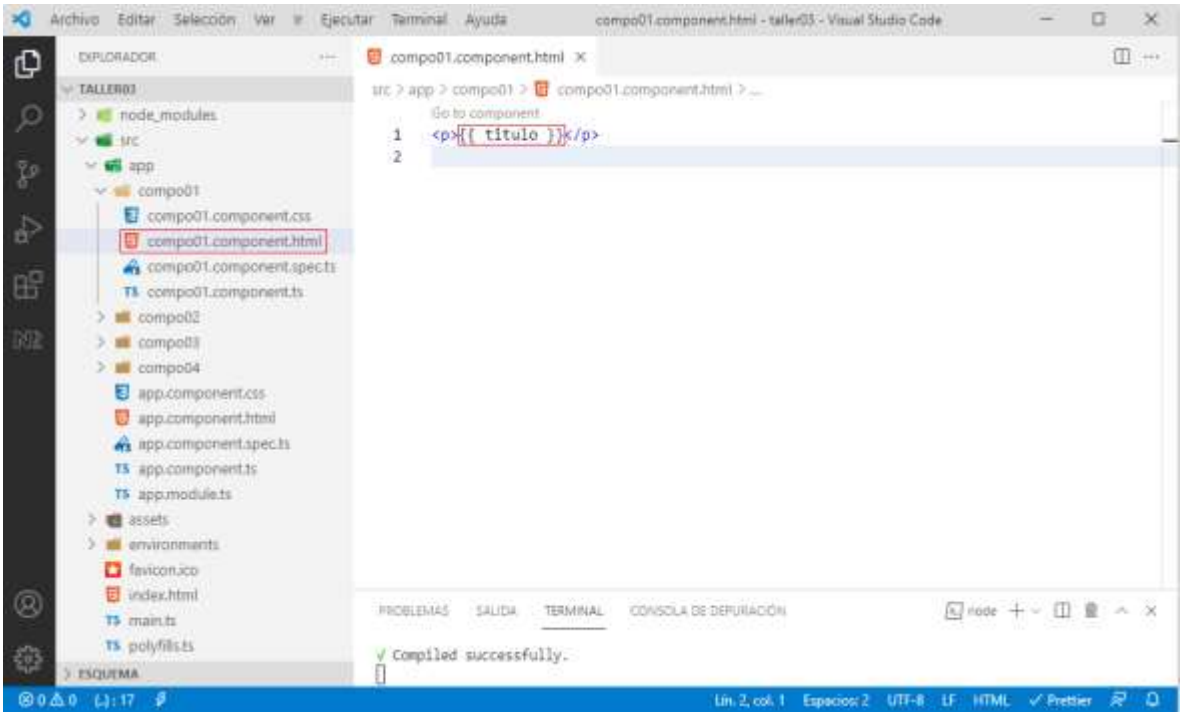


Fuente: (Autor, 2021)

La interpolación en Angular es la inyección de una cadena o valor en una plantilla, se conoce también como "***string interpolation***". Para realizar la interpolación solo basta con usar llaves dobles de apertura y de cierre "***{{ }}***" en la plantilla del componente deseado. La

Imagen 53 presenta cómo quedarían las plantillas de los componentes usando la interpolación.

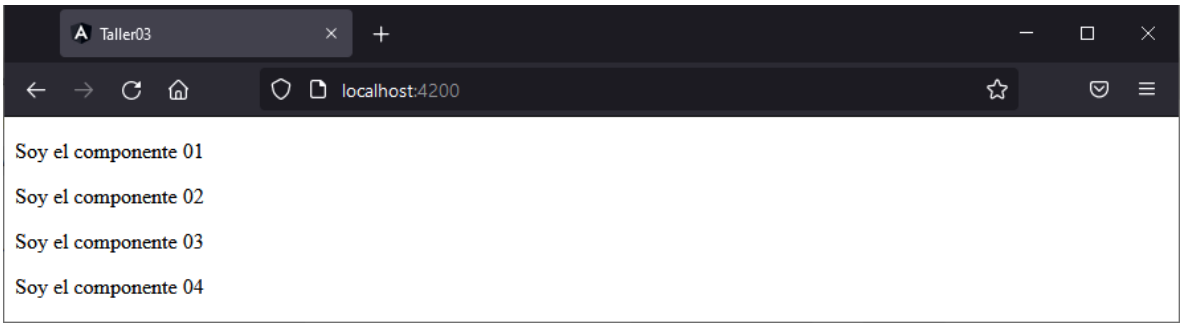
Imagen 53. Taller03, ejemplo interpolación en plantilla compo01



Fuente: (Autor, 2021)

Se puede verificar en el navegador de su preferencia, el resultado de los ajustes realizados. La Imagen 54 presenta el resultado al finalizar todos los ajustes sugeridos en esta sección.

Imagen 54. Taller003 en ejecución



Fuente: (Autor, 2021)

Idea clave

Dobles llaves {{ x }}: El valor entre ellas es llamado expresión. Se pueden crean expresiones simples o complejas, en este ejercicio se utilizó el valor de una propiedad como ejemplo, sin embargo, pueden ser valores calculados o métodos

7.3. Guía 3

7.3.1. Taller04 librerías Jquery y Bootstrap

7.3.1.1 Ubicación del proyecto

Los talleres se crearán en la carpeta: "c:\angular\htdocs", tal como se puede apreciar la Imagen 36.

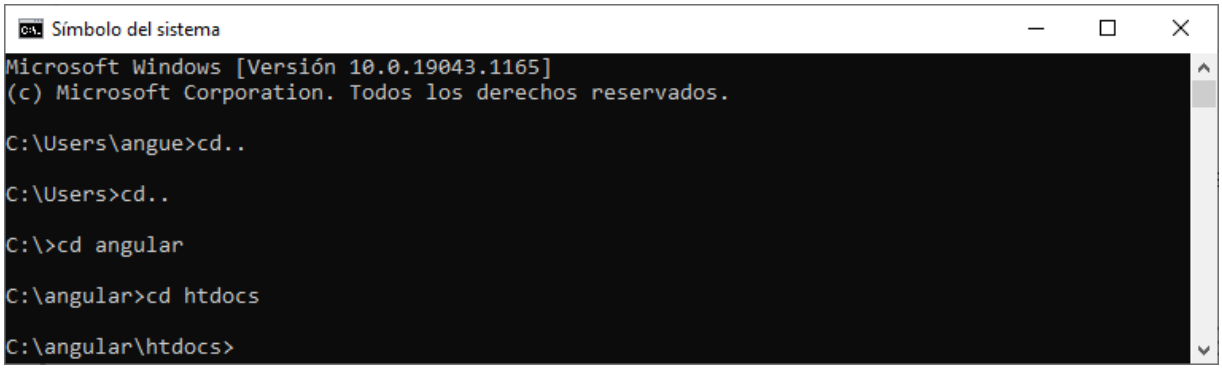
Imagen 55. Ruta para almacenar los proyectos

Fuente: (Autor, 2021)

7.3.1.2 Creación del proyecto

Para crear el taller se debe abrir una consola comandos de Windows (CMD), luego se cambia la carpeta, asegurándose que la ruta final sea "***c:\angular\htdocs***". Imagen 37 presenta la consola de comandos con los pasos.

Imagen 56. Consola de comando para crear el taller04



Fuente: (Autor, 2021)

Idea clave

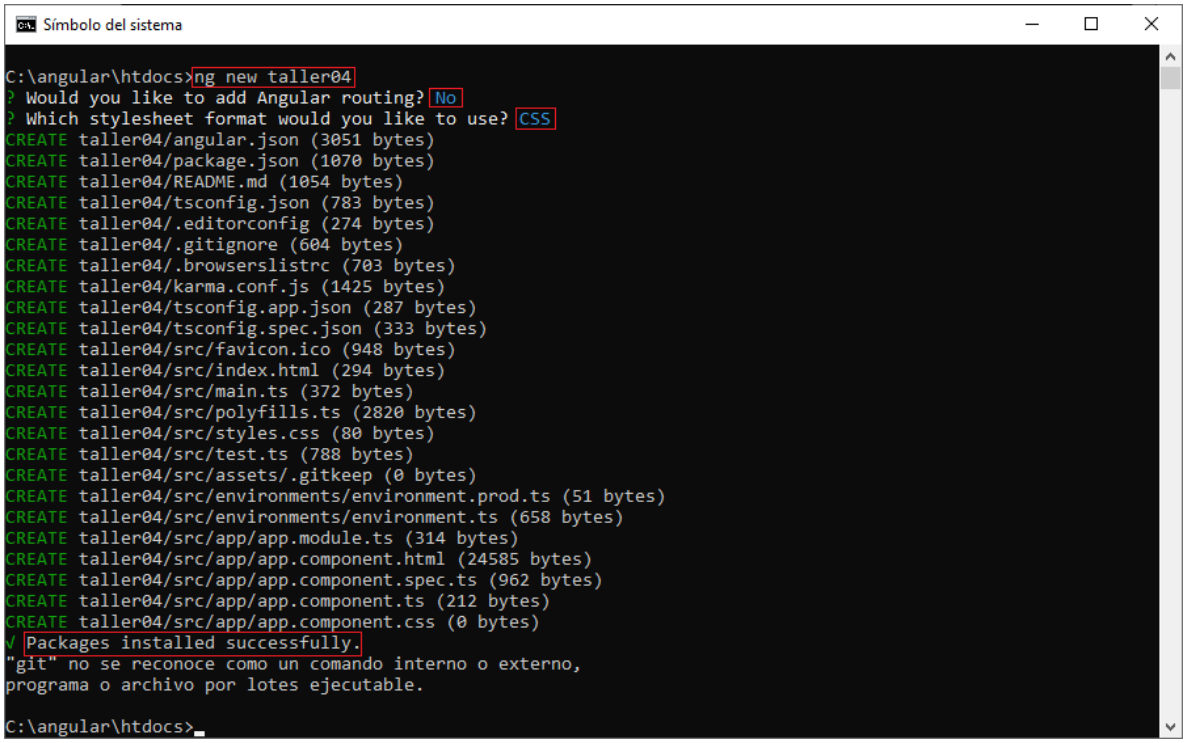
En estos primeros ejercicios no es necesario habilitar rutas en la creación del proyecto, además, es recomendable escoger el estilo *CSS* para al momento de crear el proyecto

Una vez en la carpeta "***c:\angular\htdocs***" se debe ejecutar el comando:

```
C:\angular\htdocs> ng new taller04
```

La Imagen 57 presenta el resultado de la creación del taller04.

Imagen 57. Creación del taller04



Fuente: (Autor, 2021)

7.3.1.3 Creación de componentes

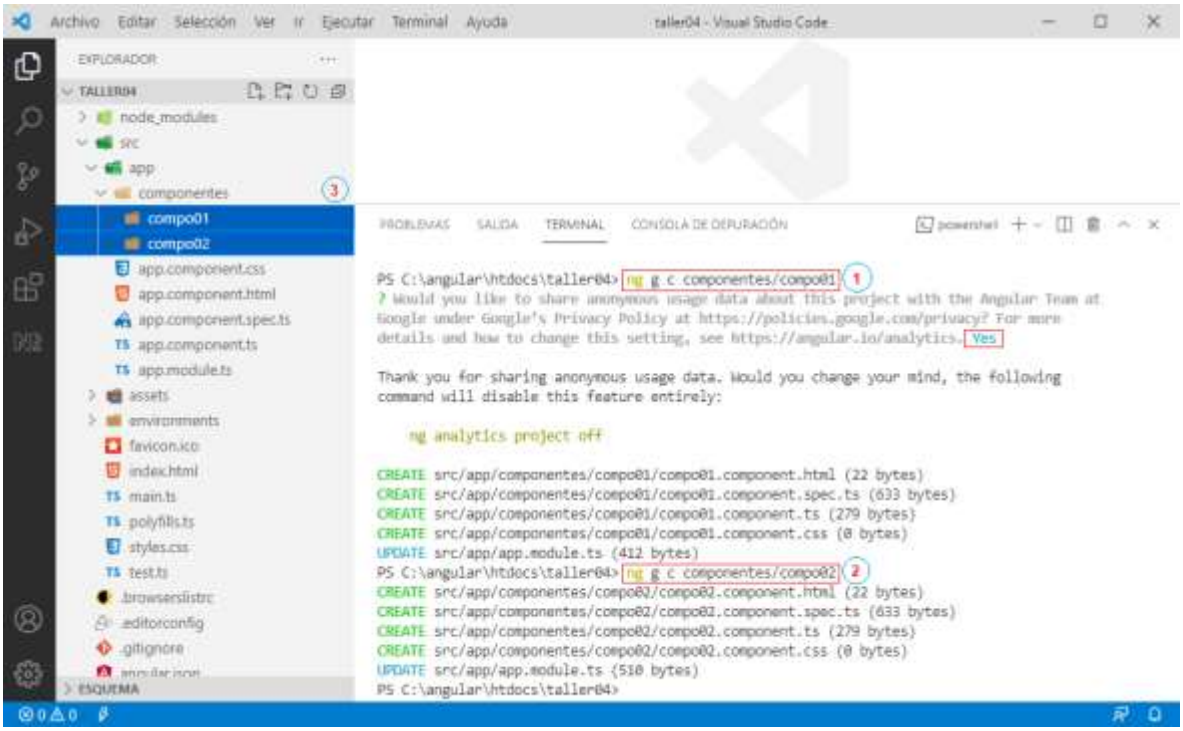
Se debe abrir la carpeta del taller04 en Visual Estudio Code. De esta manera se podrá utilizar la terminal de comandos. Para este taller, se van a crear dos componentes, por lo tanto, se deben ejecutar las siguientes instrucciones:

```
ng g c componentes/compo01
ng g c componentes/compo02
```

La

Imagen 58 presenta la carpeta de proyecto cargada en VSC y la ejecución de las instrucciones en la terminal.

Imagen 58. Creación de componentes en la terminal



Fuente: (Autor, 2021)

En el archivo "***src\app\app.component.html***" aparece el código HTML por defecto al crear el componente, todo este código se debe cambiar por el código presentado en la Tabla 7.

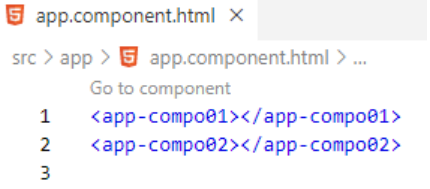
Imagen	Código reutilizable
	<app-compo01></app-compo01> <app-compo02></app-compo02>

Tabla 7. Taller004, selectores de los componentes creados

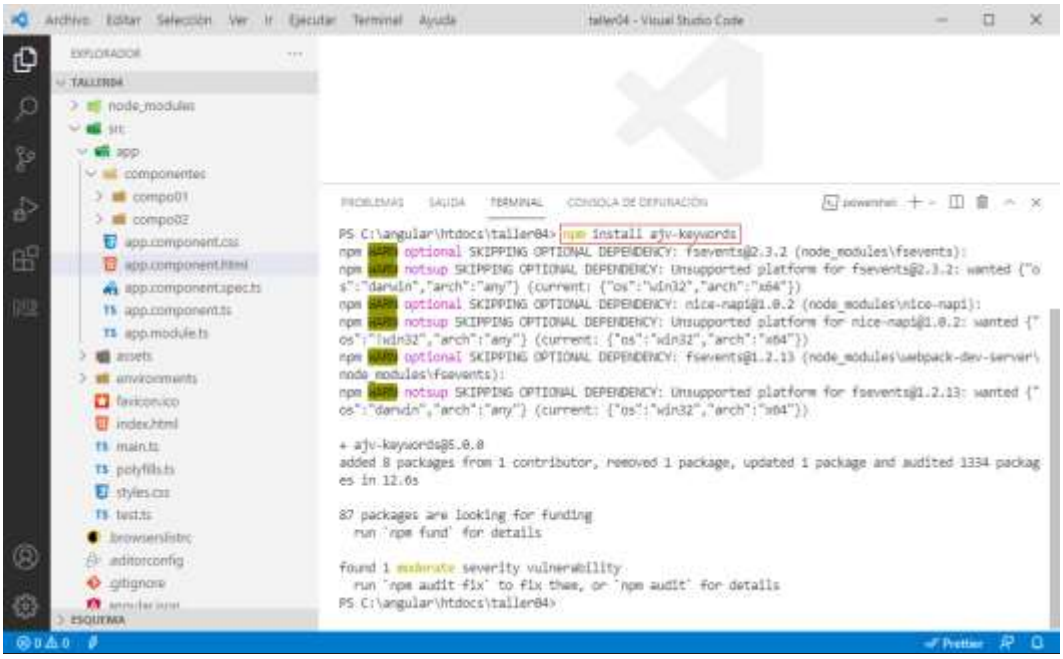
Para organizar un poco la presentación de la interfaz Web y no realizar estilos desde cero, en este taller se utilizará Bootstrap. Bootstrap es un framework de hojas de estilo que se utiliza en el desarrollo de aplicaciones frontend. Con el uso de este tipo de frameworks, el desarrollo frontend avanza con mayor velocidad.

Para instalar Bootstrap, es necesario instalar varias librerías adicionales, como por ejemplo JQuery.
JQuery es una librería de JavaScript de código abierto, la cual permite una interacción más sencilla entre el desarrollador y el código que se desea manipular con JavaScript. A su vez, Jquery requiere de otras librerías adicionales para su correcto funcionamiento. A continuación, se presenta el orden de instalación de las librerías y del framework Bootstrap.

```
npm install ajv-keywords
```

La librería "***AJV***" permite implementar una lógica compleja para la validación de datos en formato JSON a través de esquemas declarativos, sin necesidad de escribir código. "***ajv-keywords***" provee claves personalizadas parea el validador "***AJV***".

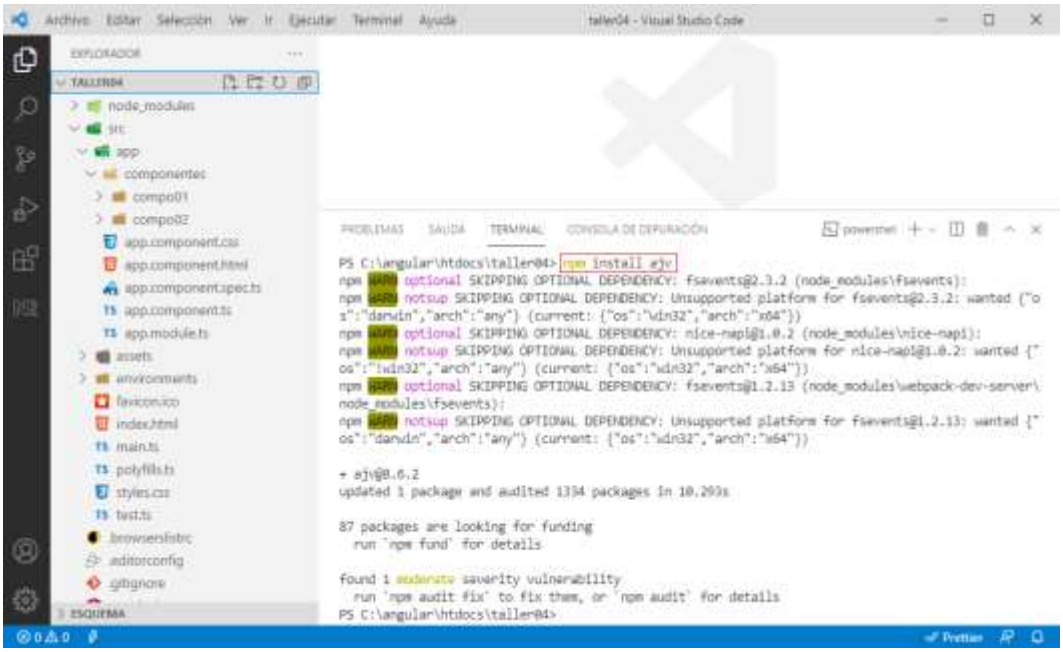
Imagen 59. Taller04, instalación de ajv-keywords



Fuente: (Autor, 2021)

```
npm install ajv
```

Imagen 60. Taller04, instalación de la librería AJV

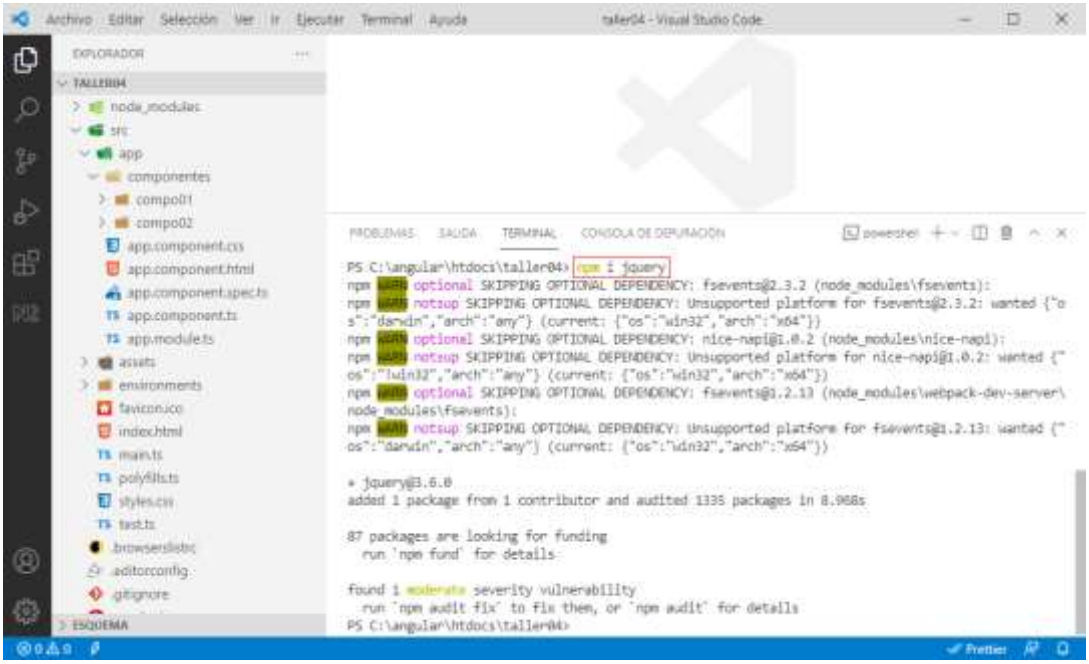


Fuente: (Autor, 2021)

El comodín "***install***" de los comandos anteriores se puede reemplazar por la letra "***i***":

```
npm i jquery
```

Imagen 61. Taller04, instalación de Jquery

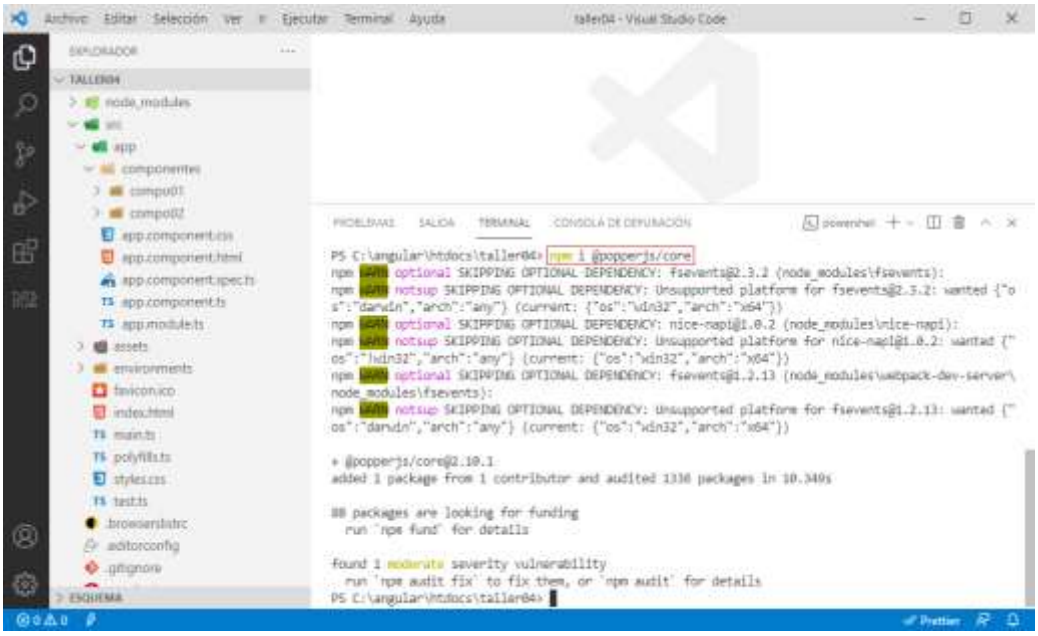


Fuente: (Autor, 2021)

```
npm i @popperjs/core
```

La "***popperJS***" es una librería que permite crear etiquetas o tooltips sobre un elemento con solo escribir una línea de código. El uso de esta librería potencia el uso de algunas opciones básicas de "***Bootstrap***".

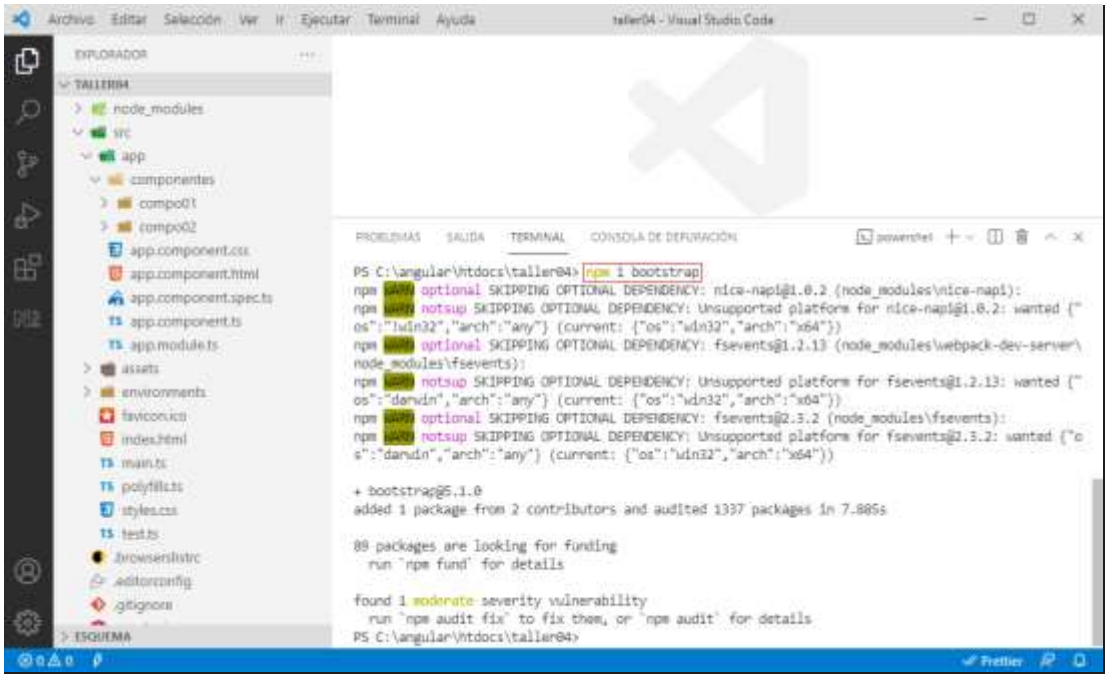
Imagen 62. Taller04, instalación de popperJS



Fuente: (Autor, 2021)

```
npm i Bootstrap
```

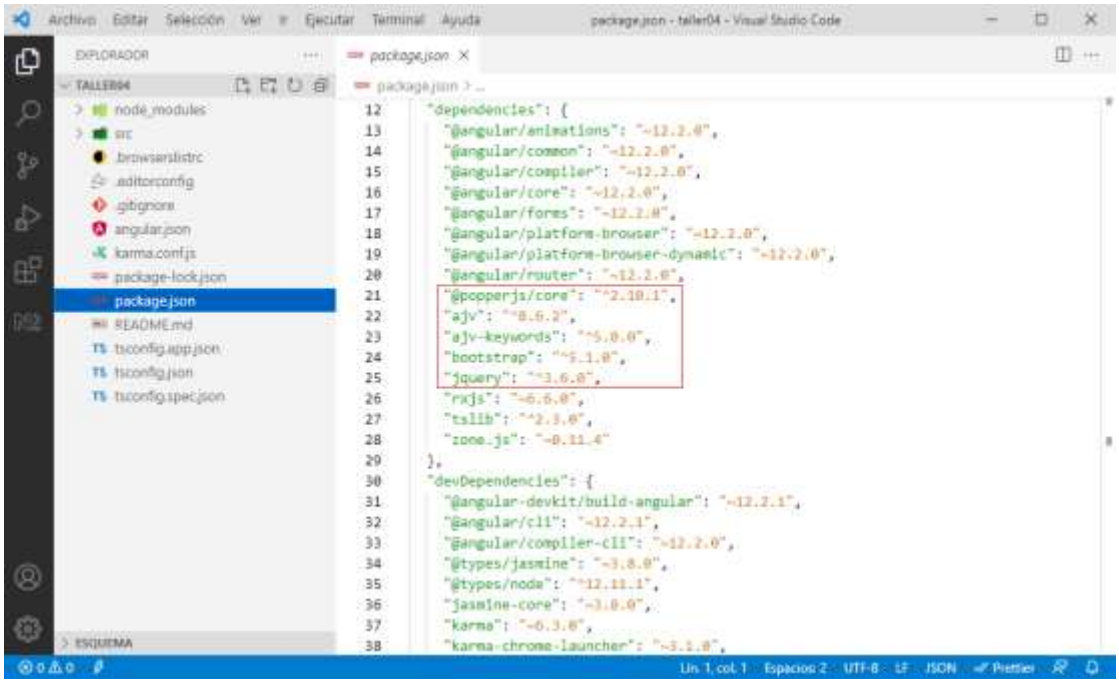
Imagen 63. Taller04, instalación de bootstrap



Fuente: (Autor, 2021)

De manera automática, en el archivo "***package.json***" aparecerán las referencias de las librerías y frameworks instalados con los comandos anteriores. La Imagen 64 presenta parte del contenido del "***package.json***"

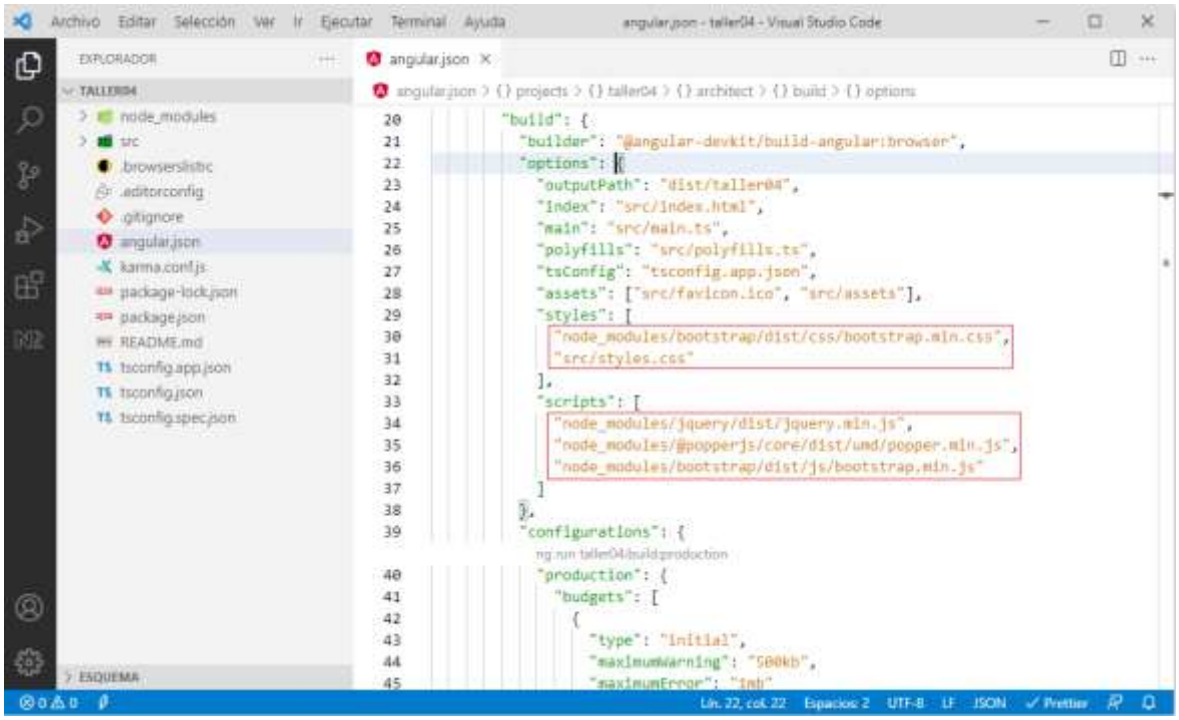
Imagen 64. Contenido del archivo package.json



Fuente: (Autor, 2021)

Un proceso que NO es automático es la inclusión de los archivos fuentes en el proyecto en desarrollo. Se debe editar el archivo "***angular.json***" e incluir las instrucciones presentadas en la Imagen 65.

Imagen 65. Taller04, inclusión de estilos y JavaScripts en angular.json



Fuente: (Autor, 2021)

Para efectos prácticos se comparte el código para copiarlo y pegarlo en el archivo "***angular.json***"

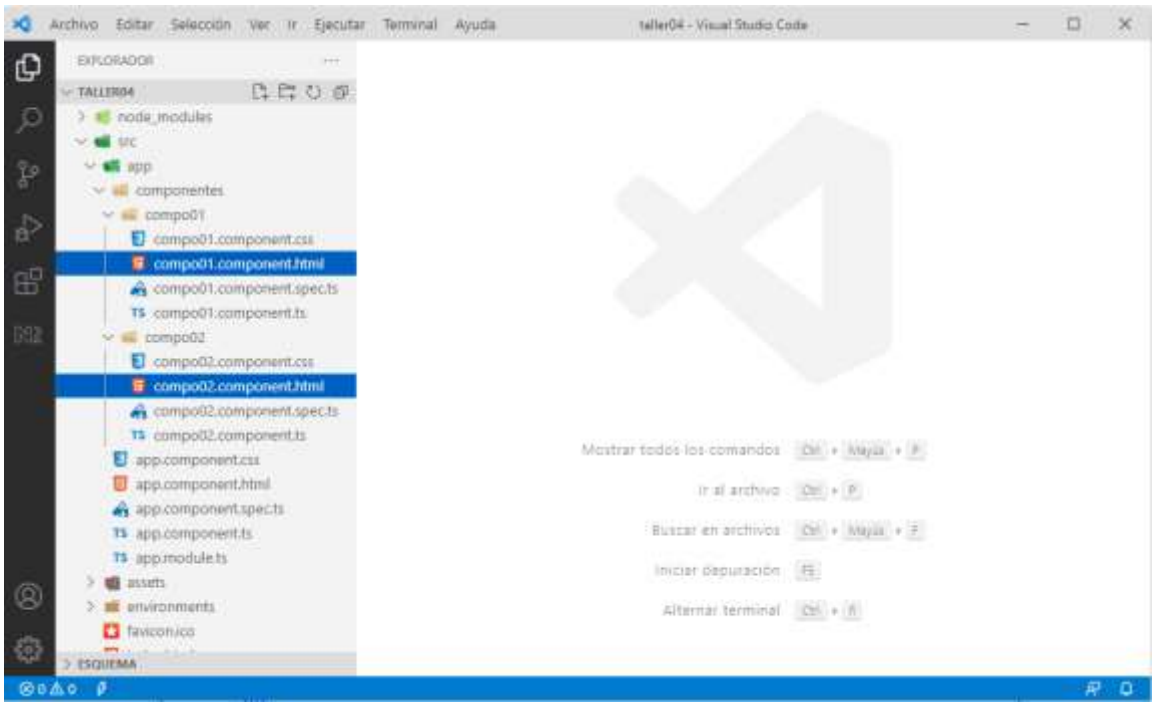
```
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/taller04",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": ["src/favicon.ico", "src/assets"],
            "styles": [
              "node_modules/bootstrap/dist/css/bootstrap.min.css",
              "src/styles.css"
            ],
            "scripts": [
              "node_modules/jquery/dist/jquery.min.js",
              "node_modules/@popperjs/core/dist/umd/popper.min.js",
```

```
            "node_modules/bootstrap/dist/js/bootstrap.min.js"
          ]
        },
```

Tabla 8. Taller04, Inclusión de estilos y JavaScripts en el archivo angular.json

En las carpetas de cada componente, se debe acceder a cada una de sus plantillas y modificar el código HTML. La Imagen 66 presenta los dos archivos de tipo plantilla.

Imagen 66. Taller04, plantillas HTML de los componentes



Fuente: (Autor, 2021)

A manera de ejemplo se sugiere el siguiente código para el archivo:

"***src\app\componentes\compo01\compo01.component.html***"

```
<div class="container" style="padding-top: 10px; padding-bottom: 10px">
  <div class="h-100 p-5 text-white bg-dark rounded-3">
    <h2>Change the background</h2>
    <p>
      Swap the background-color utility and add a `.text-*` color utility to mix
      up the jumbotron look. Then, mix and match with additional component
```

```
      themes and more.
    </p>
    <button class="btn btn-outline-light" type="button">Example button</button>
  </div>

  <div class="dropdown" style="padding-top: 10px">
    <button
      class="btn btn-secondary dropdown-toggle"
      type="button"
      id="dropdownMenuButton1"
      data-bs-toggle="dropdown"
      aria-expanded="false"
    >
      Dropdown button
    </button>
    <ul class="dropdown-menu" aria-labelledby="dropdownMenuButton1">
      <li><a class="dropdown-item" href="#">Action</a></li>
      <li><a class="dropdown-item" href="#">Another action</a></li>
      <li><a class="dropdown-item" href="#">Something else here</a></li>
    </ul>
  </div>
</div>
```

Tabla 9. Taller004, plantilla del componente compo01

A manera de ejemplo se sugiere el siguiente código para el archivo:

"***src\app\componentes\compo02\compo02.component.html***"

```
<div class="container">
  <div class="b-example-divider"></div>
  <footer class="">
    <p class="col-md-4 mb-0 text-muted">© 2021 Company, Inc</p>
  </footer>
</div>
```

Tabla 10. Taller004, plantilla del componente compo02

A manera de ejemplo se sugiere el siguiente código para el archivo:

"***src\app\componentes\compo02\ compo02.component.css***"

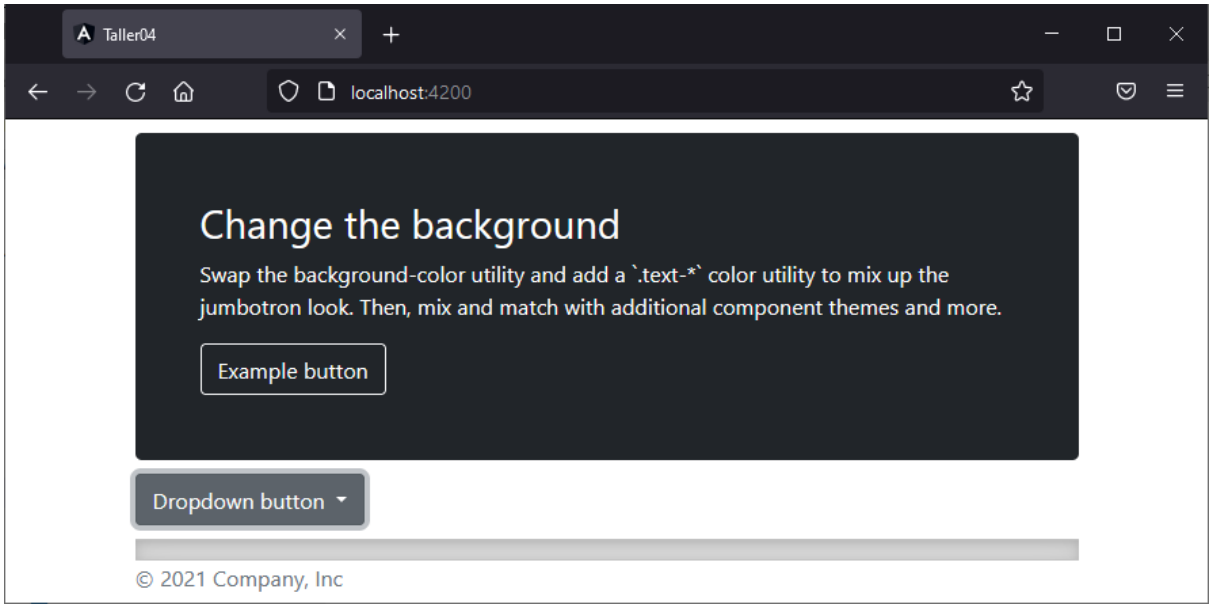
```
.b-example-divider {
  height: 1rem;
```

```
  background-color: rgba(0, 0, 0, 0.1);
  border: solid rgba(0, 0, 0, 0.15);
  border-width: 1px 0;
  box-shadow: inset 0 0.5em 1.5em rgba(0, 0, 0, 0.1),
    inset 0 0.125em 0.5em rgba(0, 0, 0, 0.15);
}
```

Tabla 11. Taller004, estilos del componente compo02

Finalizada la configuración de los componentes y la inclusión de los estilos y el maquetado de Bootstrap, se procede a ejecutar el proyecto con el comando “ng serve” desde la terminal de Visual Estudio Code. La Imagen 67 presenta el resultado del taller04 en el navegador Firefox.

Imagen 67. Taller04 en funcionamiento



Fuente: (Autor, 2021)

7.3.2. Taller05 íconos externos y puerto personalizado

7.3.2.1 Ubicación del proyecto

Los talleres se crearán en la carpeta: “***c:\angular\htdocs***”, tal como se puede apreciar la Imagen 68.

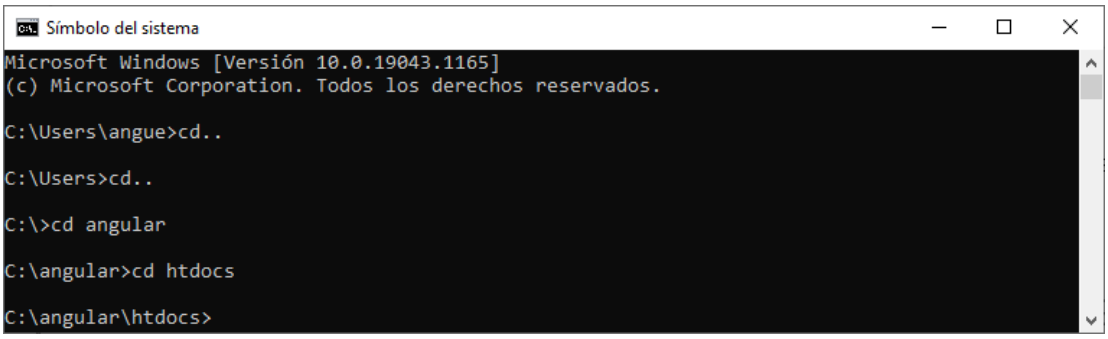
Imagen 68. Ruta para almacenar los proyectos

Fuente: (Autor, 2021)

7.3.2.2 Creación del proyecto

Para crear el taller se debe abrir una consola comandos de Windows (CMD), luego se cambia la carpeta, asegurándose que la ruta final sea “c:\angular\htdocs”. Imagen 69 presenta la consola de comandos con los pasos.

Imagen 69. Consola de comando para crear el proyecto taller005



Fuente: (Autor, 2021)

Idea clave

En estos primeros ejercicios no es necesario habilitar rutas en la creación del proyecto, además, es recomendable escoger el estilo *CSS* al momento de crear el proyecto

Una vez en la carpeta “***c:\angular\htdocs***” se debe ejecutar el comando desde la consola de Windows:

```
ng new taller05
```

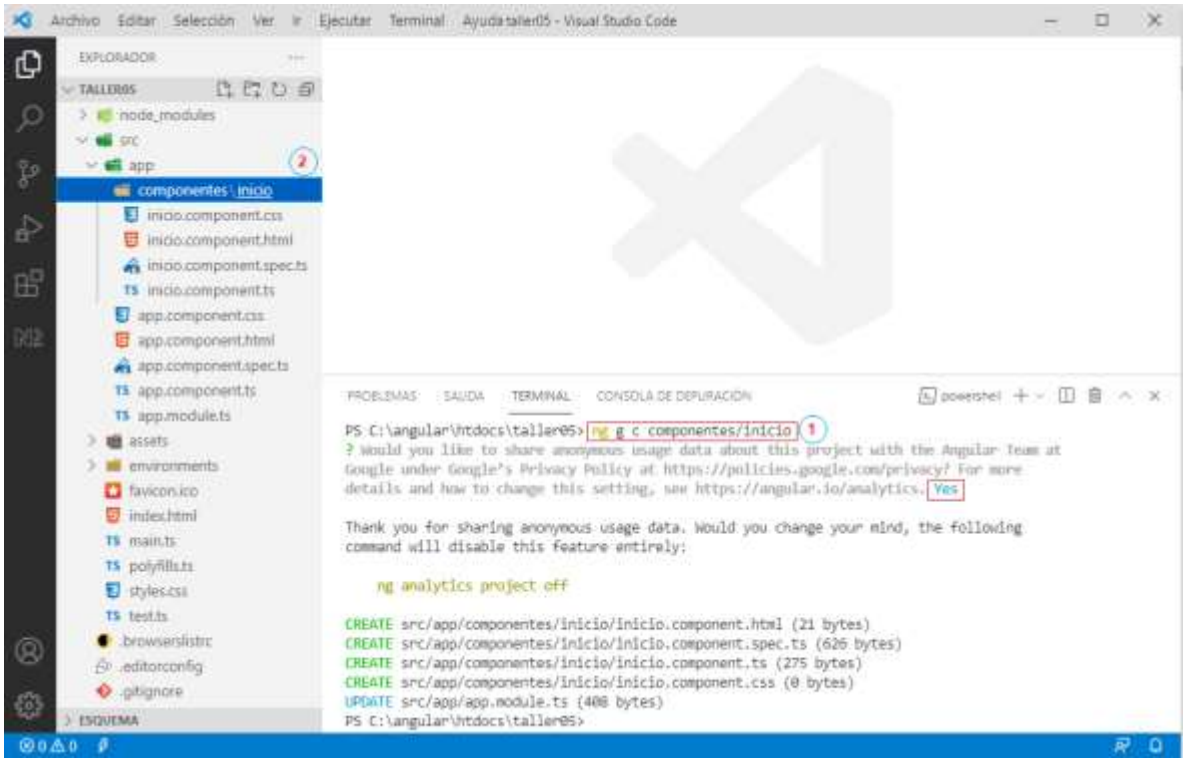
7.3.2.3 Creación de componentes

Después de crear el taller05, se procede a abrir la carpeta del proyecto en Visual Estudio Code. Para este taller, se creará un solo componente, para esto ejecutamos la siguiente instrucción desde la terminal de Visual Estudio Code.

```
ng g c componentes/inicio
```

La Imagen 70 presenta la carpeta de proyecto cargada en Visual Studio Code, así como la creación del componente inicio.

Imagen 70. Taller05, creación del componente inicio



Fuente: (Autor, 2021)

En el archivo "***src\app\app.component.html***" aparece el código HTML por defecto al crear el componente, todo este código se debe cambiar por el siguiente código:

Imagen	Código reutilizable

app.component.html × src > app > app.component.html > app-inicio Go to component 1 <app-inicio></app-inicio>	<app-inicio></app-inicio>

Tabla 12. Taller005, selector del componente inicio

Para organizar un poco la presentación de la interfaz Web y no realizar estilos desde cero, en este taller se utilizará Bootstrap. Bootstrap es un framework de hojas de estilo que se utiliza en el desarrollo de aplicaciones frontend. Con el uso de este tipo de frameworks, el desarrollo frontend avanza con mayor velocidad.

7.3.2.4 Instalación de librerías

A continuación, se presenta el orden de instalación de las librerías Y framework a utilizar en el presente taller.

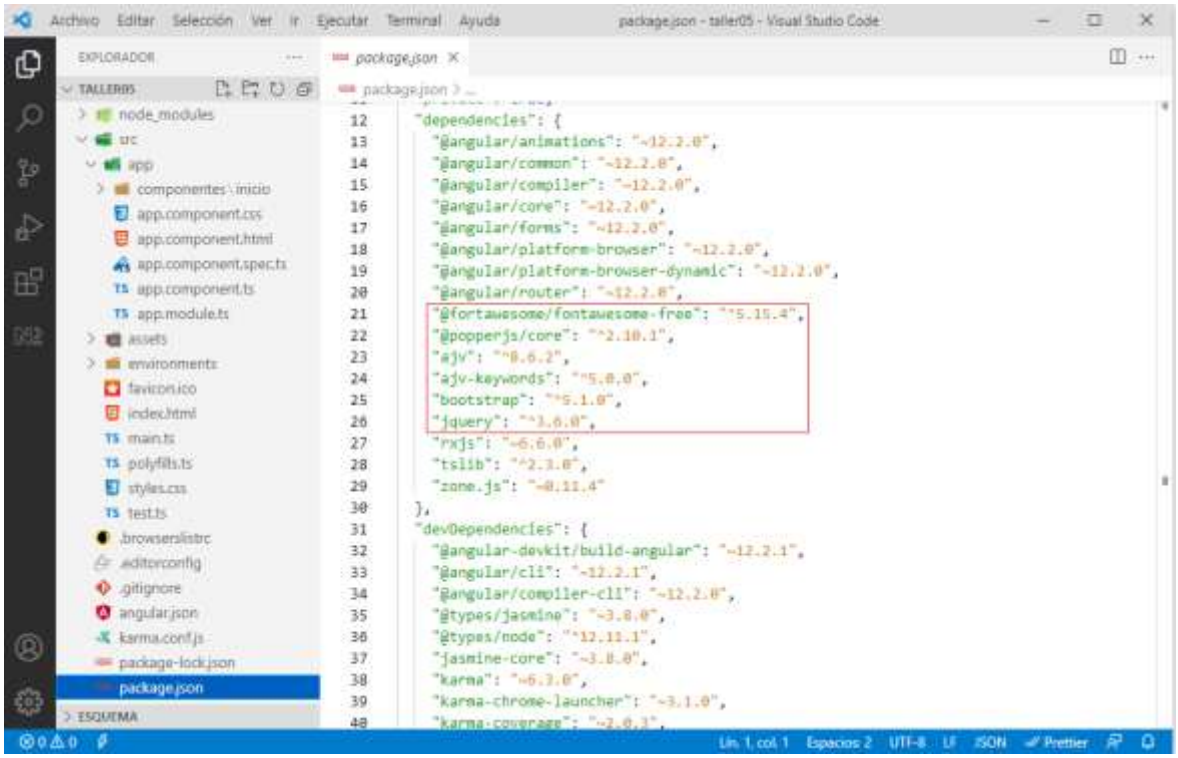
```
npm i ajv-keywords
npm i ajv
npm i jquery
npm i @popperjs/core
npm i bootstrap
```

Adicionalmente a las librerías ya utilizadas, en este taller se incluirá una librería para permitir el acceso a iconos de la librería fontawesome. El comando para su instalación es el siguiente:

```
npm i @fortawesome/fontawesome-free
```

De manera automática, en el archivo "***package.json***" aparecerán las referencias de las librerías y frameworks instalados con los comandos anteriores. La Imagen 71 presenta parte del contenido del "***package.json***"

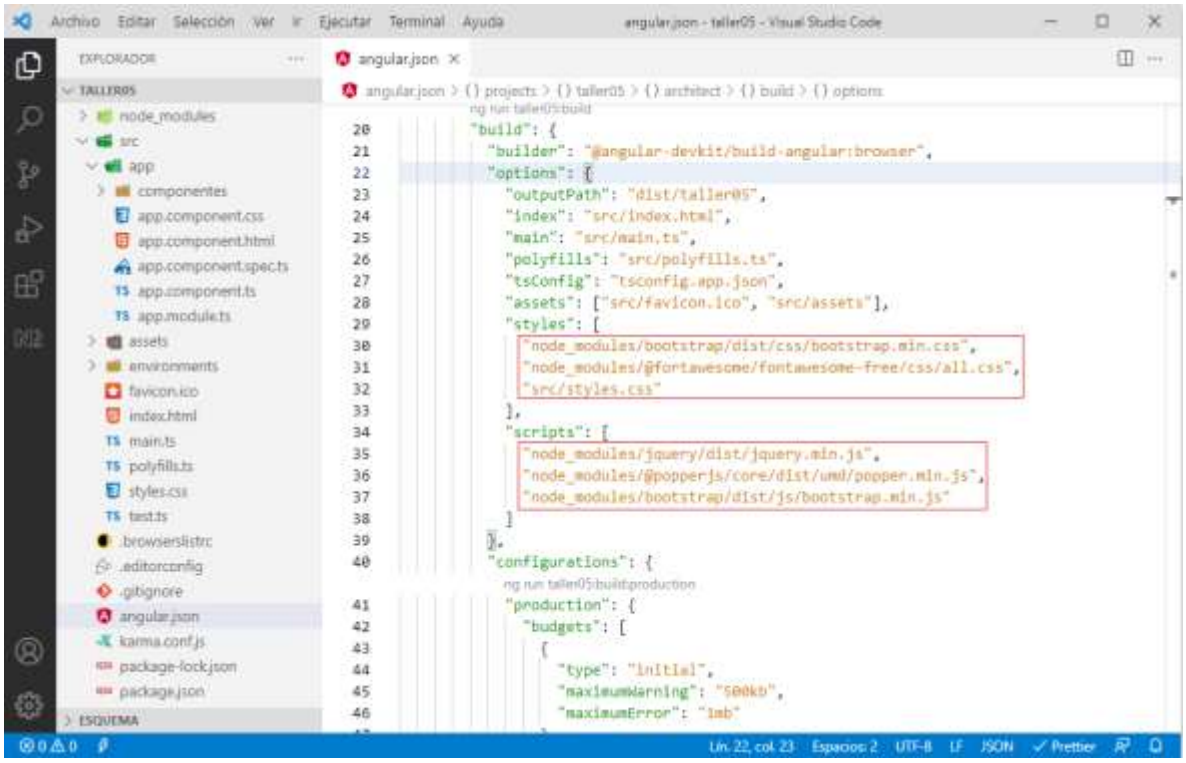
Imagen 71. Taller05, Contenido del archivo package.json



Fuente: (Autor, 2021)

Es importante recordar que los archivos CSS y JS no se incluyen de manera automática en el archivo "***angular.json***" Se debe editar el archivo "***angular.json***" e incluir las instrucciones presentadas en la Imagen 72.

Imagen 72. Taller05, Contenido del archivo angular.json



Fuente: (Autor, 2021)

Para efectos prácticos se comparte el código para copiarlo y pegarlo en el archivo "***angular.json***"

```
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/taller05",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": ["src/favicon.ico", "src/assets"],
            "styles": [
              "node_modules/bootstrap/dist/css/bootstrap.min.css",
              "node_modules/@fortawesome/fontawesome-free/css/all.css",
              "src/styles.css"
            ],
            "scripts": [
              "node_modules/jquery/dist/jquery.min.js",
```

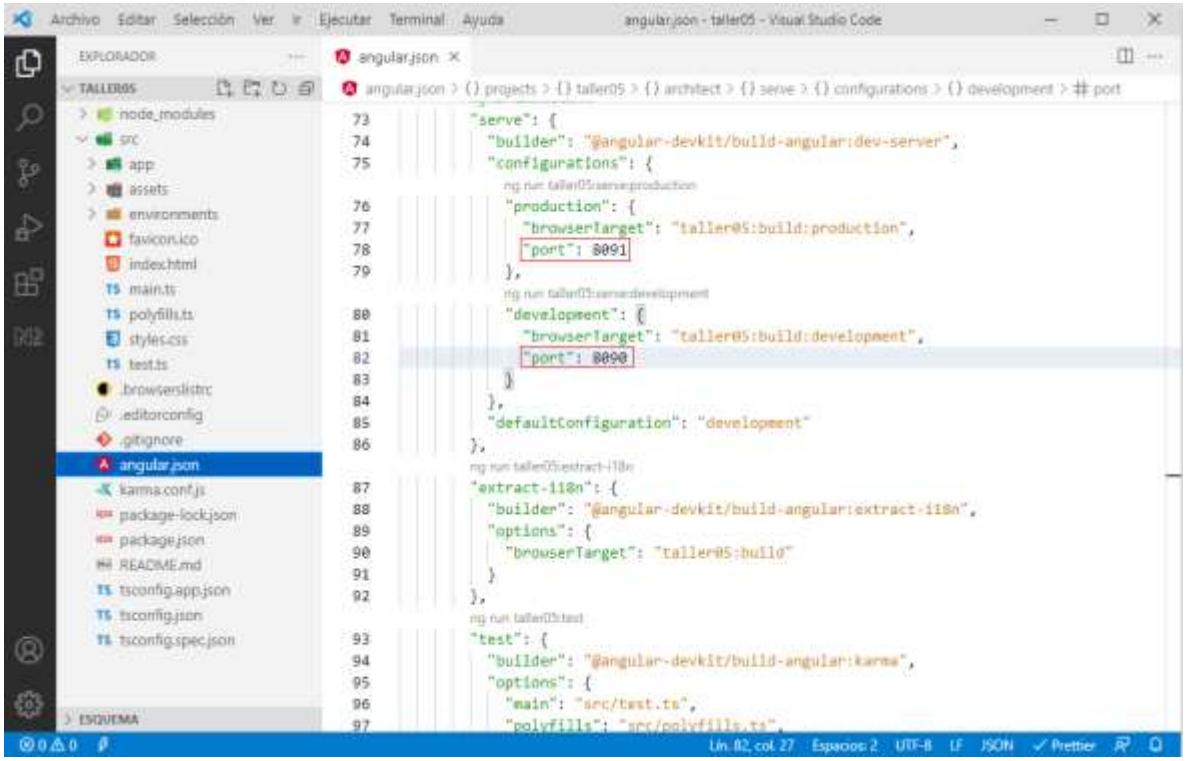
```
            "node_modules/@popperjs/core/dist/umd/popper.min.js",
            "node_modules/bootstrap/dist/js/bootstrap.min.js"
          ]
        },
```

Tabla 13. Taller005, inclusión de estilos y JavaScripts en angular.json

7.3.2.5 Configuración del puerto del proyecto

En el mismo archivo "***angular.json***" se puede configurar el puerto por defecto en donde se ejecutará el proyecto al ser visualizado en un navegador. Basta con agregar la propiedad "***port***" y asignar un valor válido de puerto. La Imagen 73 presenta la inclusión de la propiedad que permite definir el puerto en el cual se podrá visualizar el proyecto.

Imagen 73. Taller005, configuración del puerto en el archivo angular.json



Fuente: (Autor, 2021)

En la Tabla 14 se encuentra el bloque de código que se debe agregar para configurar los puertos en modo producción y desarrollo.

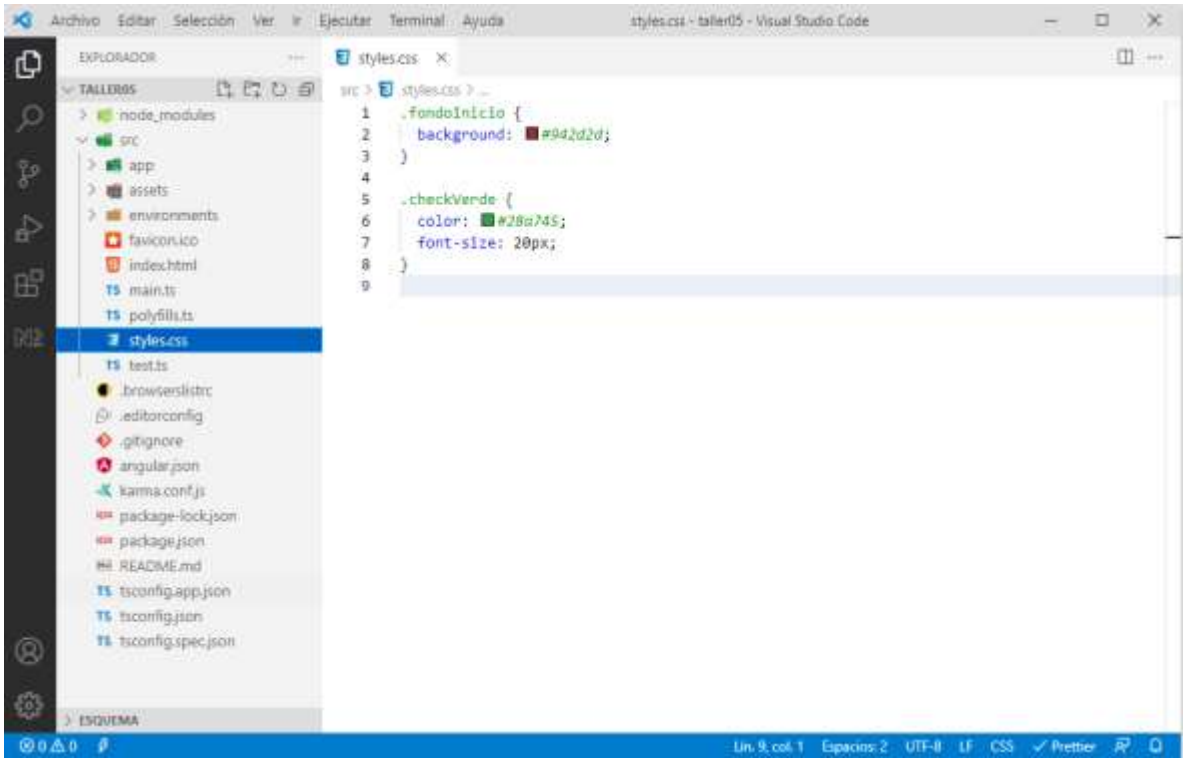
```
        "serve": {
          "builder": "@angular-devkit/build-angular:dev-server",
          "configurations": {
            "production": {
              "browserTarget": "taller005:build:production",
              "port": 8090
            },
            "development": {
              "browserTarget": "taller005:build:development",
              "port": 8090
            }
          },
          "defaultConfiguration": "development"
        },
```

Tabla 14. Taller005, configuración de puertos en angular.json

7.3.2.6 Estilos globales del taller005

A continuación, se agregará un estilo personalizado de manera global a la aplicación, esto quiere decir que cualquier componente interno tendrá acceso a los estilos definidos. Para definir estilos globales se debe incluir en el archivo "***src\styles.css***". La Imagen 74 presenta los estilos de ejemplo utilizados en este taller.

Imagen 74. Taller005, estilos globales del archivo src\styles.css



Fuente: (Autor, 2021)

A continuación, en la Tabla 15 se presenta el código del estilo utilizado:

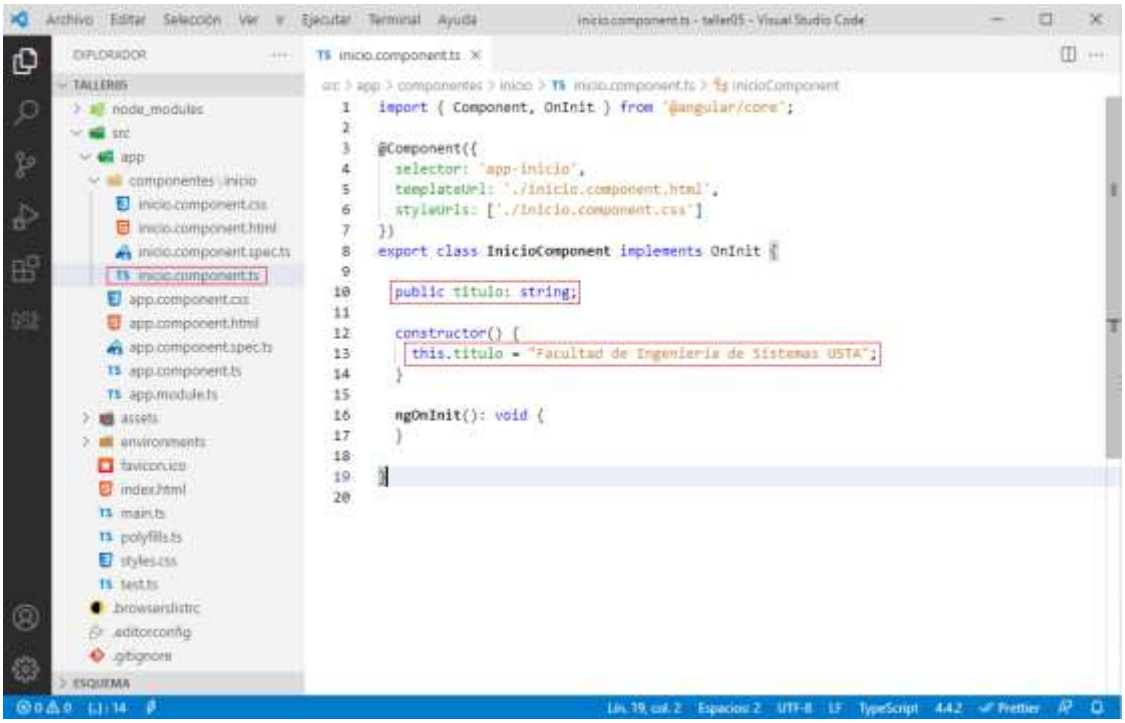
```
.fondoInicio {
  background: #942d2d;
}

.checkVerde {
  color: #28a745;
  font-size: 20px;
}
```

Tabla 15. Taller005, estilos globales del archivo src\styles.css

7.3.2.7 Ajuste a la plantilla del componente inicio

El objetivo de este taller es presentar una librería para la presentación de íconos, así como ajustar el puerto en donde se puede consumir la aplicación frontend desarrollada. La Imagen 75 presenta un ejemplo de interpolación utilizado con la propiedad título en la clase del componente inicio "***src\app\componentes\inicio\inicio.component.ts***":

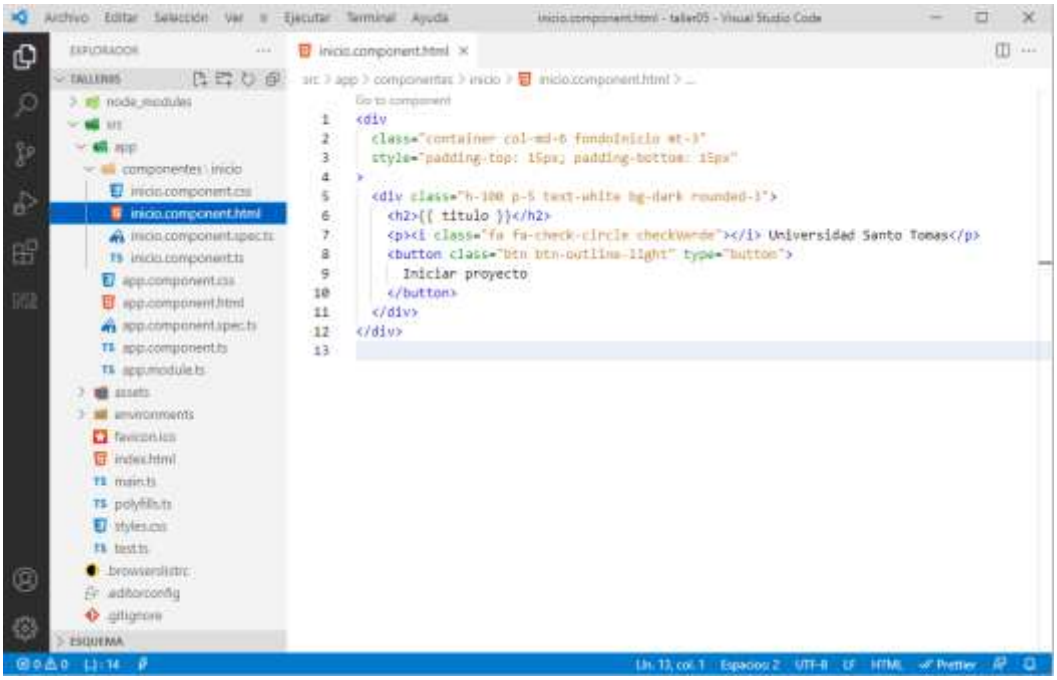
Imagen 75. Taller005, nueva propiedad de la clase del componente inicio



Fuente: (Autor, 2021)

La Imagen 76 presenta el código sugerido para visualizar el uso de la librería fontawesome y Bootstrap.

Imagen 76. Taller005, plantilla del componente inicio



Fuente: (Autor, 2021)

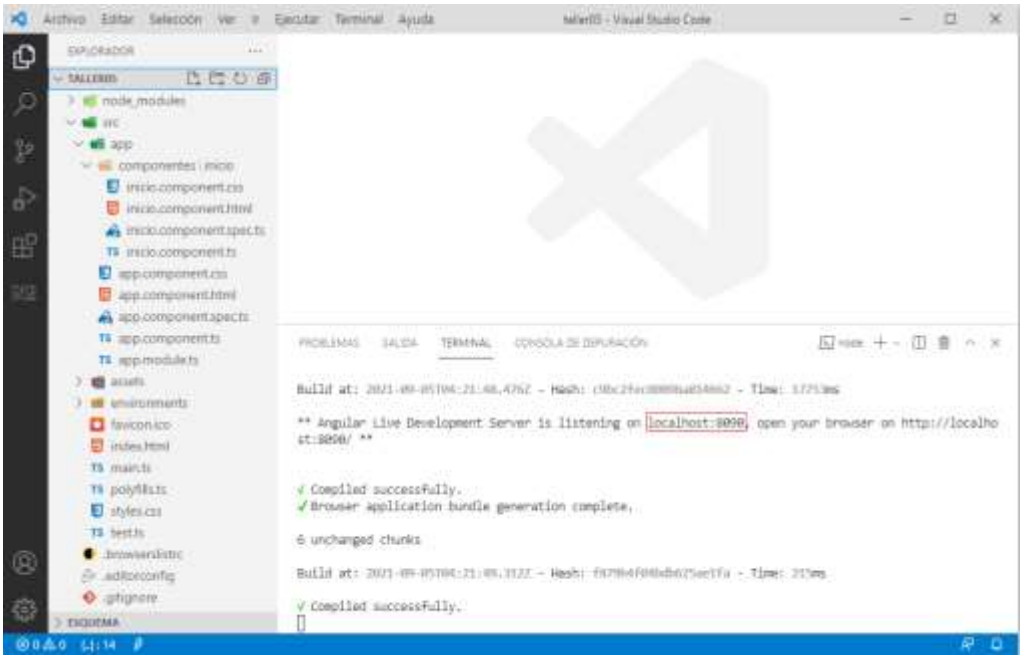
A continuación, en la Tabla 16 se presenta el código del estilo utilizado:

```
<div
  class="container col-md-6 fondoInicio mt-3"
  style="padding-top: 15px; padding-bottom: 15px"
>
  <div class="h-100 p-5 text-white bg-dark rounded-3">
    <h2>{{ titulo }}</h2>
    <p><i class="fa fa-check-circle checkVerde"></i> Universidad Santo Tomas</p>
    <button class="btn btn-outline-light" type="button">
      Iniciar proyecto
    </button>
  </div>
</div>
```

Tabla 16. Taller005, plantilla del componente inicio

Para ejecutar el proyecto y visualizar el resultado del taller05 se debe ejecutar "***ng serve***" en la terminal de Visual Estudio Code.

Imagen 77. Taller005, ejecución de ng serve modo desarrollo



Fuente: (Autor, 2021)

EL proyecto en ejecución se presenta en la Imagen 78.

Imagen 78. Taller005, ejecución de ng serve modo desarrollo

Fuente: (Autor, 2021)

7.4. Guía 4

7.4.1. Taller06 CRUD básico, modelos y mocks

7.4.1.1 Ubicación del proyecto

Los talleres se crearán en la carpeta: “c:\angular\htdocs”, tal como se puede apreciar la Imagen 36.

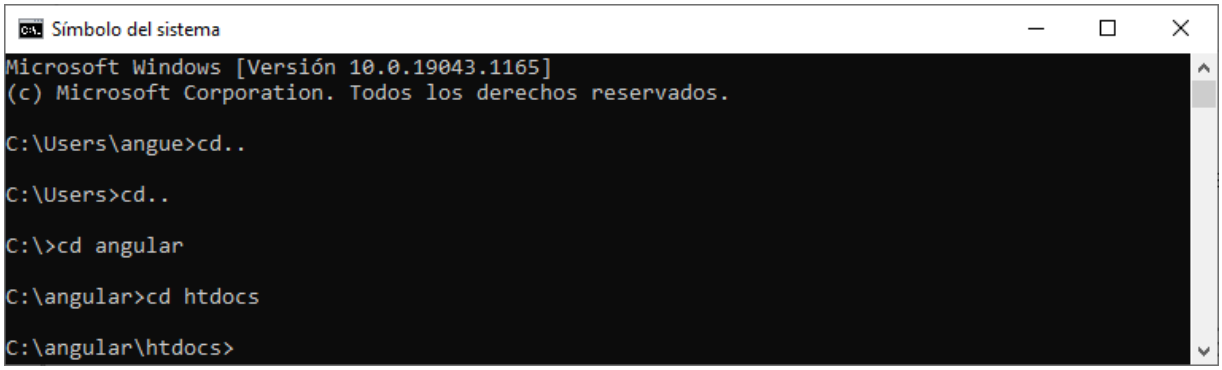
Imagen 79. Ruta para almacenar los proyectos

Fuente: (Autor, 2021)

7.4.1.2 Creación del proyecto

Para crear el taller se debe abrir una consola comandos de Windows (CMD), luego se cambia la carpeta, asegurándose que la ruta final sea “***c:\angular\htdocs***”. Imagen 37 presenta la consola de comandos con los pasos.

Imagen 80. Consola de comando para crear el taller06



Fuente: (Autor, 2021)

Idea clave

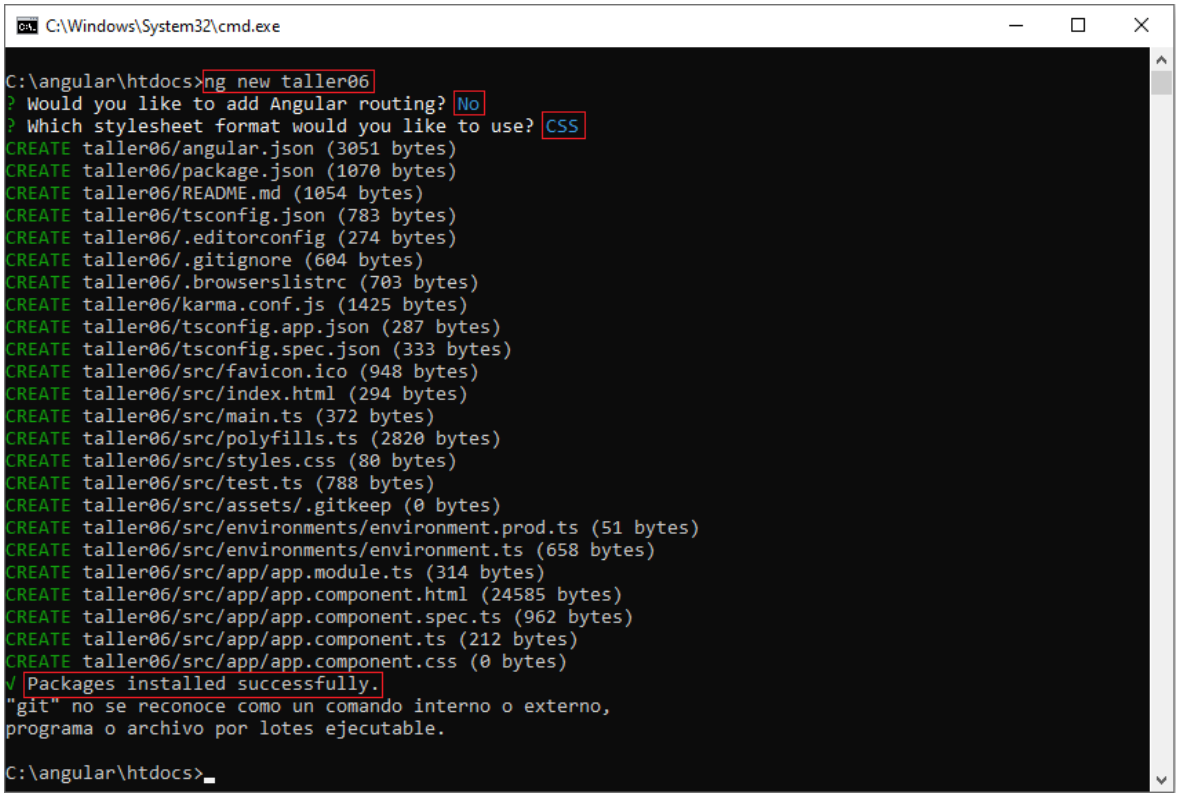
En estos primeros ejercicios no es necesario habilitar rutas en la creación del proyecto, además, es recomendable escoger el estilo *CSS* para al momento de crear el proyecto

Una vez en la carpeta "***c:\angular\htdocs***" se debe ejecutar el comando:

```
C:\angular\htdocs> ng new taller06
```

La Imagen 57 presenta el resultado de la creación del taller06.

Imagen 81. Creación del taller06



Fuente: (Autor, 2021)

7.4.1.3 Creación de componentes

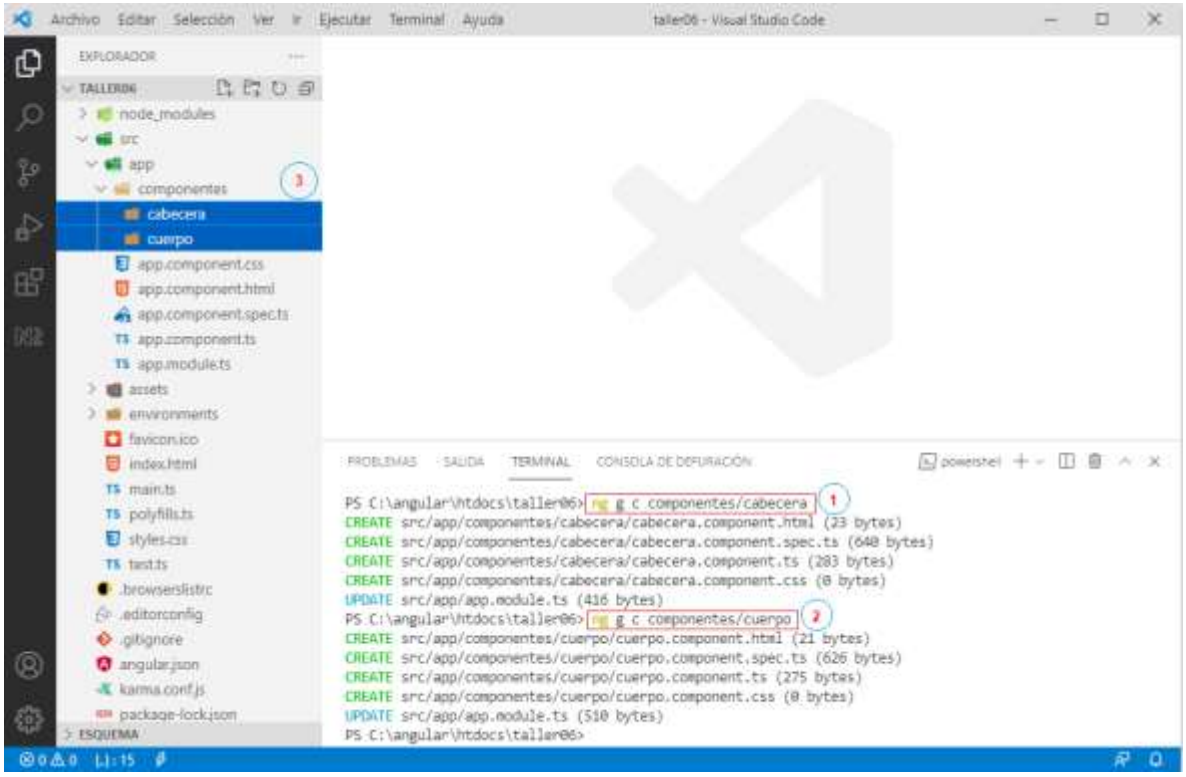
Se debe abrir la carpeta del taller06 en Visual Estudio Code. De esta manera se podrá utilizar la terminal de comandos. Para este taller, se van a crear dos componentes, por lo tanto, se deben ejecutar las siguientes instrucciones:

```
ng g c componentes/cabecera
ng g c componentes/cuerpo
```

La

Imagen 58 presenta la carpeta de proyecto cargada en Visual Estudio Code y la ejecución de las instrucciones en la terminal.

Imagen 82. Taller06, Creación de componentes en la terminal



Fuente: (Autor, 2021)

En el archivo "***src\app\app.component.html***" aparece el código HTML por defecto al crear el componente, todo este código se debe cambiar por el código presentado en la Tabla 7.

Imagen	Código reutilizable

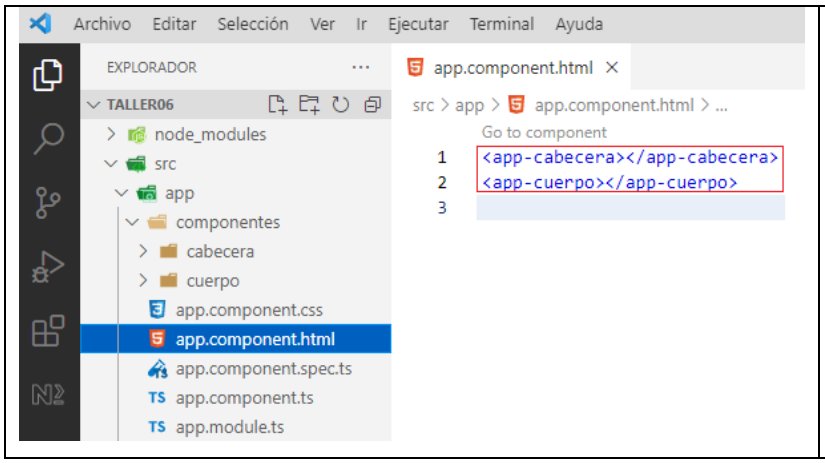


Tabla 17. Taller06, selectores de los componentes creados

Para organizar un poco la presentación de la interfaz Web y no realizar estilos desde cero, en este taller se utilizará Bootstrap. Bootstrap es un framework de hojas de estilo que se utiliza en el desarrollo de aplicaciones frontend. Con el uso de este tipo de frameworks, el desarrollo frontend avanza con mayor velocidad.

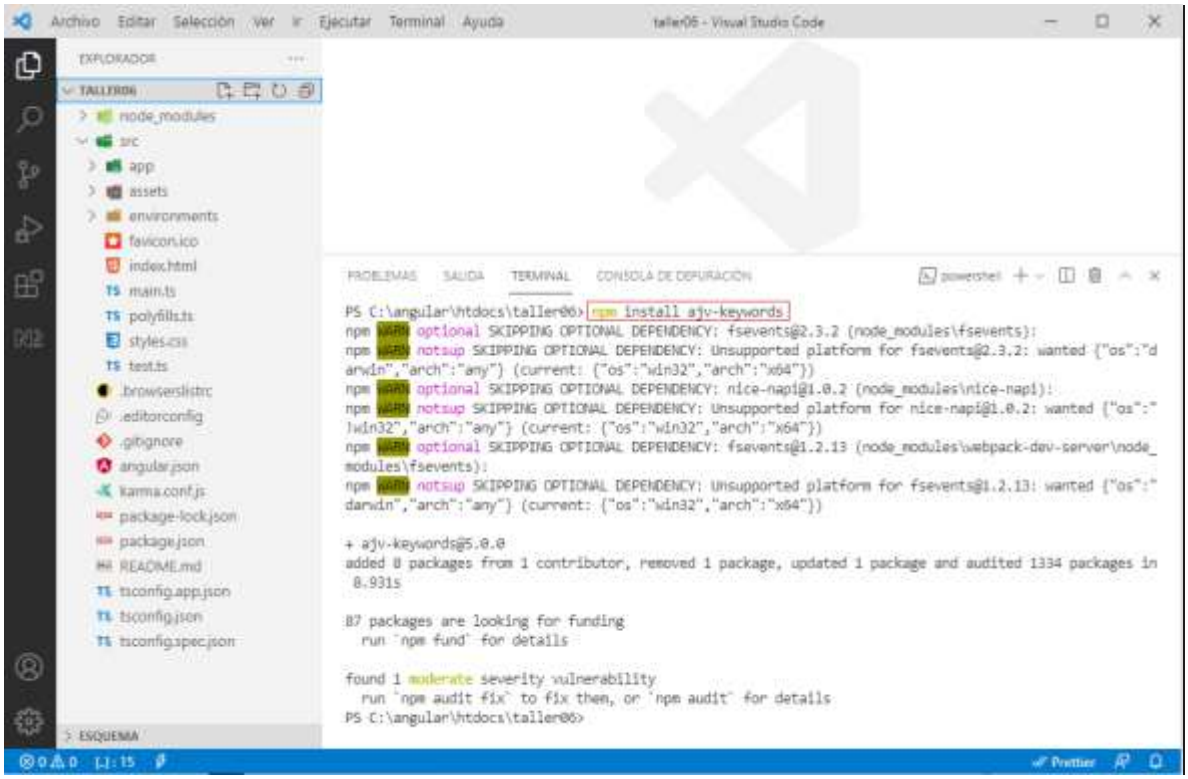
Para instalar Bootstrap, es necesario instalar varias librerías adicionales, como por ejemplo JQuery.

JQuery es una librería de JavaScript de código abierto, la cual permite una interacción más sencilla entre el desarrollador y el código que se desea manipular con JavaScript. A su vez, Jquery requiere de otras librerías adicionales para su correcto funcionamiento. A continuación, se presenta el orden de instalación de las librerías y del framework Bootstrap.

```
npm install ajv-keywords
```

La librería "***AJV***" permite implementar una lógica compleja para la validación de datos en formato JSON a través de esquemas declarativos, sin necesidad de escribir código. "***ajv-keywords***" provee claves personalizadas parea el validador "***AJV***".

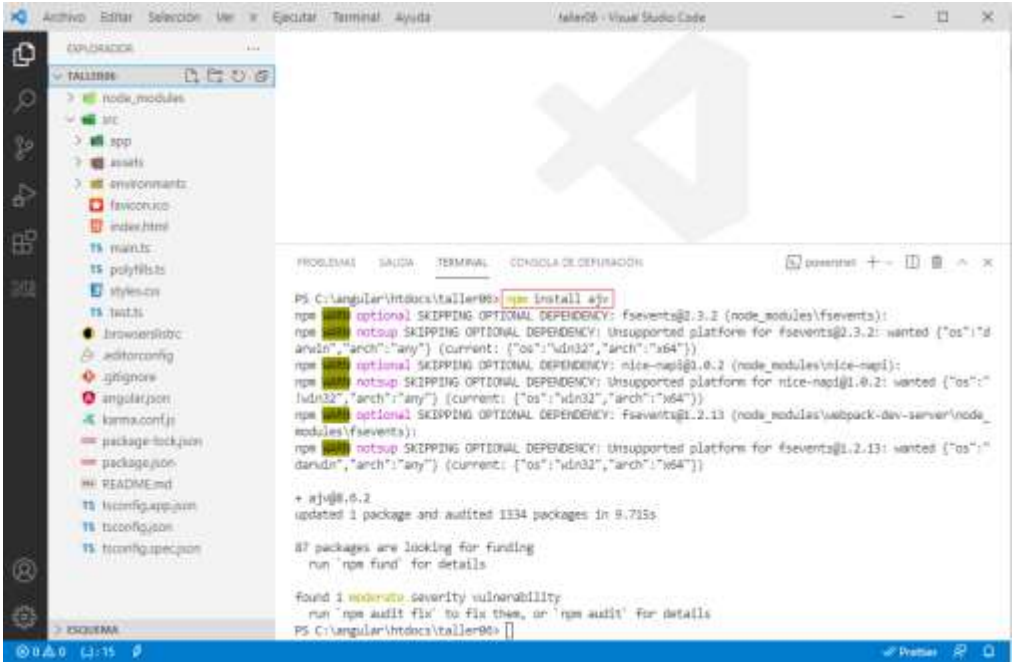
Imagen 83. Taller04, instalación de ajv-keywords



Fuente: (Autor, 2021)

```
npm install ajv
```

Imagen 84. Taller06, instalación de la librería AJV

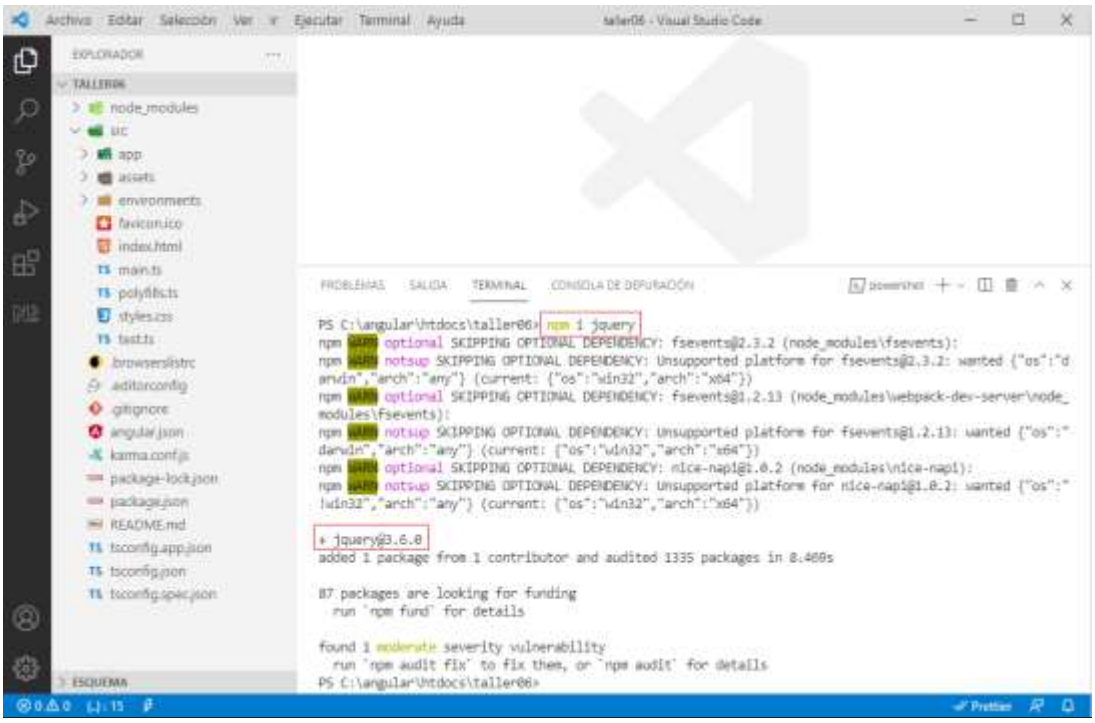


Fuente: (Autor, 2021)

El comodín "***install***" de los comandos anteriores se puede reemplazar por la letra "***i***":

```
npm i jquery
```

Imagen 85. Taller06, instalación de Jquery

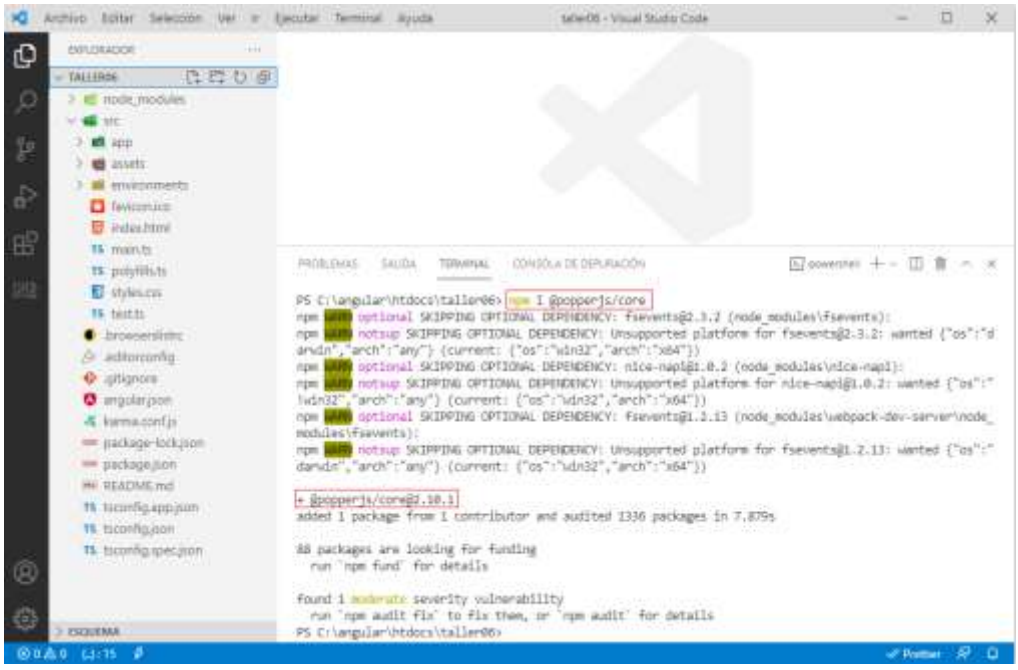


Fuente: (Autor, 2021)

```
npm i @popperjs/core
```

La "***popperJS***" es una librería que permite crear etiquetas o tooltips sobre un elemento con solo escribir una línea de código. El uso de esta librería potencia el uso de algunas opciones básicas de "***Bootstrap***".

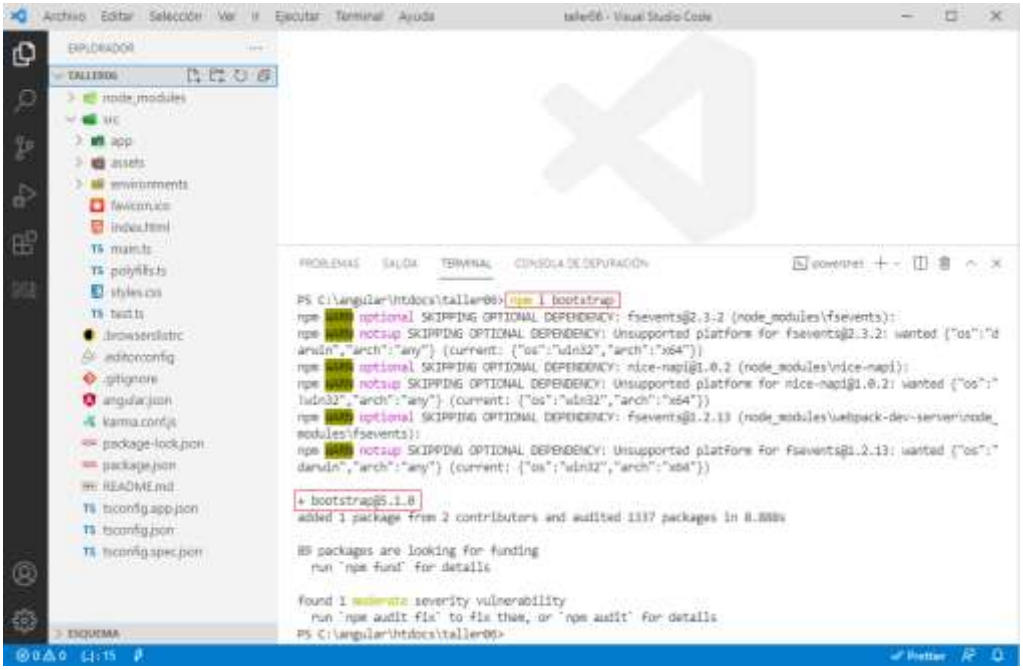
Imagen 86. Taller06, instalación de popperJS



Fuente: (Autor, 2021)

```
npm i bootstrap
```

Imagen 87. Taller06, instalación de bootstrap

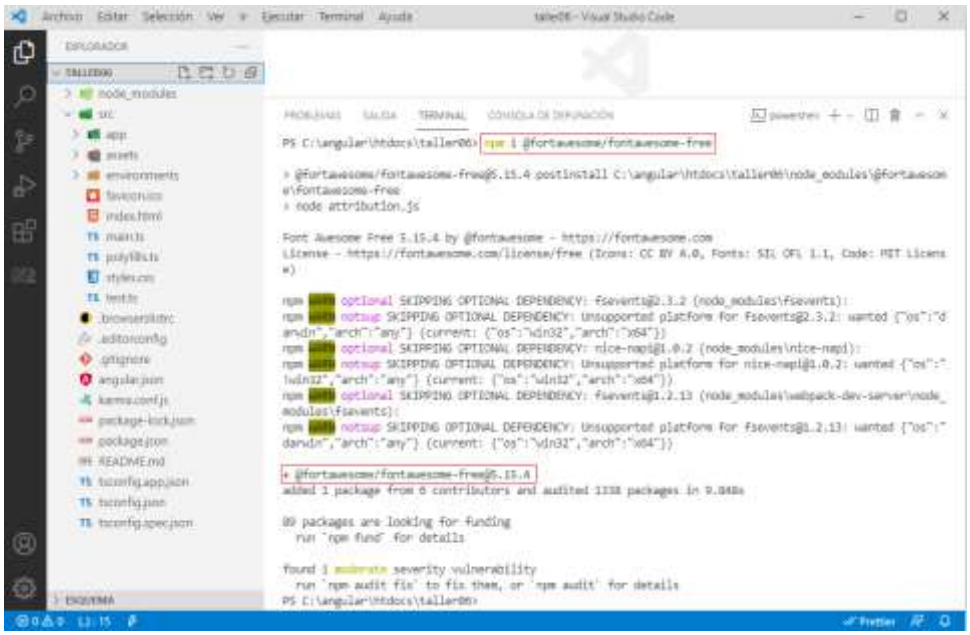


Fuente: (Autor, 2021)

Adicionalmente a las librerías ya utilizadas, en este taller se incluirá una librería para permitir el acceso a iconos de la librería fontawesome. El comando para su instalación es el siguiente:

```
npm i @fortawesome/fontawesome-free
```

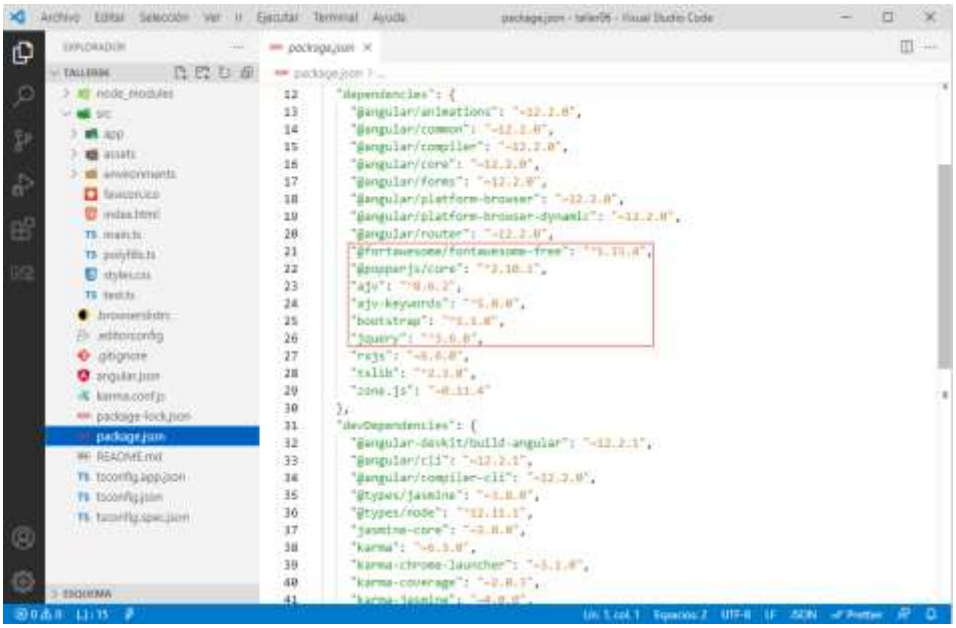
Imagen 88. Taller05, instalación de la fontawesome



Fuente: (Autor, 2021)

De manera automática, en el archivo "***package.json***" aparecerán las referencias de las librerías y frameworks instalados con los comandos anteriores. La Imagen 64 presenta parte del contenido del "***package.json***"

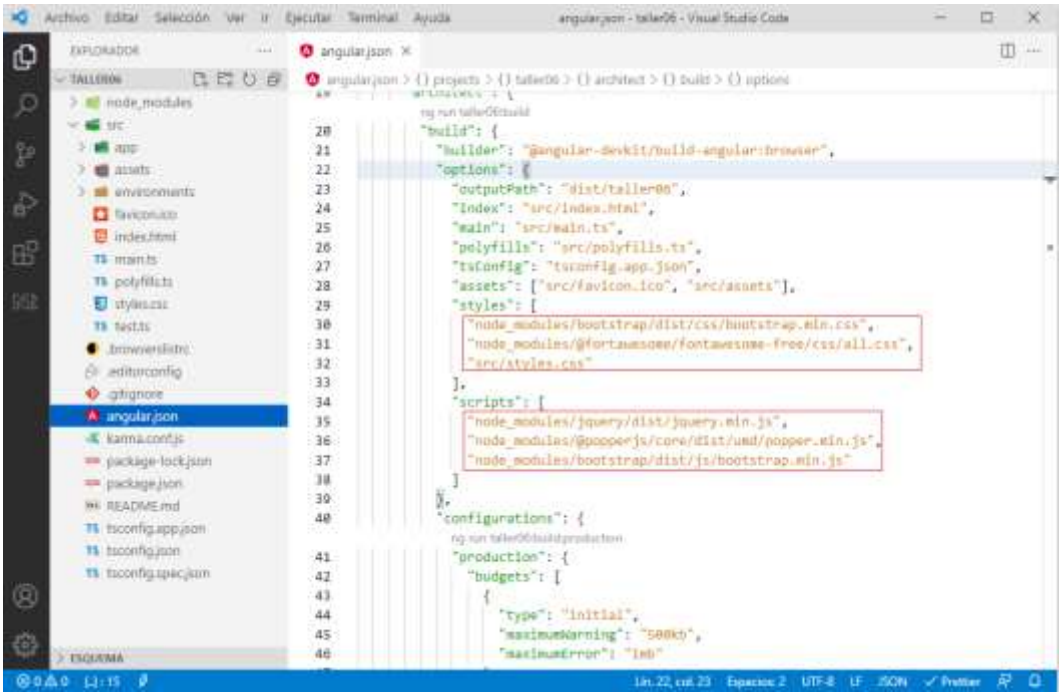
Imagen 89. Taller06, Contenido del archivo package.json



Fuente: (Autor, 2021)

Un proceso que NO es automático es la inclusión de los archivos fuentes en el proyecto en desarrollo. Se debe editar el archivo "***angular.json***" e incluir las instrucciones presentadas en la Imagen 65.

Imagen 90. Taller06, inclusión de estilos y JavaScripts en angular.json



Fuente: (Autor, 2021)

Para efectos prácticos se comparte el código para copiarlo y pegarlo en el archivo "***angular.json***"

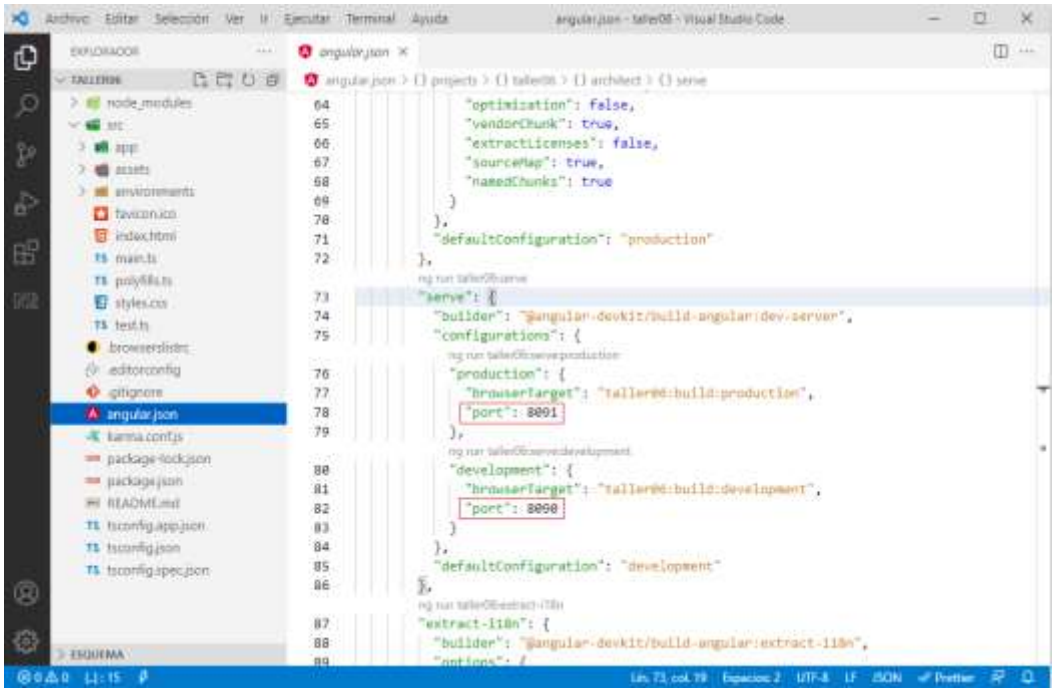
```
        "build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/taller06",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": ["src/favicon.ico", "src/assets"],
            "styles": [
              "node_modules/bootstrap/dist/css/bootstrap.min.css",
              "node_modules/@fortawesome/fontawesome-free/css/all.css",
              "src/styles.css"
            ],
            "scripts": [
              "node_modules/jquery/dist/jquery.min.js",
              "node_modules/@popperjs/core/dist/umd/popper.min.js",
              "node_modules/bootstrap/dist/js/bootstrap.min.js"
            ]
          },
```

Tabla 18. Taller06, Inclusión de estilos y JavaScripts en el archivo angular.json

7.4.1.4 Configuración del puerto del proyecto

En el mismo archivo "***angular.json***" se puede configurar el puerto por defecto en donde se ejecutará el proyecto al ser visualizado en un navegador. Basta con agregar la propiedad "***port***" y asignar un valor válido de puerto. La Imagen 91 presenta la inclusión de la propiedad que permite definir el puerto en el cual se podrá visualizar el proyecto.

Imagen 91. Taller06, configuración del puerto en el archivo angular.json



Fuente: (Autor, 2021)

En la Tabla 19 se encuentra el bloque de código que se debe agregar para configurar los puertos en modo producción y desarrollo.

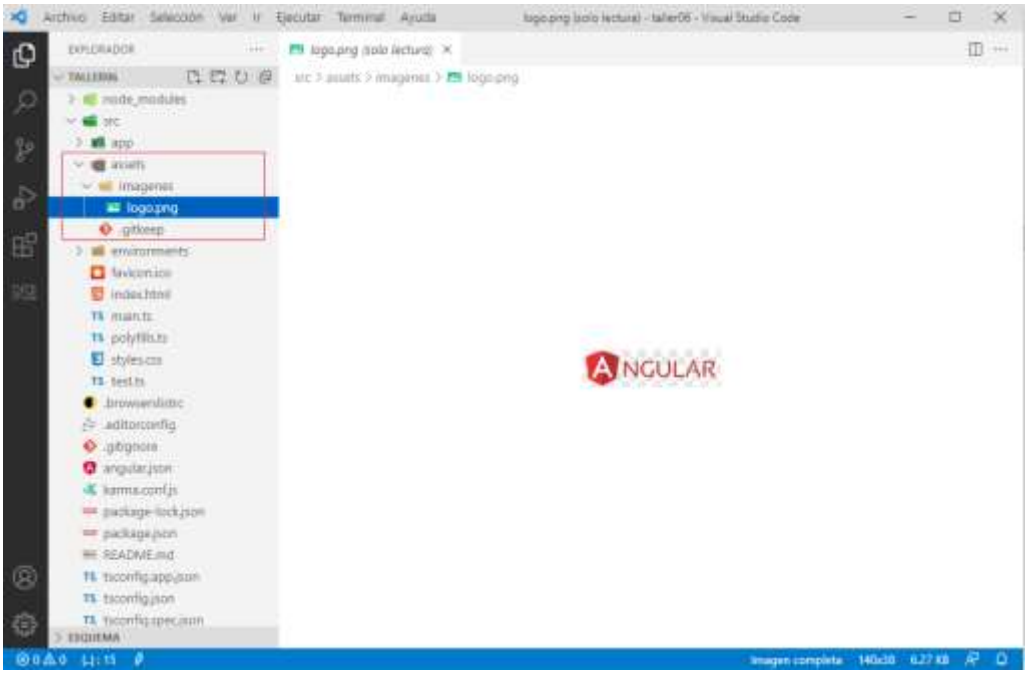
```
      "serve": {
        "builder": "@angular-devkit/build-angular:dev-server",
        "configurations": {
          "production": {
            "browserTarget": "taller06:build:production",
            "port": 8091
          },
          "development": {
            "browserTarget": "taller06:build:development",
            "port": 8090
          }
        },
        "defaultConfiguration": "development"
      },
```

Tabla 19. Taller06, configuración de puertos en angular.json

7.4.1.5 Inclusión de logos, imágenes estáticas

Los logos o imágenes permiten enriquecer visualmente la aplicación, en este taller se incluirá una imagen que se pueda acceder desde cualquier componente. Se sugiere descargar de Internet una imagen pequeña y con transparencia, que se pueda utilizar a manera de logo en la componente cabecera. La Imagen 92 presenta el sitio de almacenamiento de las imágenes estáticas dentro del taller06.

Imagen 92. Taller06, carpeta para almacena imágenes

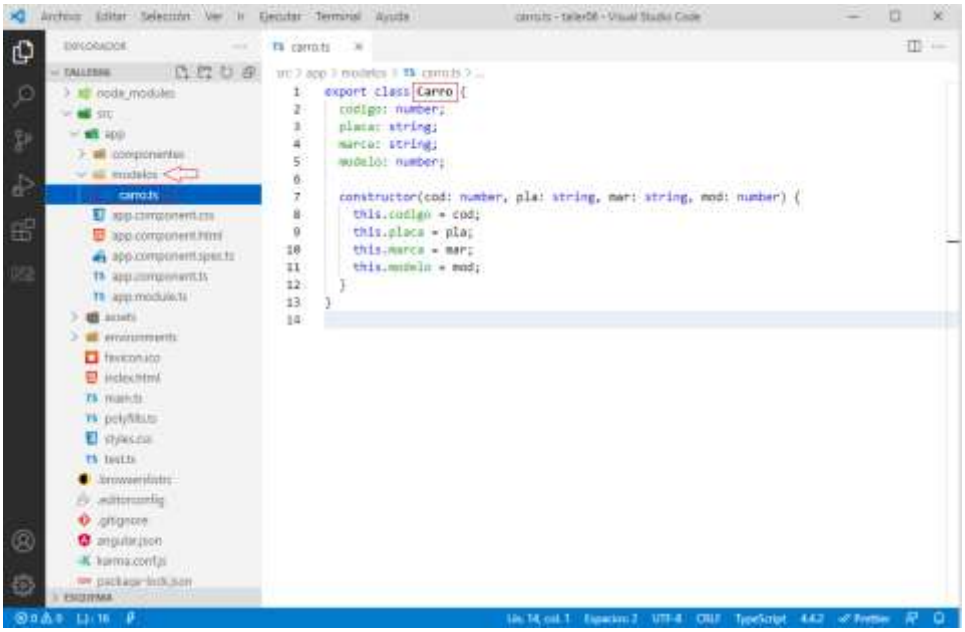


Fuente: (Autor, 2021)

7.4.1.6 Modelos

Un modelo es una clase que mapea información por medio de sus propiedades o atributos, son útiles para enviar, recibir y procesar información con un formato determinado. La Imagen 69 presenta el modelo llamado "***Carro***" que se utilizará en este taller, el cual estará almacenado en el archivo: "***src\app\modelos\carro.ts***". El lector deberá crear la carpeta "modelos" y el archivo "***carro.ts***" utilizando el clic derecho sobre la carpeta "app".

Imagen 93. Taller06, estructura de carpetas de los modelos, modelo Carro



Fuente: (Autor, 2021)

A manera de ejemplo se presenta el código del modelo carro en la Tabla 20.

```
export class Carro {
  codigo: number;
  placa: string;
  marca: string;
  modelo: number;

  constructor(cod: number, pla: string, mar: string, mod: number) {
    this.codigo = cod;
    this.placa = pla;
    this.marca = mar;
    this.modelo = mod;
  }
}
```

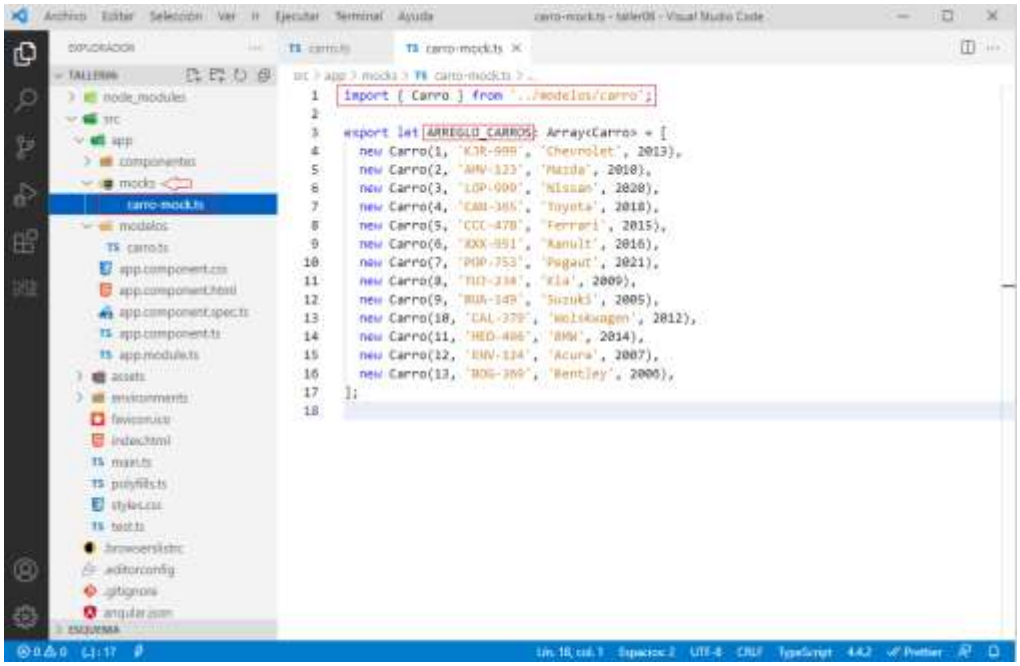
Tabla 20. Taller06, código de la clase Carro

7.4.1.7 Mocks

Los mocks son datos de prueba que permiten al desarrollador verificar la aplicación sin necesidad de tener conexión con un componente backend. La ventaja de usar mocks radica en la sencillez que existe al cambiar los datos de prueba por servicios que consumen la data desde una API externa.

El mock utilizado en este taller hará uso del modelo "***Carro***" y tendrá información de prueba para que sea desplegada en la plantilla del componente "***cuerpo***". La imagen xxx presenta la organización de carpetas para el mock y su contenido.

Imagen 94. Taller06, estructura de carpetas de los mocks, mock para Carro



Fuente: (Autor, 2021)

La variable que se utiliza para almacenar la información del mock es "ARREGLO_CARROS", inicialmente es una constante que alimentará la información de alguna plantilla, sin embargo, este arreglo puede llegar a ser modificado en tiempo de ejecución, razón por la cual se sugiere definir la variable con "***let***".

A manera de ejemplo se presenta el código del mock para carros en la Tabla 21.

```
import { Carro } from '../modelos/carro';


export let ARREGLO_CARROS: Array<Carro> = [
  new Carro(1, 'KJR-999', 'Chevrolet', 2013),
  new Carro(2, 'AMV-123', 'Mazda', 2010),
  new Carro(3, 'LOP-999', 'Nissan', 2020),
  new Carro(4, 'CAN-365', 'Toyota', 2018),
```

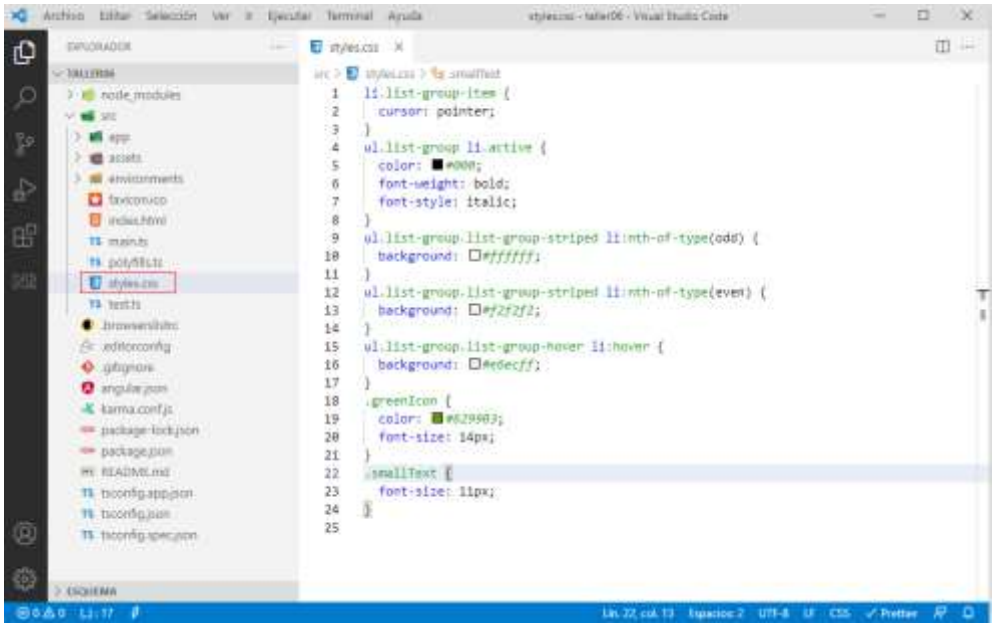
```
  new Carro(5, 'CCC-478', 'Ferrari', 2015),
  new Carro(6, 'XXX-951', 'Ranult', 2016),
  new Carro(7, 'POP-753', 'Pegaut', 2021),
  new Carro(8, 'TUJ-234', 'Kia', 2009),
  new Carro(9, 'BUA-149', 'Suzuki', 2005),
  new Carro(10, 'CAL-379', 'Wolskwagen', 2012),
  new Carro(11, 'MED-496', 'BMW', 2014),
  new Carro(12, 'ENV-124', 'Acura', 2007),
  new Carro(13, 'BOG-369', 'Bentley', 2006),
];
```

Tabla 21. Taller06, código del mock de prueba

7.4.1.8 Estilos globales

El objetivo de este taller es construir un CRUD que presente la información de una lista de carros y permita su edición, creación o eliminación. El uso de estilos globales a todo el taller ayudará a mejorar la presentación de la información, por ende, a mejorar la usabilidad de este. Dado que en este taller se está utilizando bootstrap, se ha seleccionado el componente "***List group***" para presentar la información en forma de lista. Por lo anterior, se sugiere adicionar los siguientes estilos al archivo "***src\styles.css***" para mejorar la presentación de la información. La Imagen 95 presenta los estilos sugeridos.

Imagen 95. Taller06, estilos para el componente list-group



Fuente: (Autor, 2021)

En la Tabla 22 se presenta a manera de ejemplo el contenido de los estilos sugeridos para este taller.

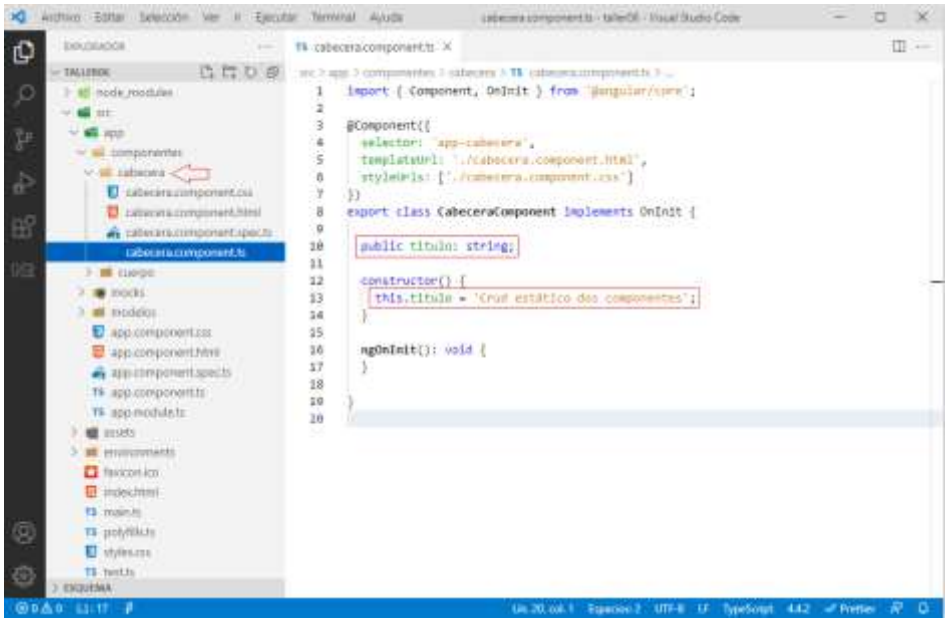
```
li.list-group-item {
  cursor: pointer;
}
ul.list-group li.active {
  color: #000;
  font-weight: bold;
  font-style: italic;
}
ul.list-group.list-group-striped li:nth-of-type(odd) {
  background: #ffffff;
}
ul.list-group.list-group-striped li:nth-of-type(even) {
  background: #f2f2f2;
}
ul.list-group.list-group-hover li:hover {
  background: #e6ecff;
}
.greenIcon {
  color: #629903;
  font-size: 14px;
}
.smallText {
  font-size: 11px;
}
```

Tabla 22. Taller06, estilos globales para las listas, íconos y tamaño de textos

7.4.1.9 Componente cabecera

En la clase del componente "***Cabecera***" se adicionará una propiedad con su respectiva inicialización en el constructor. La Imagen 96 presenta el contenido de la clase y los ajustes sugeridos.

Imagen 96. Taller06, clase componente cabecera con ajustes



Fuente: (Autor, 2021)

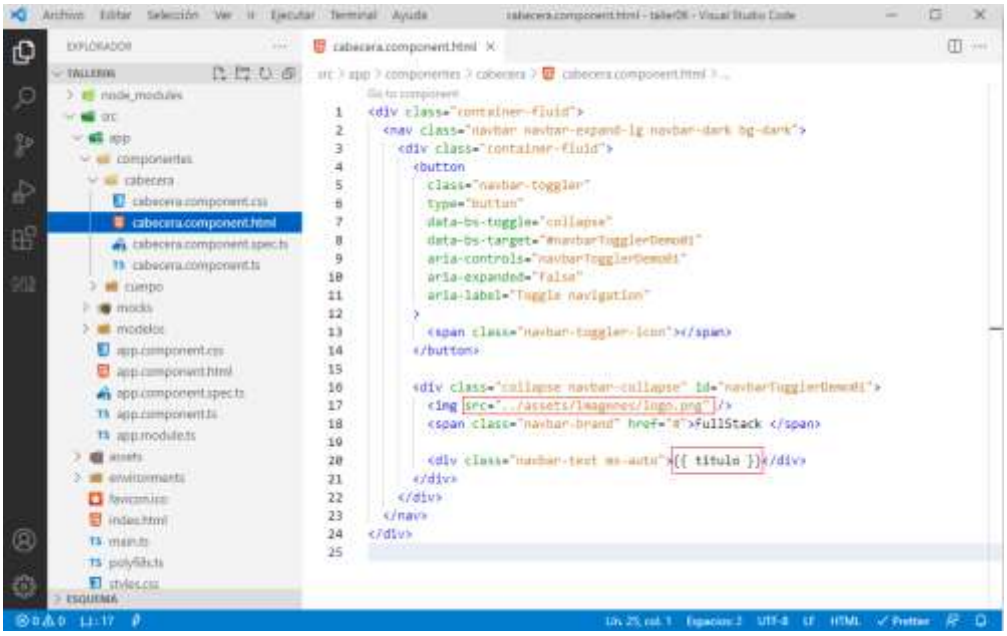
A manera de ejemplo se presenta el código de la clase cabecera en la Tabla 21.

```
import { Component, OnInit } from '@angular/core';

@Component({
  selector: 'app-cabecera',
  templateUrl: './cabecera.component.html',
  styleUrls: ['./cabecera.component.css']
})
export class CabeceraComponent implements OnInit {

  public titulo: string;

  constructor() {
    this.titulo = 'Crud estático dos componentes';
  }

  ngOnInit(): void {
  }

}
```

Tabla 23. Taller06, clase del componente cabecera

Para la plantilla del componente cabecera se utilizará un maquetado básico con base en la librería Bootstrap. La Imagen 97 presenta el maquetado sugerido para este componente.

Imagen 97. Taller06, plantilla del componente cabecera

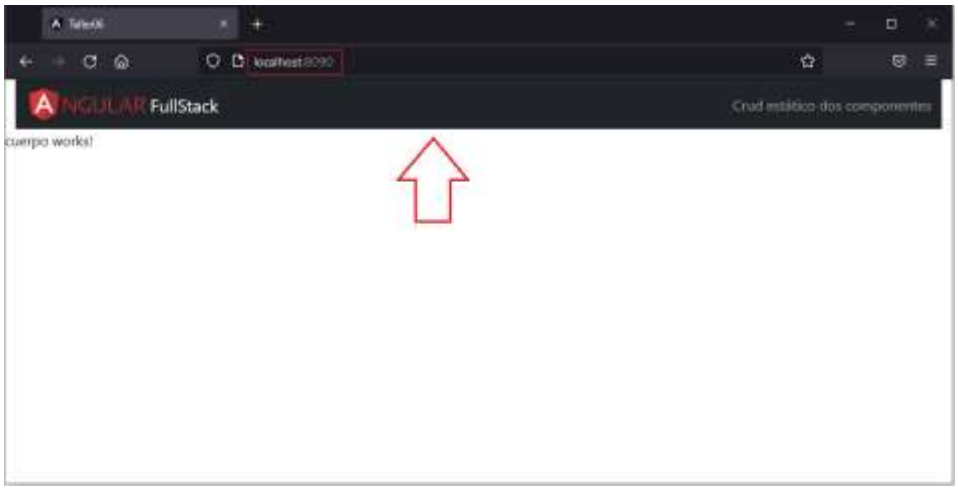


Fuente: (Autor, 2021)

7.4.1.10 Prueba parcial del taller06

Para verificar el avance del taller, se sugiere al lector que ejecute el proyecto. Desde la terminal de Visual Estudio Code se debe ejecutar el comando “ng serve”. La Imagen 98 presenta el resultado del avance del taller hasta este punto.

Imagen 98. Taller06 en ejecución

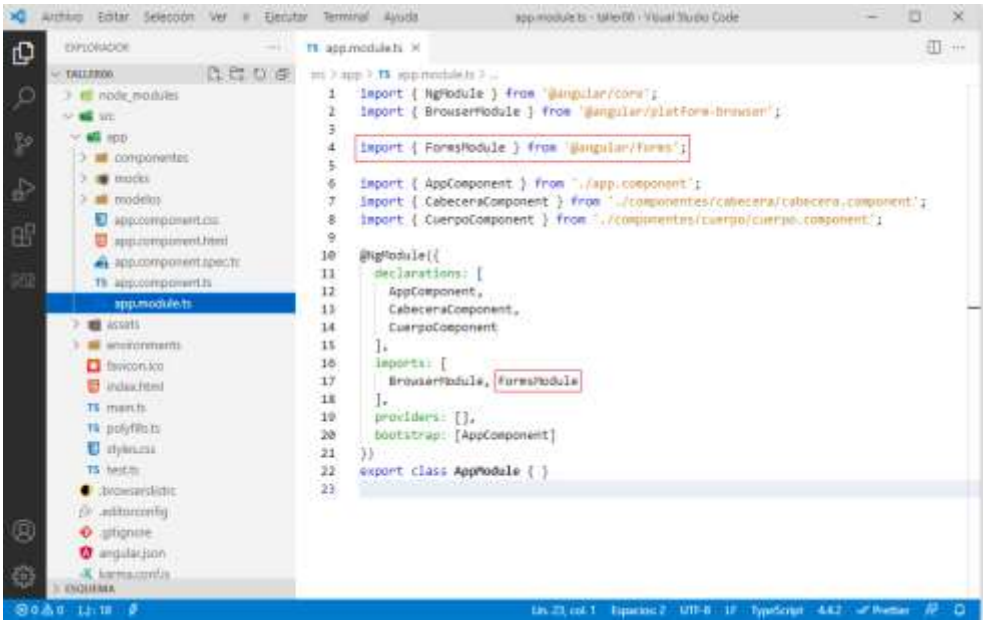


Fuente: (Autor, 2021)

7.4.1.11 Módulo @angular/forms

Para la manipulación de formularios en Angular es necesaria la inclusión de la librería “***@angular/forms***”, esta librería se debe incluir en el archivo “***src\app\app.module.ts***”. La imagen xxx presenta la inclusión de la librería.

Imagen 99. Taller06, inclusión de la librería @angular/forms

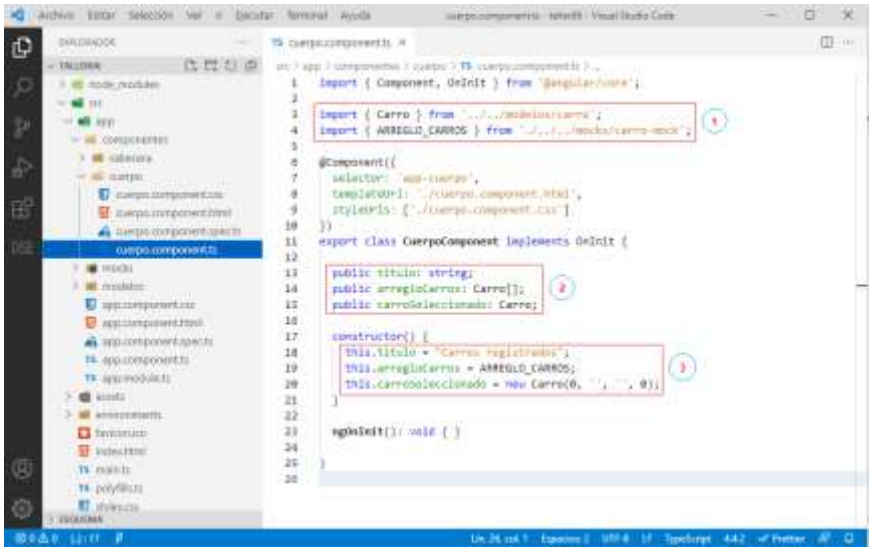


Fuente: (Autor, 2021)

7.4.1.12 Componente cuerpo

La clase del componente cuerpo requiere propiedades y métodos que permitan incluir las funcionalidades para el CRUD a implementar. La Imagen 100 presenta la inclusión del modelo y el mock a utilizar en este taller, así como la definición de las propiedades o atributos de la clase y su inicialización en el constructor.

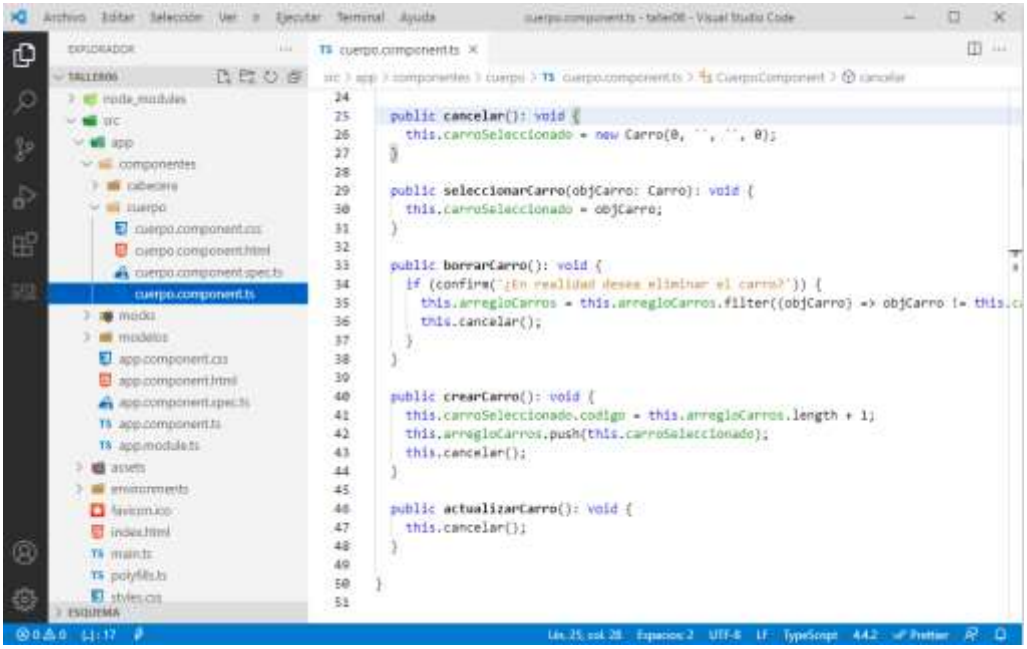
Imagen 100. Inclusión de modelos y mocks, propiedades e inicialización



Fuente: (Autor, 2021)

Dos métodos claves cuando se trabaja con Angular en el desarrollo de un CRUD son: 1) método para inicializar nuevamente el objeto que sirve para manipular información y 2) Método seleccionar el objeto sobre el cual se desean realizar las operaciones. Adicionalmente se deben crear métodos para crear, eliminar y actualizar la información. La Imagen 101 presenta la lógica de los métodos enunciados.

Imagen 101. Métodos requeridos para el CRUD



Fuente: (Autor, 2021)

En la Tabla 24 se presenta a manera de ejemplo el contenido de la clase del componente cuerpo.

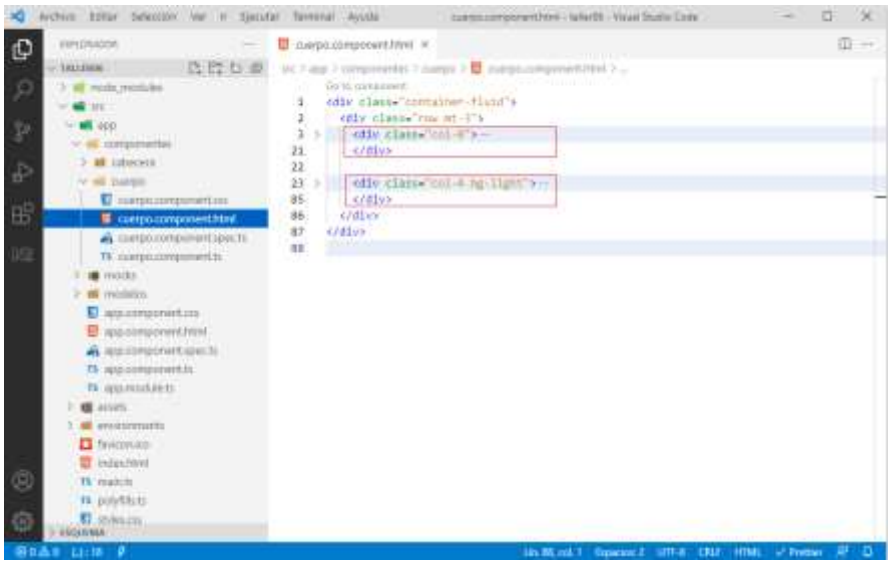
```
import { Component, OnInit } from '@angular/core';

import { Carro } from '../../modelos/carro';
import { ARREGLO_CARROS } from './../../mocks/carro-mock';

@Component({
  selector: 'app-cuerpo',
  templateUrl: './cuerpo.component.html',
  styleUrls: ['./cuerpo.component.css']
})
```

```
export class CuerpoComponent implements OnInit {

  public titulo: string;
  public arregloCarros: Carro[];
  public carroSeleccionado: Carro;

  constructor() {
    this.titulo = "Carros registrados";
    this.arregloCarros = ARREGLO_CARROS;
    this.carroSeleccionado = new Carro(0, '', '', 0);
  }

  ngOnInit(): void { }

  public cancelar(): void {
    this.carroSeleccionado = new Carro(0, '', '', 0);
  }

  public seleccionarCarro(objCarro: Carro): void {
    this.carroSeleccionado = objCarro;
  }

  public borrarCarro(): void {
    if (confirm('¿En realidad desea eliminar el carro?')) {
      this.arregloCarros = this.arregloCarros.filter((objCarro) => objCarro != this.carroSeleccio
nado);
      this.cancelar();
    }
  }

  public crearCarro(): void {
    this.carroSeleccionado.codigo = this.arregloCarros.length + 1;
    this.arregloCarros.push(this.carroSeleccionado);
    this.cancelar();
  }

  public actualizarCarro(): void {
    this.cancelar();
  }

}
```

Tabla 24. Taller06, clase del componente cuerpo

Para la plantilla del componente cuerpo se utilizará la librería bootstrap. Todo el contenido HTML estará en un bloque "***div***" utilizando la clase "***container-fluid***", en su interior se definirá un bloque que defina una fila que posteriormente tendrá dos columnas para organizar la información. Dado que el contenido de cada columna es un poco más extenso se presentará más adelante en la tabla xxx. La Imagen 102 presenta el maquetado de la plantilla del componente cuerpo.

Imagen 102. Taller06, maquetado plantilla cuerpo



Fuente: (Autor, 2021)

En la Tabla 25 se presenta a manera de ejemplo el contenido de la plantilla del componente cuerpo.

```
<div class="container-fluid">
  <div class="row mt-3">
    <div class="col-8">
      <ul class="list-group list-group-hover list-group-striped manito">
        <li class="list-group-item text-success" (click)="cancelar()">
          {{ titulo }}
        </li>

        <li class="list-group-item list-group-item-action py-1" *ngFor="let miCarro of arregloCarros"
          (click)="seleccionarCarro(miCarro)" [class.active]="miCarro == carroSeleccionado">
          <i class="fas fa-check greenIcon"></i> {{ miCarro.marca }},
          <span class="textoSmall">{{ miCarro.placa }}, {{ miCarro.modelo }}</span>
        </li>
      </ul>
```

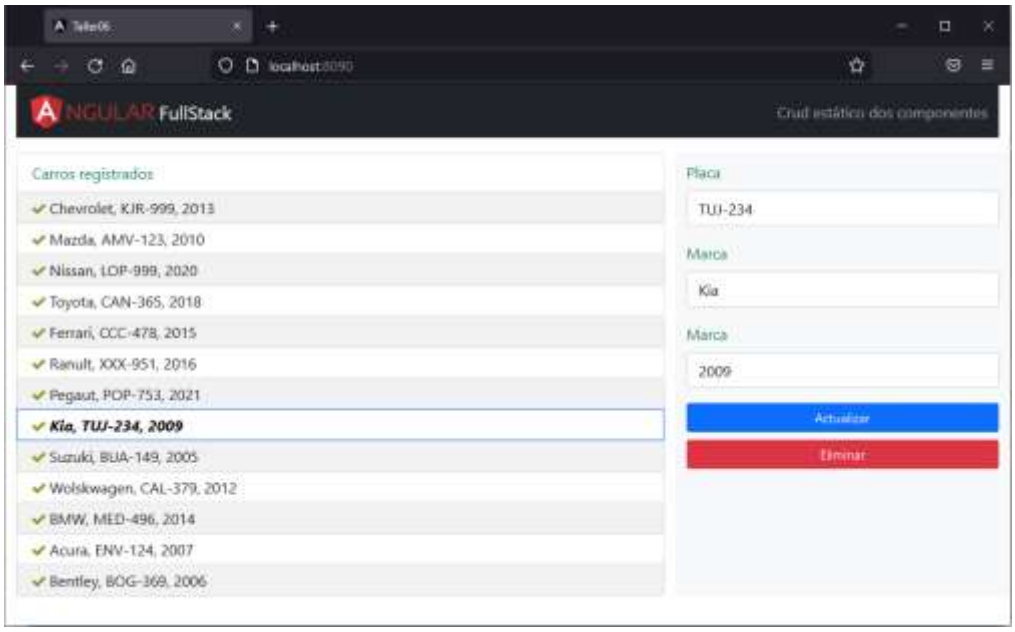
```
    </div>

    <div class="col-4 bg-light">
      <form class="mt-2">
        <div class="mb-3">
          <label for="placa" class="form-label text-success">Placa</label>
          <input type="text" class="form-control" id="placa" name="placa" [(ngModel)]="carroSeleccionado.placa" />
        </div>
        <div class="mb-3">
          <label for="placa" class="form-label text-success">Marca</label>
          <input type="text" class="form-control" id="marca" name="marca" [(ngModel)]="carroSeleccionado.marca" />
        </div>
        <div class="mb-3">
          <label for="placa" class="form-label text-success">Marca</label>
          <input type="text" class="form-control" id="modelo" name="modelo" [(ngModel)]="carroSeleccionado.modelo" />
        </div>

        <div class="d-grid gap-2">
          <button class="btn btn-success btn-sm" type="button" *ngIf="carroSeleccionado.codigo == 0"
            (click)="crearCarro()">
            Crear
          </button>

          <button class="btn btn-primary btn-sm" type="button" *ngIf="carroSeleccionado.codigo != 0"
            (click)="actualizarCarro()">
            Actualizar
          </button>

          <button class="btn btn-danger btn-sm" type="button" *ngIf="carroSeleccionado.codigo != 0"
            (click)="borrarCarro()">
            Eliminar
          </button>
        </div>
      </form>
    </div>
  </div>
</div>
```

Tabla 25. Taller06, plantilla completa del componente cuerpo

7.4.1.13 Prueba final del taller06

Para verificar el funcionamiento del taller, se sugiere al lector que ejecute el proyecto. Desde la terminal de Visual Estudio Code se debe ejecutar el comando "***ng serve***". La Imagen 103 presenta el resultado final del taller06.

Imagen 103. Taller06 en ejecución



Fuente: (Autor, 2021)

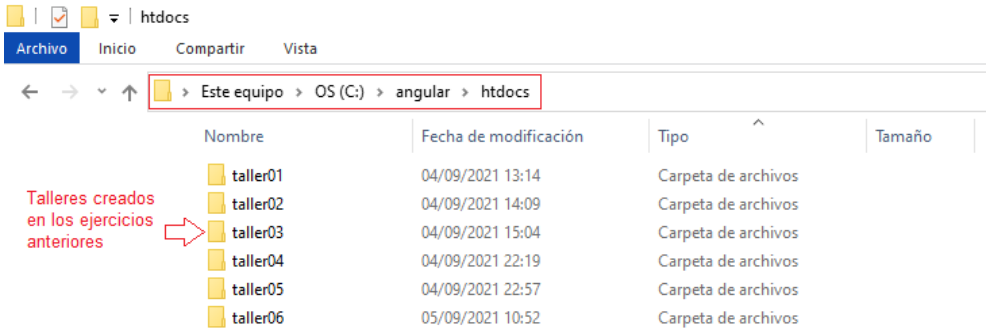
7.5. Guía 5

7.5.1. Taller07, CRUD con imágenes

7.5.1.1 Carpeta para el almacenamiento de los proyectos

Como se presenta en la Imagen 36, los talleres se crean en la carpeta: "***c:\angular\htdocs***", actualmente la carpeta contiene los talleres creados con anterioridad.

Imagen 104. Ruta para almacenar el taller07

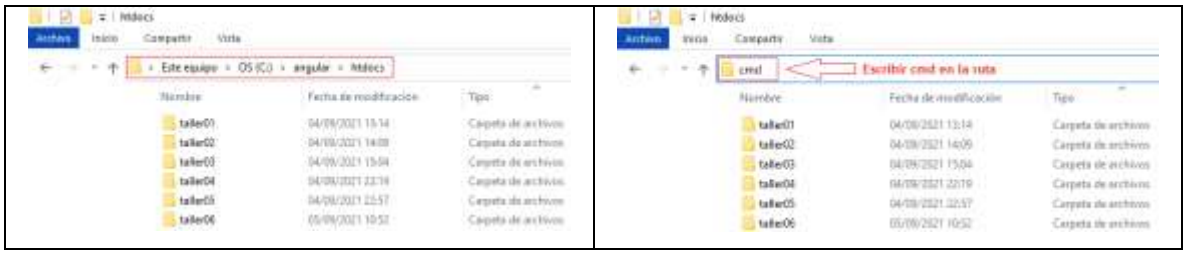


Fuente: (Autor, 2021)

7.5.1.2 Uso de la terminal de comandos

Para crear el proyecto se sugiere el uso de la terminal o consola de comandos de Windows (cmd). Este proceso de puede realizar de varias formas, en este taller se sugiere abrir el explorado de archivos y buscar la "***c:\angular\htdocs***". En la parte superior, en donde aparece la ruta se sugiere escribir directamente el comando "***cmd***", con esto, se abrirá una consola o terminal directamente en la carpeta "***c:\angular\htdocs***". La Imagen 37 presenta la alternativa para abrir la terminal de manera sencilla.

Imagen 105. Apertura de la terminal desde el explorador de archivos



Fuente: (Autor, 2021)

Idea clave

En estos primeros ejercicios no es necesario habilitar las rutas en la creación del proyecto, además, es recomendable seleccionar el estilo *CSS* al momento de crear el proyecto

Una vez en la carpeta “c:\angular\htdocs” se debe ejecutar el comando:

```
C:\angular\htdocs> ng new taller07
```

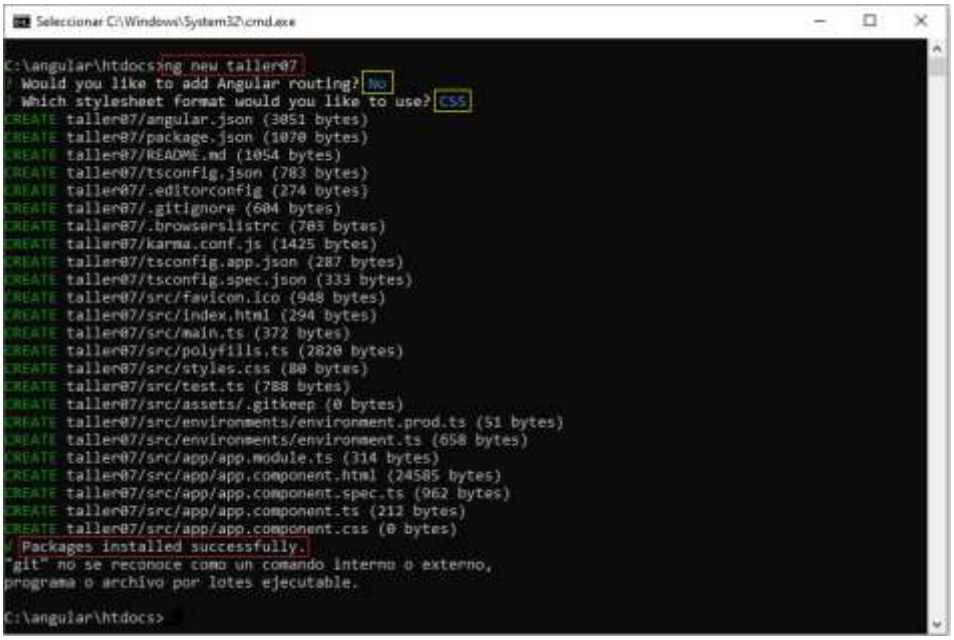
El comando anterior creará la estructura básica de los archivos, conocida como “***Scaffolding***”. La Imagen 38 presenta la ejecución del comando anterior. Recuerde que al ejecutar el comando “***ng new taller07***” aparecerán dos preguntas:

1) ¿Desea utilizar rutas?, por ser un primer ejercicio se sugiere N

2) ¿Qué tipo de hoja de estilo se desea utilizar?, se sugiere CSS

con [Enter] iniciará el proceso de creación del proyecto. El proceso consiste en la creación de una estructura básica de proyecto la cual es descargada de internet.

Imagen 106. Resultado al crear el proyecto taller07



Fuente: (Autor, 2021)

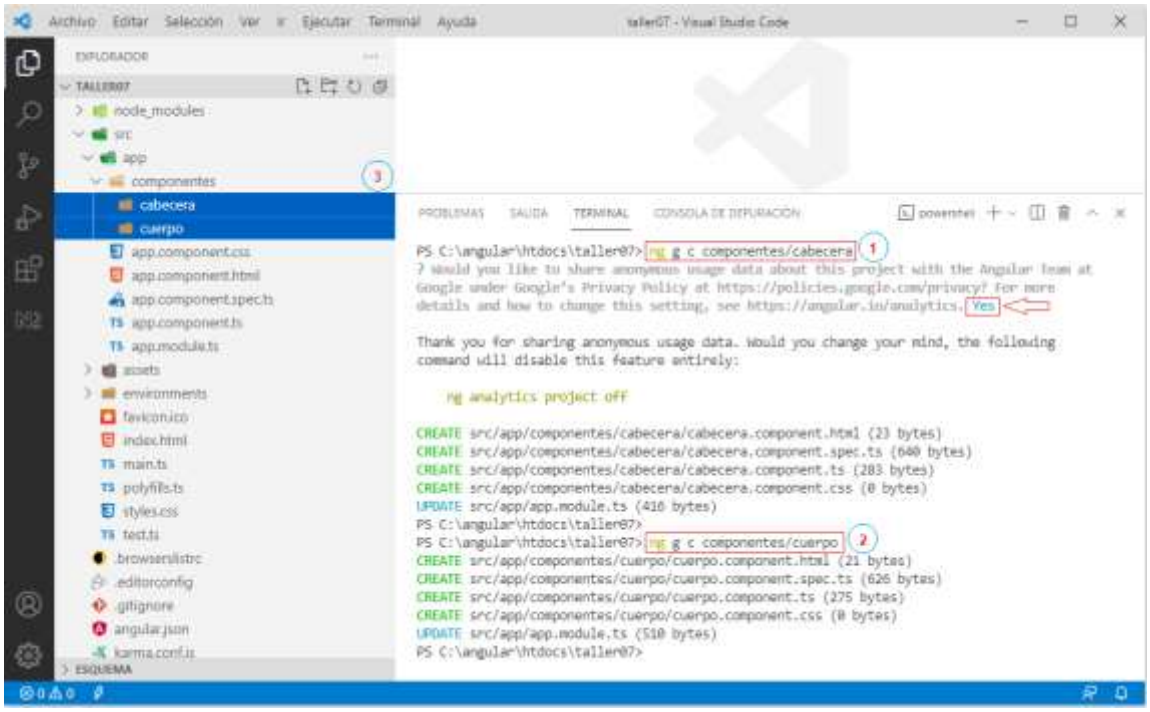
7.5.1.3 Creación de componentes

Es recomendable empezar a trabajar los proyectos con el editor Visual Studio Code (VSC), de esta manera se podrá utilizar la terminar para ejecutar los comandos. Para este taller, se van a crear dos componentes, para esto ejecutamos las siguientes instrucciones en la terminal de VSC:

```
ng g c componentes/cabecera
ng g c componentes/cuerpo
```

La Imagen 107 presenta la carpeta de proyecto cargada en VSC y la ejecución de las instrucciones en la terminal.

Imagen 107. Creación de componentes en la terminal



Fuente: (Autor, 2021)

En el archivo "***src\app\app.component.html***" aparece el código HTML por defecto al crear el componente, todo este código se debe cambiar por el código presentado en la Tabla 7.

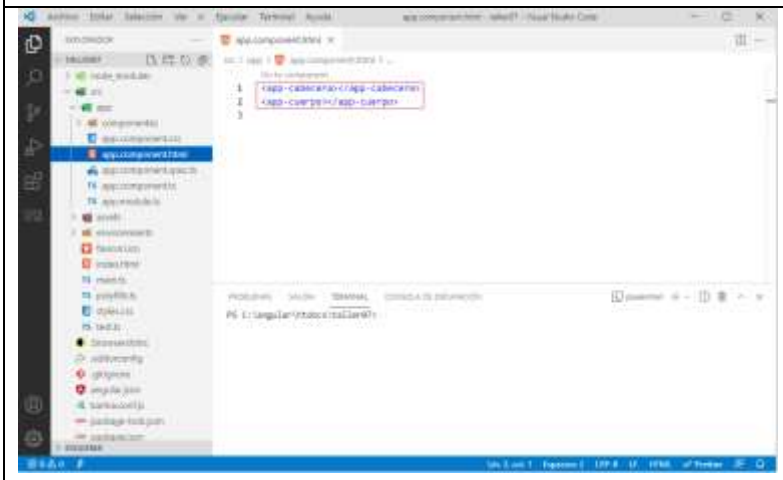
Imagen del taller	Código reutilizable
	`<app-cabecera></app-cabecera>` `<app-cuerpo></app-cuerpo>`

Tabla 26. Taller07, selectores de la plantilla principal

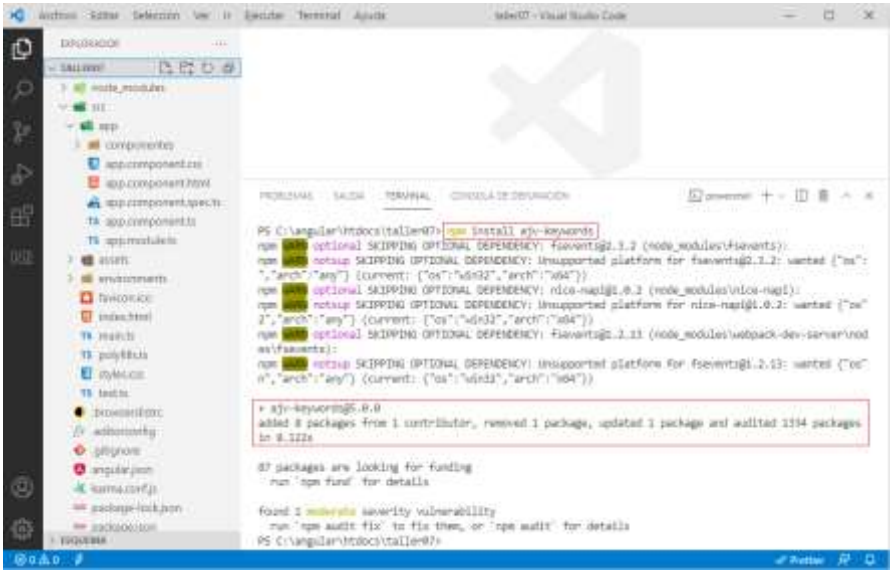
Para organizar un poco la presentación de la interfaz Web y no realizar estilos desde cero, en este taller se utilizará Bootstrap. Bootstrap es un framework de hojas de estilo que se utiliza en el desarrollo de aplicaciones frontend. Con el uso de este tipo de frameworks, el desarrollo frontend avanza con mayor velocidad.

Para instalar Bootstrap, es necesario instalar varias librerías adicionales, como por ejemplo JQuery.
JQuery es una librería de JavaScript de código abierto, la cual permite una interacción más sencilla entre el desarrollador y el código que se desea manipular con JavaScript. A su vez, Jquery requiere de otras librerías adicionales para su correcto funcionamiento. A continuación, se presenta el orden de como la instalación.

```
npm install ajv-keywords
```

La librería "***AJV***" permite implementar una lógica compleja para la validación de datos en formato JSON a través de esquemas declarativos, sin necesidad de escribir código. "***ajv-keywords***" provee claves personalizadas parea el validador "***AJV***".

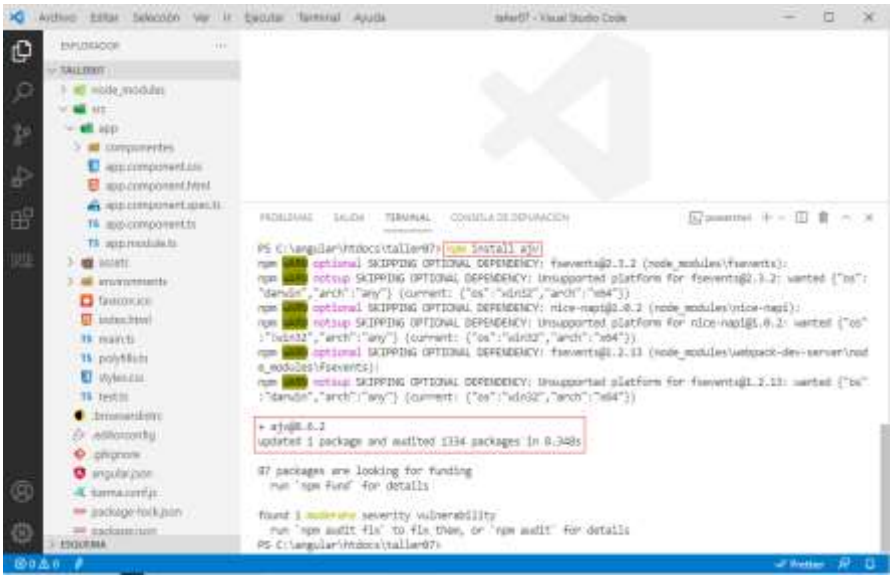
Imagen 108. Taller07, instalación de ajv-keywords



Fuente: (Autor, 2021)

```
npm install ajv
```

Imagen 109. Taller07, instalación de la librería AJV

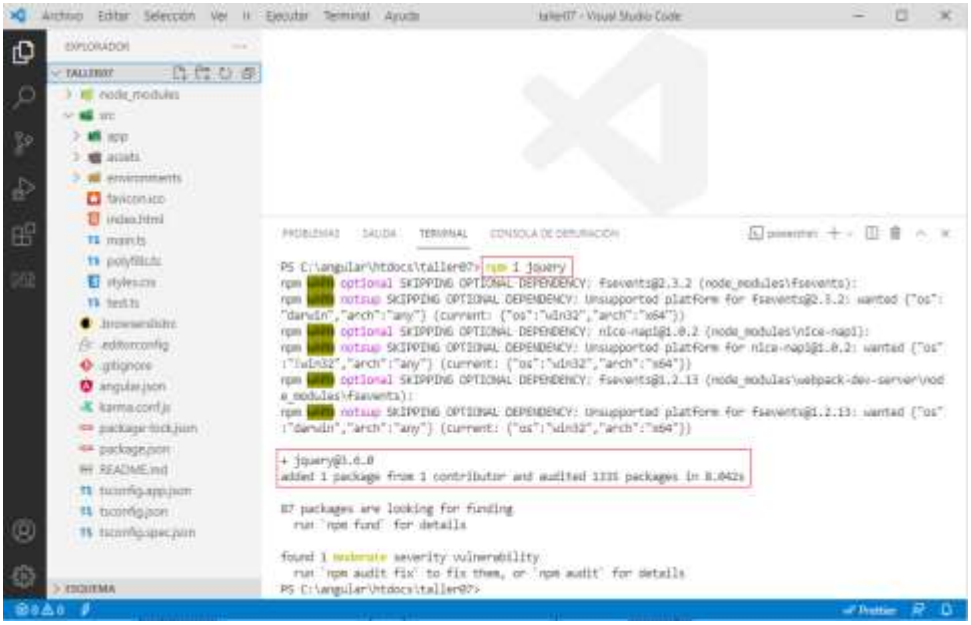


Fuente: (Autor, 2021)

El comodín "***install***" de los comandos anteriores se puede reemplazar por la letra "***i***":

```
npm i jquery
```

Imagen 110. Taller07, instalación de Jquery

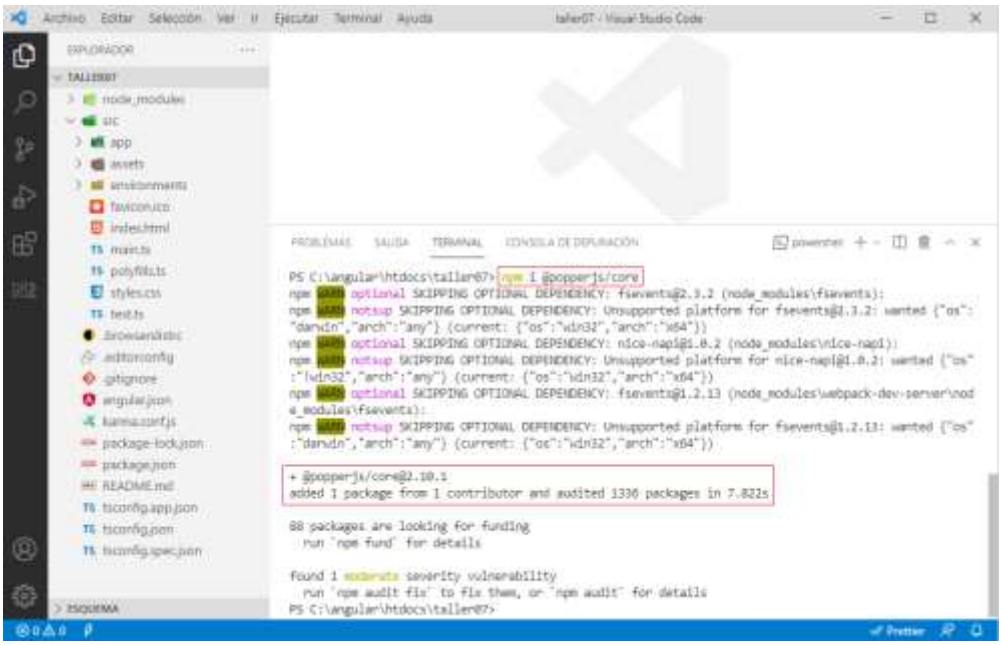


Fuente: (Autor, 2021)

```
npm i @popperjs/core
```

La "***popperJS***" es una librería que permite crear etiquetas o tooltips sobre un elemento con solo escribir una línea de código. El uso de esta librería potencia el uso de algunas opciones básicas de "***Bootstrap***".

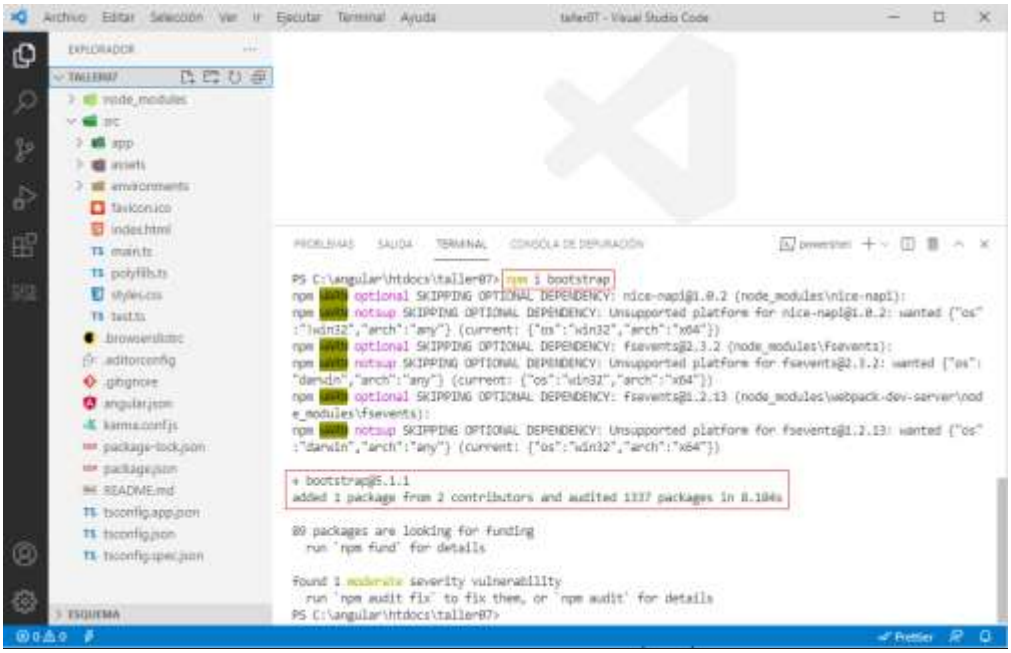
Imagen 111. Taller07, instalación de popperJS



Fuente: (Autor, 2021)

```
npm i bootstrap
```

Imagen 112. Taller07, instalación de bootstrap

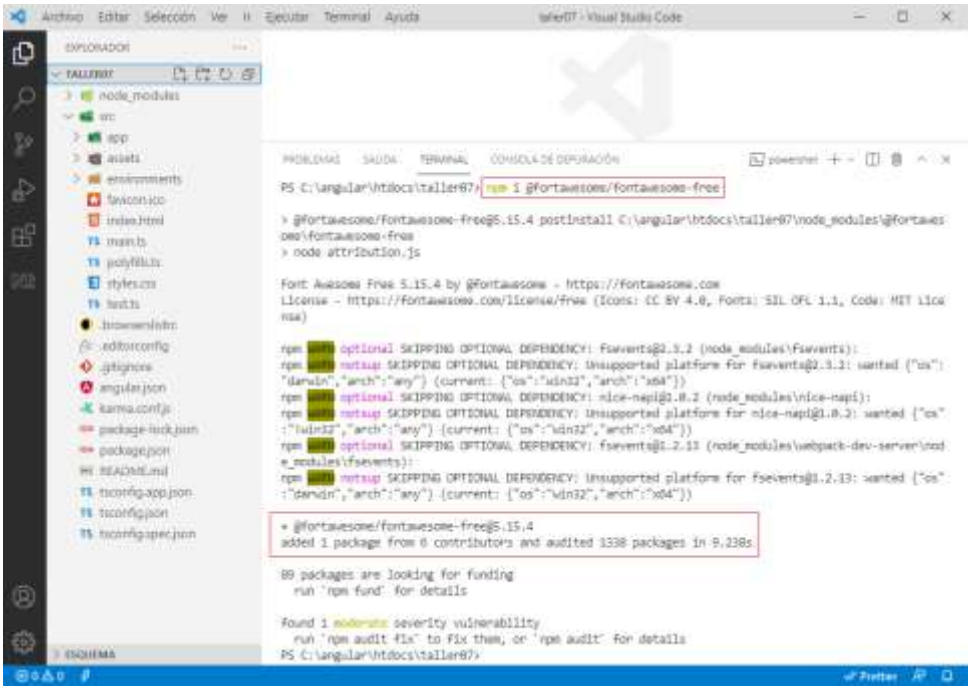


Fuente: (Autor, 2021)

En este taller se incluirá una librería para permitir el acceso a iconos de la fontawesome. El comando para su instalación es el siguiente:

```
npm i @fortawesome/fontawesome-free
```

Imagen 113. Taller07, instalación de la fontawesome

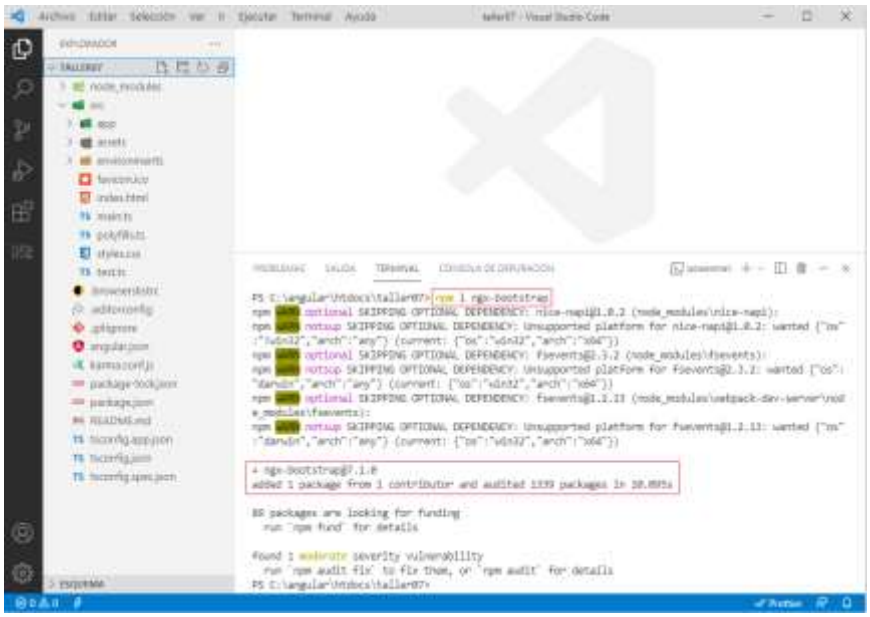


Fuente: (Autor, 2021)

Adicionalmente a las librerías ya utilizadas, en este taller se incluirán dos librerías del tipo "***ngx***", esto permite enriquecer el desarrollo y simplificar el uso de componentes. La primera librería por instalar es la "***ngx-bootstrap***", un proyecto independiente a Bootstrap que busca popularizar el desarrollo en angular de Bootstrap.

```
npm i ngx-bootstrap
```

Imagen 114. Taller07, instalación de la ngx-boostrap

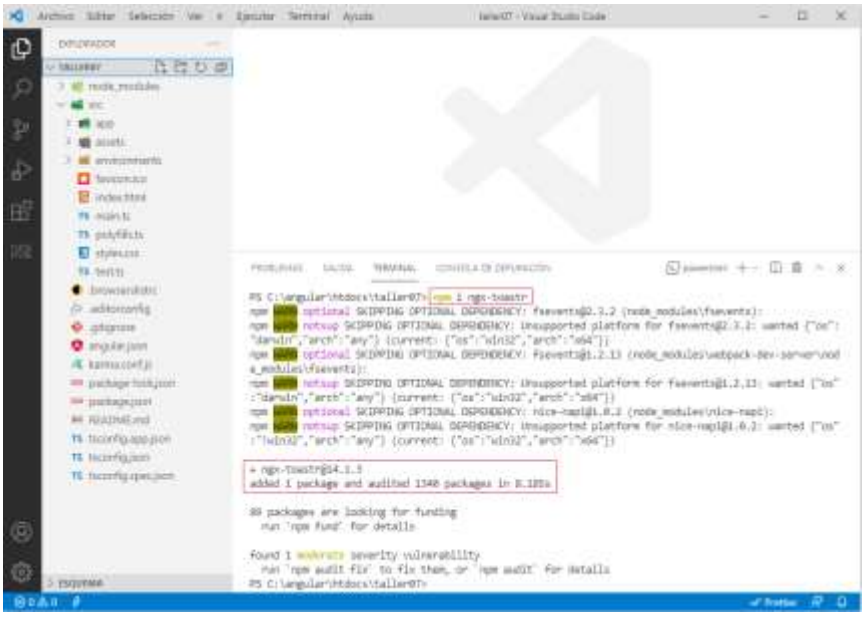


Fuente: (Autor, 2021)

la librería "***ngx-toastr***" una librería con componentes utilizado para el manejo de las alertas, el comando para su instalación es el siguiente:

```
npm i ngx-toastr
```

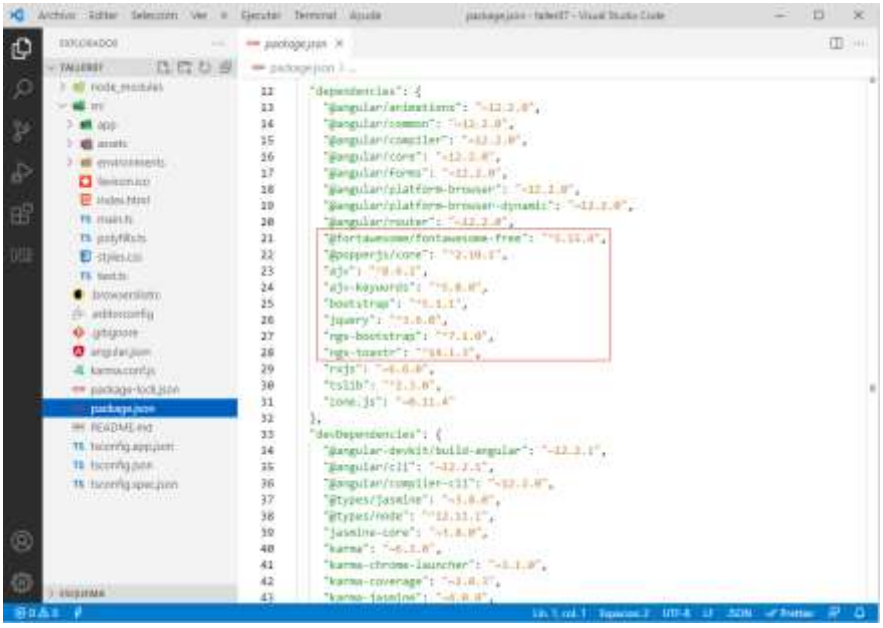
Imagen 115. Taller07, instalación de la ngx-toastr



Fuente: (Autor, 2021)

De manera automática, en el archivo "***package.json***" aparecerán las referencias de las librerías y frameworks instalados con los comandos anteriores. La Imagen 64 presenta parte del contenido del "***package.json***"

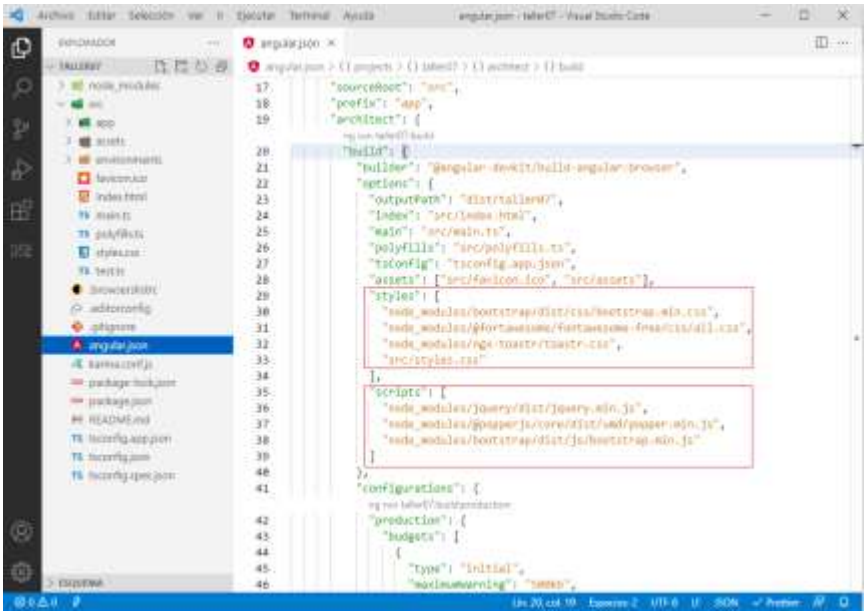
Imagen 116. Contenido del archivo package.json



Fuente: (Autor, 2021)

Un proceso que NO es automático es la inclusión de los archivos fuentes en el proyecto en desarrollo. Se debe editar el archivo "***angular.json***" e incluir las instrucciones presentadas en la Imagen 65.

Imagen 117. Contenido del archivo angular.json



Fuente: (Autor, 2021)

Para efectos prácticos se comparte el código para copiarlo y pegarlo en el archivo "***angular.json***"

```
"build": {
          "builder": "@angular-devkit/build-angular:browser",
          "options": {
            "outputPath": "dist/taller006",
            "index": "src/index.html",
            "main": "src/main.ts",
            "polyfills": "src/polyfills.ts",
            "tsConfig": "tsconfig.app.json",
            "assets": ["src/favicon.ico", "src/assets"],
            "styles": [
              "node_modules/bootstrap/dist/css/bootstrap.min.css",
              "node_modules/@fortawesome/fontawesome-free/css/all.css",
```

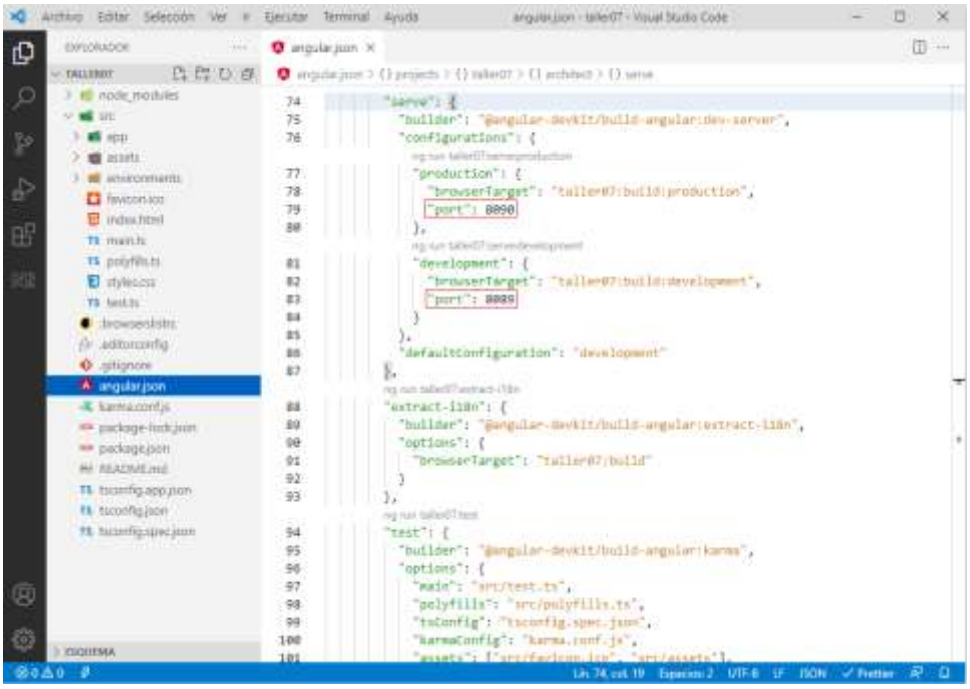
```
            "node_modules/ngx-toastr/toastr.css",
            "src/styles.css"
          ],
          "scripts": [
            "node_modules/jquery/dist/jquery.min.js",
            "node_modules/@popperjs/core/dist/umd/popper.min.js",
            "node_modules/bootstrap/dist/js/bootstrap.min.js"
          ]
        }
```

Tabla 27. Taller07, Inclusión de estilos y JavaScripts en el archivo angular.json

7.5.1.4 Asignación de puerto

Para asignar el puerto vamos a ir al archivo "***angular.json***" y en la sección servidor vamos a agregar dos líneas Imagen 118 presenta el cambio de puerto se debe que tener encuenta que los puertos no se puede .

Imagen 118. Cambio de puerto en el archivo angular.json

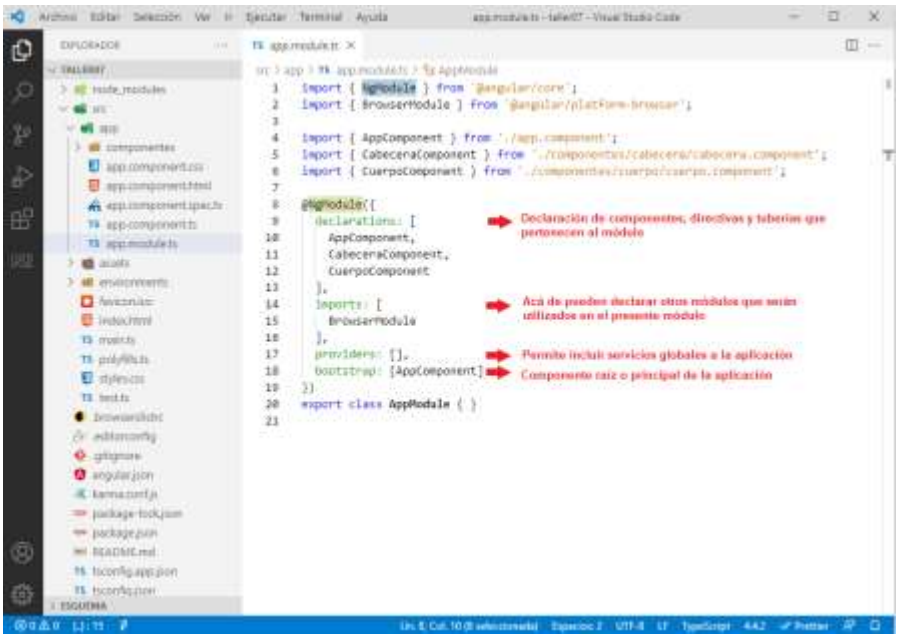


Fuente: (Autor, 2021)

7.5.1.5 Inclusión de paquetes en el módulo principal

Para utilizar las funcionalidades de algunos paquetes instalados se necesitará importar los paquetes en el módulo principal "***src\app\app.module.ts***"

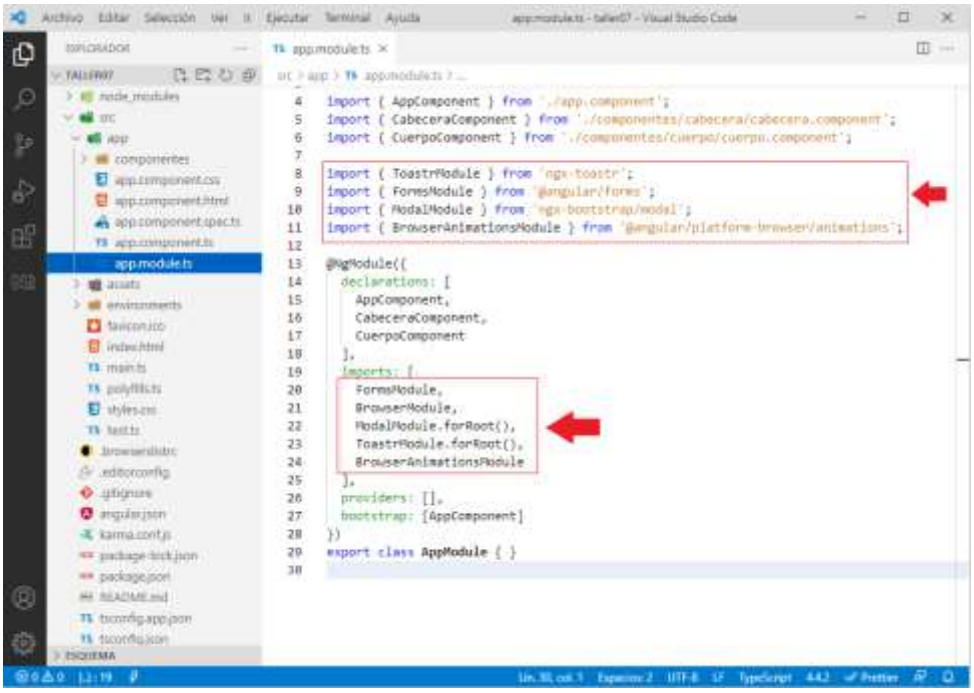
Imagen 119. Estructura modulo principal



Fuente: (Autor, 2021)

En el módulo principal "***src\app\app.module.ts***" se incluirán los módulos y componentes que harán parte de la funcionalidad para el taller07, Imagen 120 presenta la inclusión de los módulos instalados.

Imagen 120. Importación de módulos en el src\app\app.module.ts



Fuente: (Autor, 2021)

Para efectos prácticos se comparte el código para copiarlo y pegarlo en el archivo "***src\app\app.module.ts***"

```
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppComponent } from './app.component';
import { CabeceraComponent } from './componentes/cabecera/cabecera.component';
import { CuerpoComponent } from './componentes/cuerpo/cuerpo.component';

import { ToastrModule } from 'ngx-toastr';
import { FormsModule } from '@angular/forms';
import { ModalModule } from 'ngx-bootstrap/modal';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';

@NgModule({
  declarations: [
    AppComponent,
    CabeceraComponent,
    CuerpoComponent
  ],
  imports: [
    FormsModule,
```

```
    BrowserModule,
    ModalModule.forRoot(),
    ToastrModule.forRoot(),
    BrowserAnimationsModule
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Tabla 28. Inclusión de módulos en el archivo src\app\app.module.ts

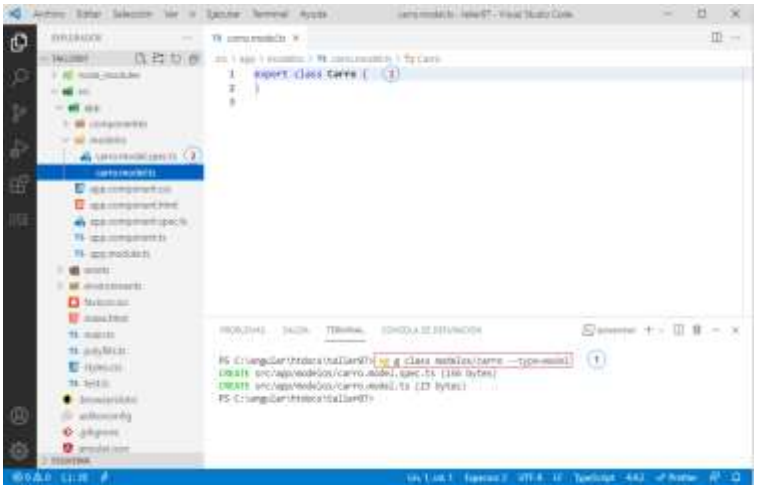
7.5.1.6 Modelos

Un modelo es la representación de un objeto por medio de atributos o propiedades. Para crear un modelo se puede utilizar los comandos de "***Angular CLI***" o se pueden crear de maneral manual. A continuación, se presenta el comando a ejecutar desde la terminal de Visual Estudio Code.

```
ng g class modelos/carro --type=model
```

La Imagen 121 presenta la terminal con el comando ejecutado, la estructura de la carpeta modelos y la clase por defecto generada con "***Angular CLI***".

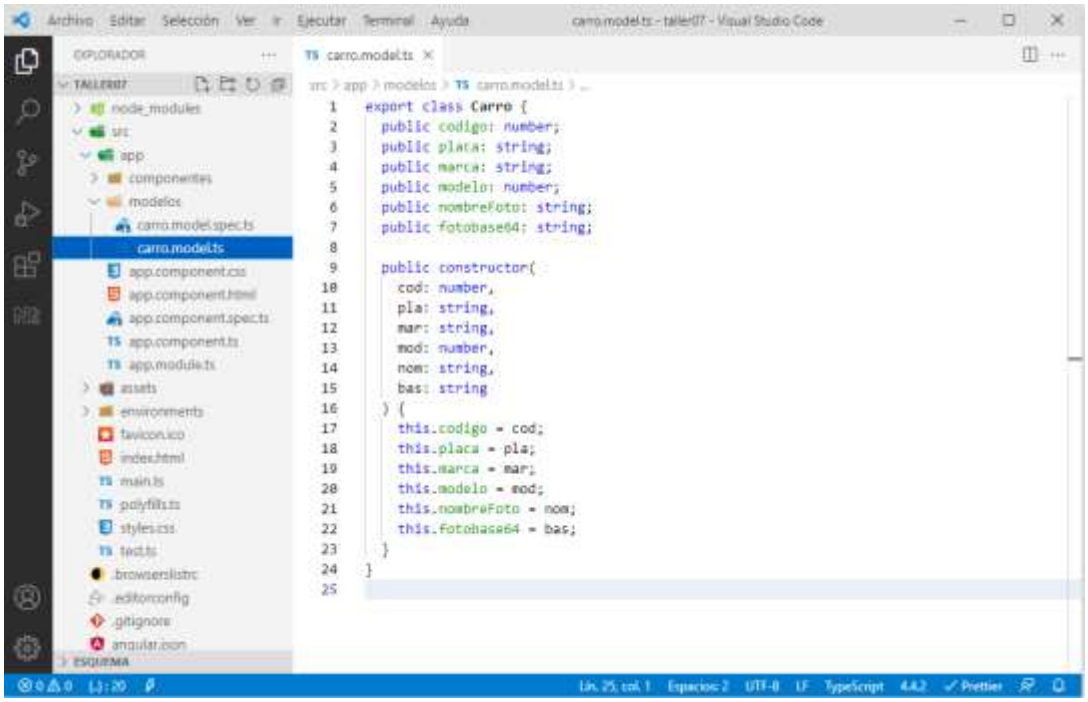
Imagen 121. Carpeta modelo y archivo carro.ts



Fuente: (Autor, 2021)

En el archivo "***src\app\modelos\carro.model.ts***" se definen las propiedades que posteriormente serán inicializadas en el constructor. La Imagen 122 presenta la clase "***Carro***" con sus propiedades y la inicialización de variables en el constructor.

Imagen 122. Clase del modelo de Carro



Fuente: (Autor, 2021)

Para efectos prácticos se comparte el código para copiarlo y pegarlo en el archivo "***src\app\modelos\carro.ts***"

```
export class Carro {
  public codigo: number;
  public placa: string;
  public marca: string;
  public modelo: number;
  public nombreFoto: string;
  public fotobase64: string;

  public constructor(
    cod: number,
    pla: string,
    mar: string,
    mod: number,
    nom: string,
    bas: string
  ) {
    this.codigo = cod;
```

```
    this.placa = pla;
    this.marca = mar;
    this.modelo = mod;
    this.nombreFoto = nom;
    this.fotobase64 = bas;
  }
}
```

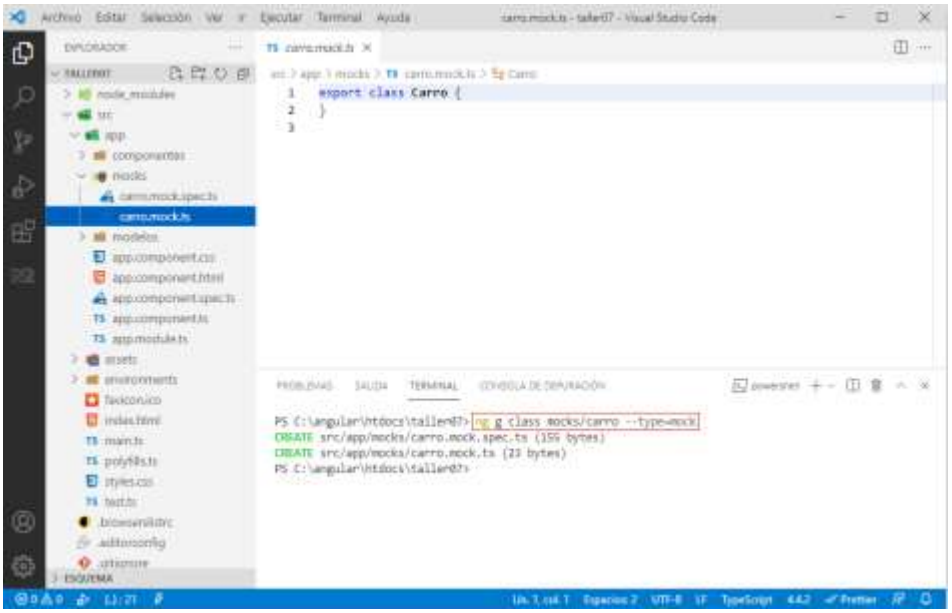
Tabla 29. Taller07, Modelo carro src\app\modelos\carro.ts

7.5.1.7 Mocks

Con el modelo "***Carro***" definido se procede a generar el mock de prueba. El mock contiene la información para realizar pruebas en el Frontend. La Imagen 123 presenta la ejecución del comando en la terminal, la estructura de datos propuesta y el contenido de archivo mock creado con el siguiente comando:

```
ng g class mocks/carro --type=mock
```

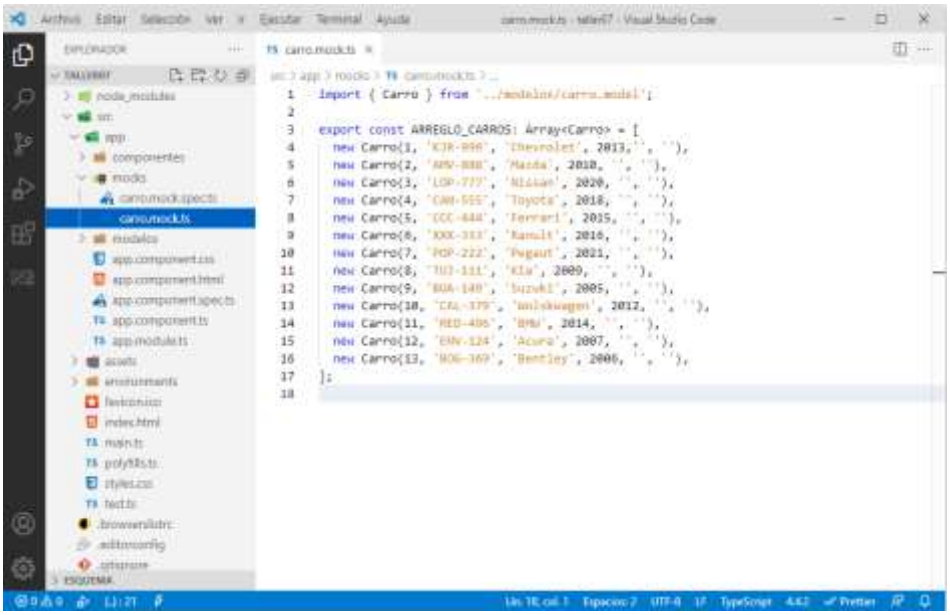
Imagen 123. Carpeta mocks y archivo carro-mock.ts



Fuente: (Autor, 2021)

El mock creado con la "Angular CLI" genera una clase, sin embargo, para los datos de prueba es suficiente con generar un arreglo de datos que permita su utilización en los componentes. La Imagen 124 presenta el contenido del archivo "***src\app\mocks\carro.mock.ts***".

Imagen 124. Preparación del mock para la clase Carro



Fuente: (Autor, 2021)

Para efectos prácticos, en la Tabla 30 se comparte el código para copiarlo y pegarlo en el archivo "***src\app\mocks\carro.mock.ts***"

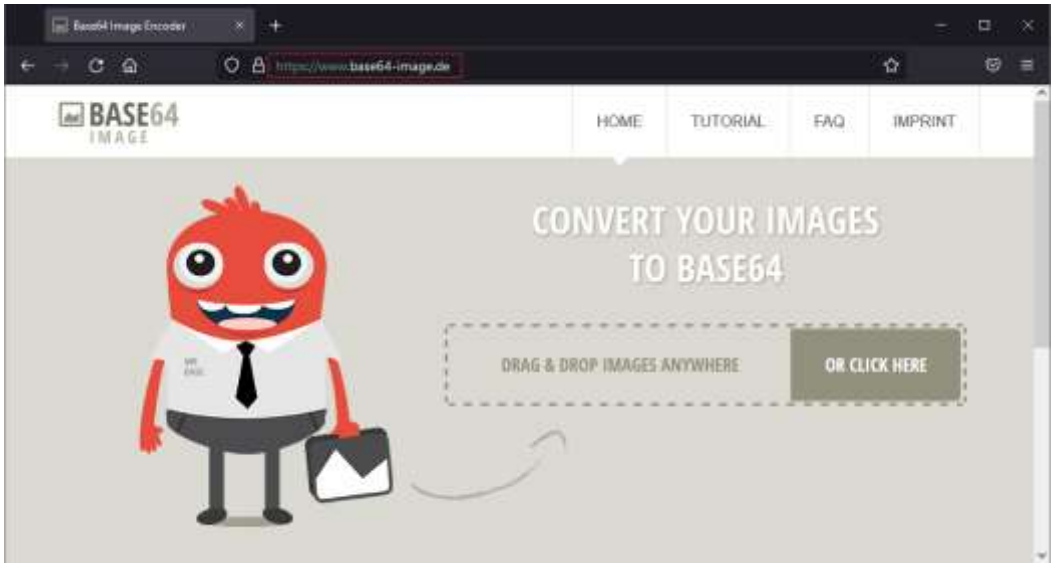
```
import { Carro } from '../modelos/carro.model';


export const ARREGLO_CARROS: Array<Carro> = [
  new Carro(1, 'KJR-999', 'Chevrolet', 2013,'', ''),
  new Carro(2, 'AMV-888', 'Mazda', 2010, '', ''),
  new Carro(3, 'LOP-777', 'Nissan', 2020, '', ''),
  new Carro(4, 'CAN-555', 'Toyota', 2018, '', ''),
  new Carro(5, 'CCC-444', 'Ferrari', 2015, '', ''),
  new Carro(6, 'XXX-333', 'Ranult', 2016, '', ''),
  new Carro(7, 'POP-222', 'Pegaut', 2021, '', ''),
  new Carro(8, 'TUJ-111', 'Kia', 2009, '', ''),
  new Carro(9, 'BUA-149', 'Suzuki', 2005, '', ''),
  new Carro(10, 'CAL-379', 'Wolskwagen', 2012, '', ''),
  new Carro(11, 'MED-496', 'BMW', 2014, '', ''),
  new Carro(12, 'ENV-124', 'Acura', 2007, '', ''),
  new Carro(13, 'BOG-369', 'Bentley', 2006, '', ''),
];
```

Tabla 30. Taller07, mock para el modelo Carro

7.5.1.8 La codificación Base 64 para imágenes

En este taller se utilizarán imágenes codificadas en base64. El algoritmo de codificación Base64 presenta puede presentar una imagen en texto basado en caracteres ASCII. En el navegador de su preferencia puede buscar "***base64 image encoder***" y acceder al sitio de su preferencia para procesar imágenes en base64. En este taller se recomienda el uso del sitio ***https://www.base64-image.de/.*** La Imagen 125 presenta el sitio web utilizado para procesar las imágenes del taller.

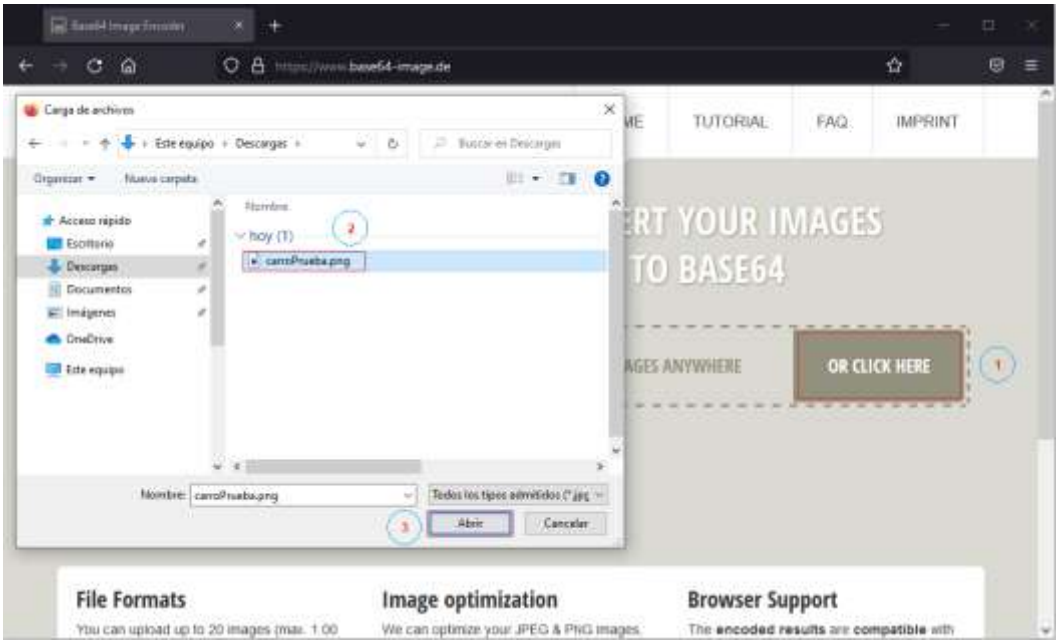
Imagen 125. Sitio Web para codificar imágenes en bade64



Fuente: (Awesome Bonn UG, 2021)

Con el buscador de su preferencia se sugiere que busque una imagen representativa de un carro para el desarrollo del taller. En este taller la imagen a descargar se almacenará en la carpeta descargas. La Imagen 126 presenta la forma cómo se debe utilizar el sitio web sugerido para obtener la imagen codificada en base 64.

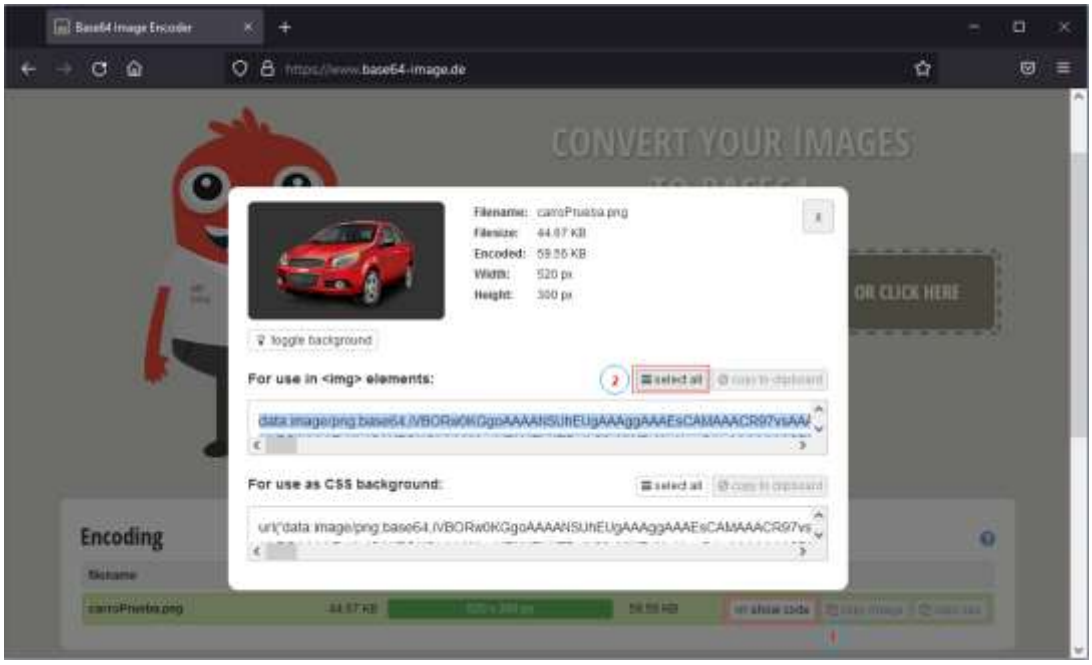
Imagen 126. Utilizando el sitio web para codificar una imagen con base64



Fuente: (Awesome Bonn UG, 2021)

Una vez se carga la imagen en el sitio web sugerido, aparecerá en la parte inferior el resultado de la codificación de la imagen. Haciendo clic en la opción "***show code***" aparecerá en pantalla una ventana modal que contiene la cadena de texto en modo ASCII con la codificación de la imagen. La Imagen 127 presenta el resultado de la codificación de la imagen de prueba.

Imagen 127. Imagen de prueba codificada en base64



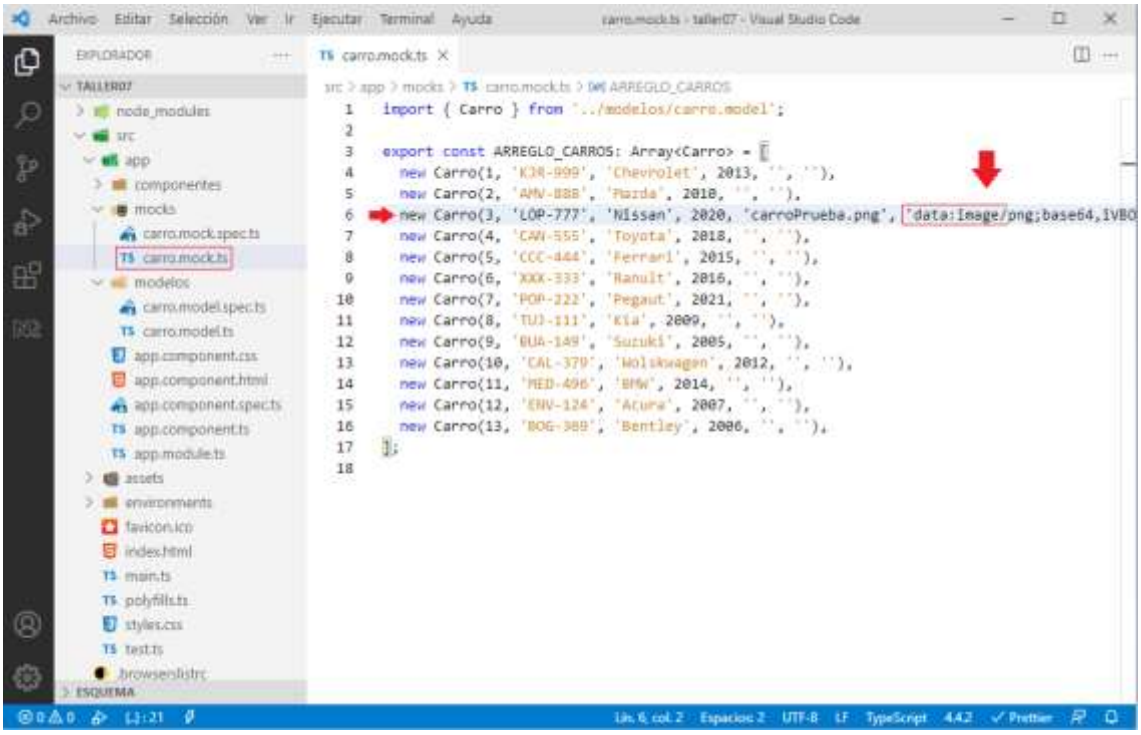
Fuente: (Awesome Bonn UG, 2021)

Seleccionado el base64, se procede a copiar la codificación en el archivo "***src\app\mocks\carro.mock.ts***" en una de las filas de ejemplo. Es clave asegurarse en qué parte del mock se debe agregar la imagen codificada. La Imagen 128 presenta la inclusión de la imagen codificada en el mock.

Advertencia

Al copiar la codificación en el mock, es muy común dejar un espacio en blanco al comienzo del texto "data:image/" este espacio en blanco no permitirá la visualización correcta de la imagen posteriormente.

Imagen 128. Inclusión de la imagen codificada en base64 al mock



Fuente: (Autor, 2021)

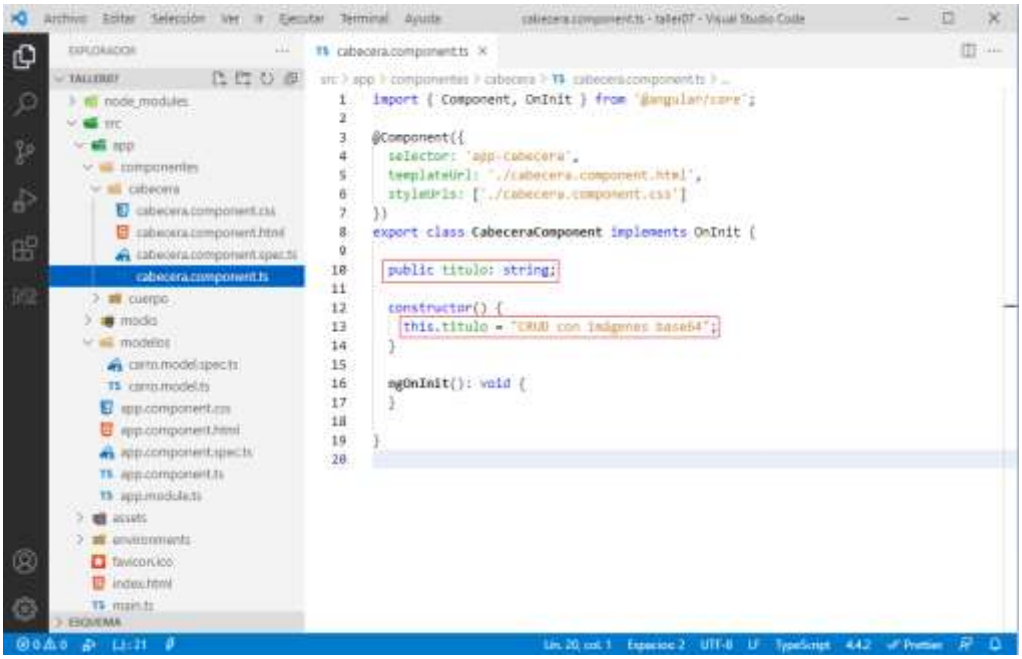
7.5.1.9 *Componente Cabecera*

El componente que hace referencia a la cabecera permitirá visualizar un estilo de encabezado compuesto por un logo y un texto, de tal manera la distribución visual del taller sea agradable para el usuario final. Para presentar un título en el encabezad, se utilizará un atributo de la clase, con su respectiva interpolación en la plantilla del componente.

Las propiedades o atributos deben ser inicializadas en el constructor como una buena practica de desarrollo. La

Imagen 129 presenta el contenido de la clase del componente cabecera.

Imagen 129. Clase del componente cabecera



Fuente: (Autor, 2021)

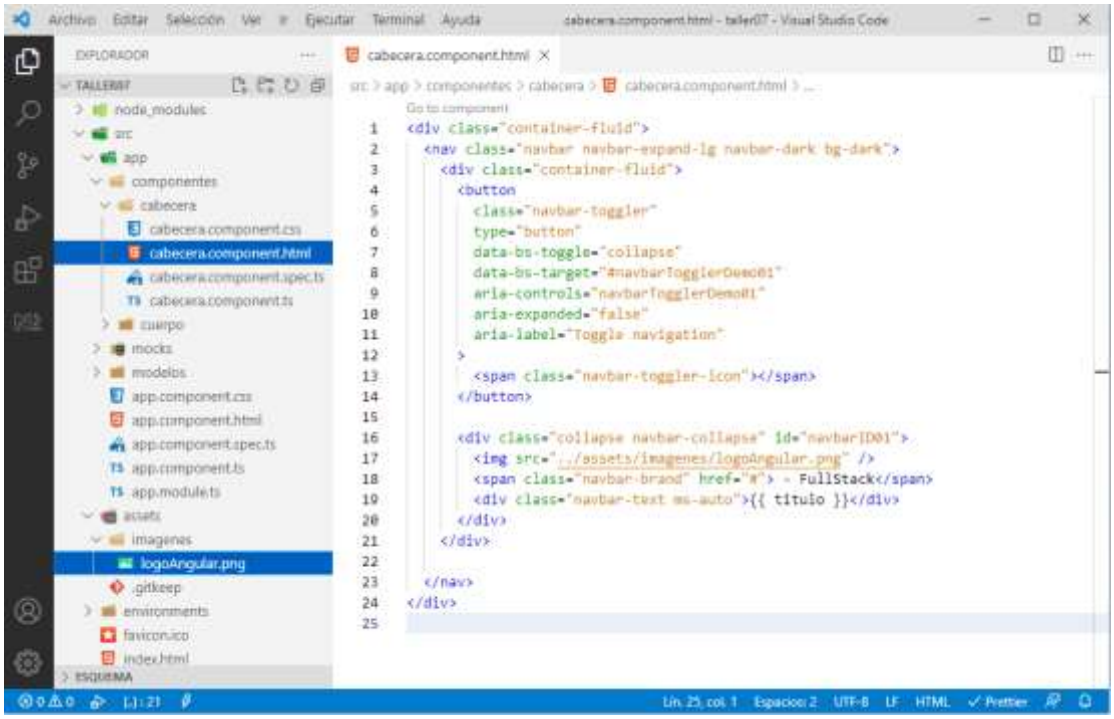
La cabecera debe presentar una imagen en la parte izquierda y un texto en la parte derecha, por ende, es necesario crear en el proyecto una carpeta para almacenar las imágenes. En la carpeta "***src\assets***" se debe crear una nueva carpeta llamada imágenes y en su interior se debe colocar la imagen que se desee presentar en el taller. La Imagen 130 presenta la imagen la ruta en donde debe ser almacenada la imagen a presentar.

Imagen 130. Logo componente cabecera en folder

Fuente: (Autor, 2021)

En la plantilla del componente cabecera "***src\app\componentes\cabecera\cabecera.component.html***" se presenta el maquetado sugerido en este taller. La Imagen 131 el maquetado en Bootstrap sugerido en este taller.

Imagen 131. Plantilla componente cabecera con maquetado en Bootstrap



Fuente: (Autor, 2021)

Para efectos prácticos, en la Tabla 31 se comparte el código para copiarlo y pegarlo en el archivo "***src\app\componentes\cabecera\cabecera.component.html***".

```
<div class="container-fluid">
  <nav class="navbar navbar-expand-lg navbar-dark bg-dark">
    <div class="container-fluid">
      <button
        class="navbar-toggler"
        type="button"
        data-bs-toggle="collapse"
        data-bs-target="#navbarTogglerDemo01"
        aria-controls="navbarTogglerDemo01"
        aria-expanded="false"
        aria-label="Toggle navigation"
      >
        <span class="navbar-toggler-icon"></span>
      </button>

      <div class="collapse navbar-collapse" id="navbarID01">
        <img src="../assets/imagenes/logoAngular.png" />
        <span class="navbar-brand" href="#"> - FullStack</span>
        <div class="navbar-text ms-auto">{{ titulo }}</div>
```

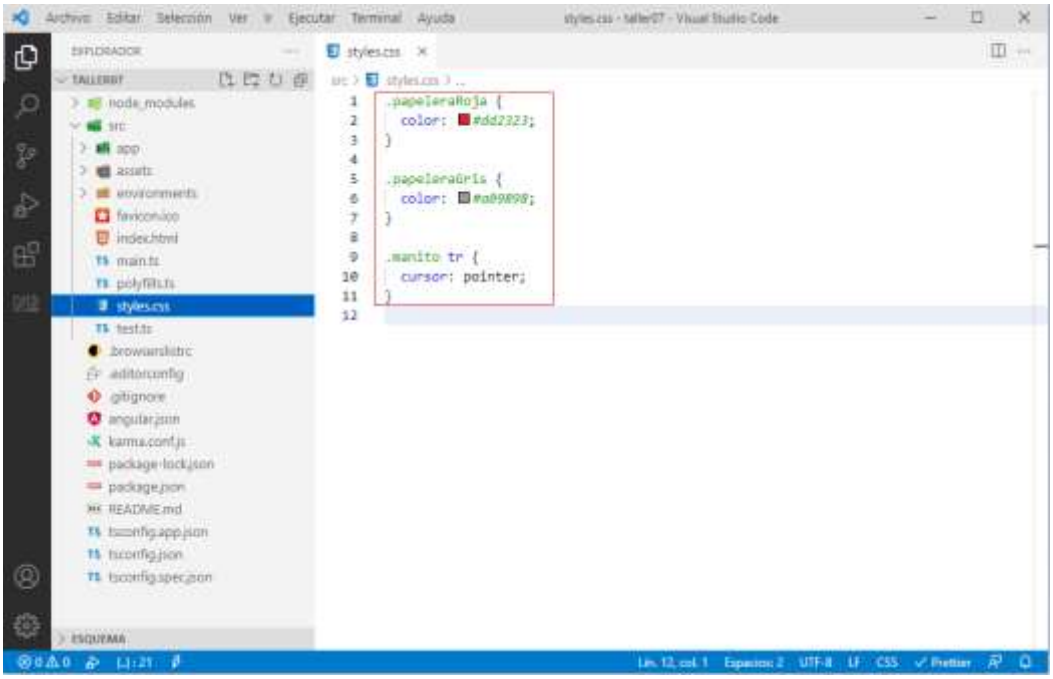
```
      </div>
    </div>


  </nav>
</div>
```

Tabla 31. Taller07, plantilla de la componente cabecera con maquetado

7.5.1.10 Estilos Globales

Para dar más fluidez visual al taller, se agregarán estilos globales a toda la aplicación. Para esto se modificará el archivo "***src\styles.cs***". La Imagen 132 presenta 3 estilos sencillos que pueden ser utilizados en cualquier componente.

Imagen 132. Estilos para la plantilla del componente



Fuente: (Autor, 2021)

Para efectos prácticos, en la Tabla 32 se presenta el código para copiarlo y pegarlo en el archivo "***src\styles.css***"

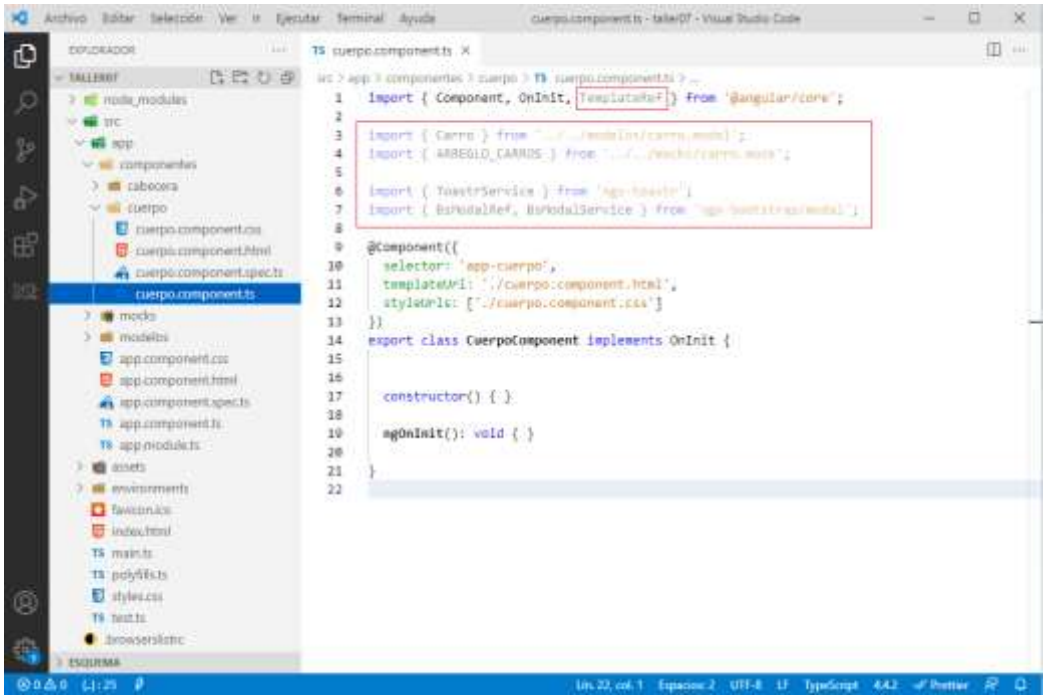
```
.papeleraRoja {
  color: #dd2323;
}

.papeleraGris {
  color: #a09898;
}

.manito tr {
  cursor: pointer;
}
```

Tabla 32. Estilos globales del taller07

7.5.1.11 Componente Cuerpo

El objetivo del componente cuerpo es presentar las opciones para crear, listar, actualizar y eliminar registros de un arreglo de objetos de tipo Carro, el cual será utilizado en el Front como datos de prueba. La Imagen 133 presenta la inclusión del modelo Carro y su correspondiente arreglo de datos de prueba.

Imagen 133. Inclusión del modelo y datos de prueba



Fuente: (Autor, 2021)

El código utilizado en el archivo "**src\app\componentes\cuerpo\cuerpo.component.ts**" se presenta en la Tabla 33.

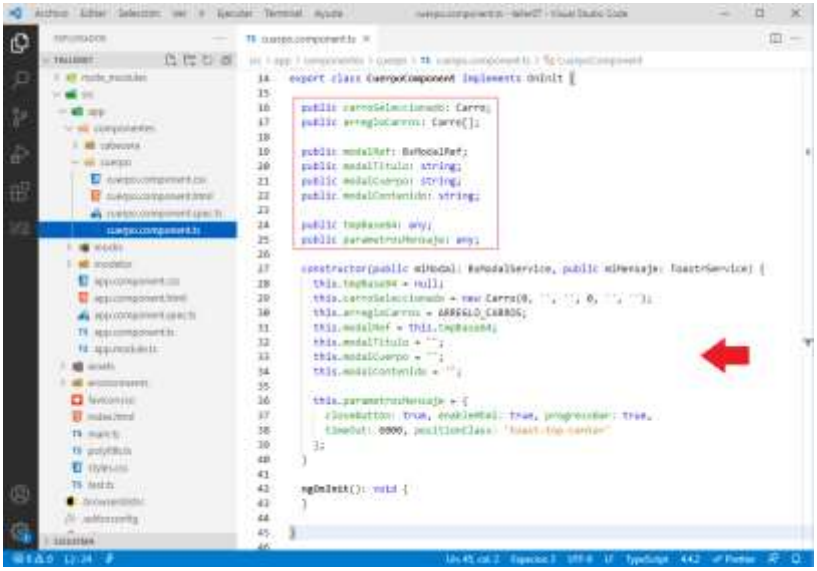
```
import { Component, OnInit } from '@angular/core';

import { Carro } from '../../modelos/carro.model';
import { ARREGLO_CARROS } from '../../mocks/carro.mock';

import { ToastrService } from 'ngx-toastr';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';

@Component({
  selector: 'app-cuerpo',
  templateUrl: './cuerpo.component.html',
  styleUrls: ['./cuerpo.component.css']
})
export class CuerpoComponent implements OnInit {

  constructor() { }

  ngOnInit(): void {
```

```
  }
}
```

Tabla 33. Clase del componente cuerpo, inclusión del modelo y mock

En el archivo "**src\app\componentes\cuerpo\cuerpo.component.ts**", el cual corresponde a la clase del cuerpo, se crean las propiedades y se inicializan en el constructor. La Imagen 134 presenta las propiedades requeridas para el componente.

Imagen 134. Propiedades e inicialización en el constructor



Fuente: (Autor, 2021)

El código utilizado en el archivo "**src\app\componentes\cuerpo\cuerpo.component.ts**" se presenta en la Tabla 34.

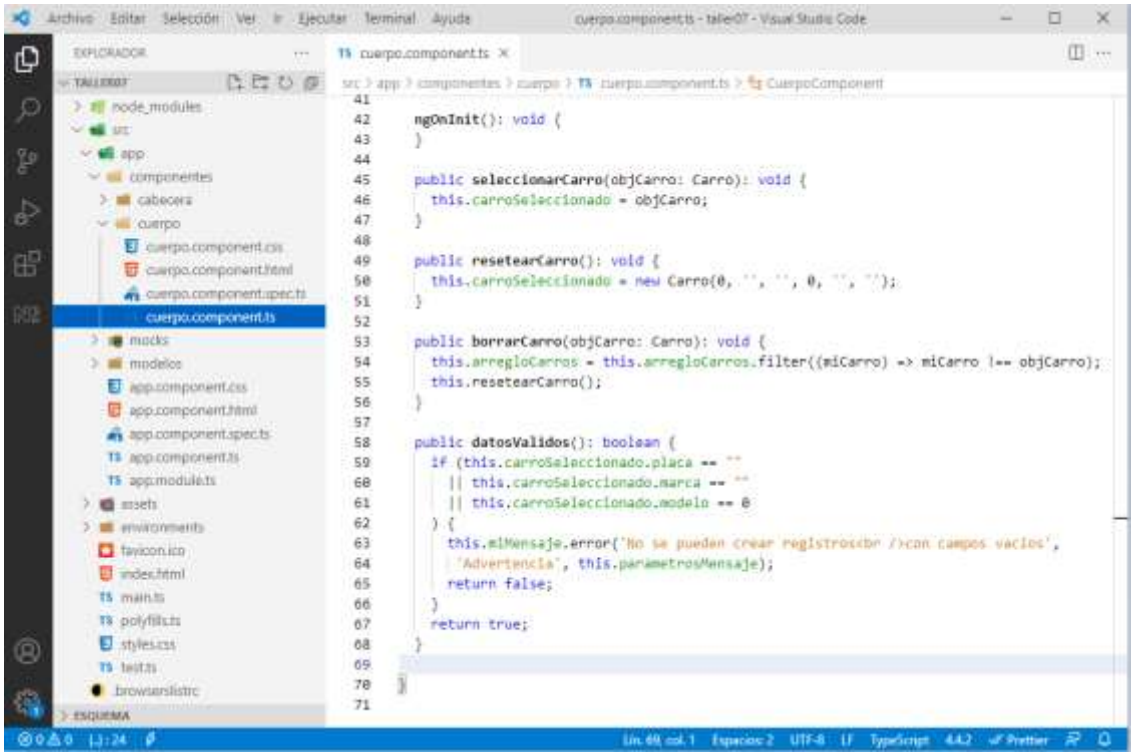
```
public carroSeleccionado: Carro;
 public arregloCarros: Carro[];

 public modalRef: BsModalRef;
 public modalTitulo: string;
 public modalCuerpo: string;
 public modalContenido: string;

 public tmpBase64: any;

 public parametrosMensaje: any;

 constructor(public miModal: BsModalService, public miMensaje: ToastrService) {
```

```
    this.tmpBase64 = null;
    this.carroSeleccionado = new Carro(0, '', '', 0, '', '');
    this.arregloCarros = [];

    this.modalRef = this.tmpBase64;
    this.modalTitulo = "";
    this.modalCuerpo = "";
    this.modalContenido = "";

    this.parametrosMensaje = {
      closeButton: true,
      enableHtml: true,
      progressBar: true,
      timeOut: 6000,
      positionClass: 'toast-top-center'
    };
  }

  ngOnInit(): void {
    this.arregloCarros = ARREGLO_CARROS;
  }
```

Tabla 34. Taller07, clase de la componente cuerpo propiedades e inicialización

La Imagen 135 presenta los métodos necesarios para apoyar la creación del CRUD en este componentes.

Imagen 135. Métodos requeridos en la clase del componente cuerpo



Fuente: (Autor, 2021)

El código utilizado en el archivo "**src\app\componentes\cuerpo\cuerpo.component.ts**" se presenta en la Tabla 35.

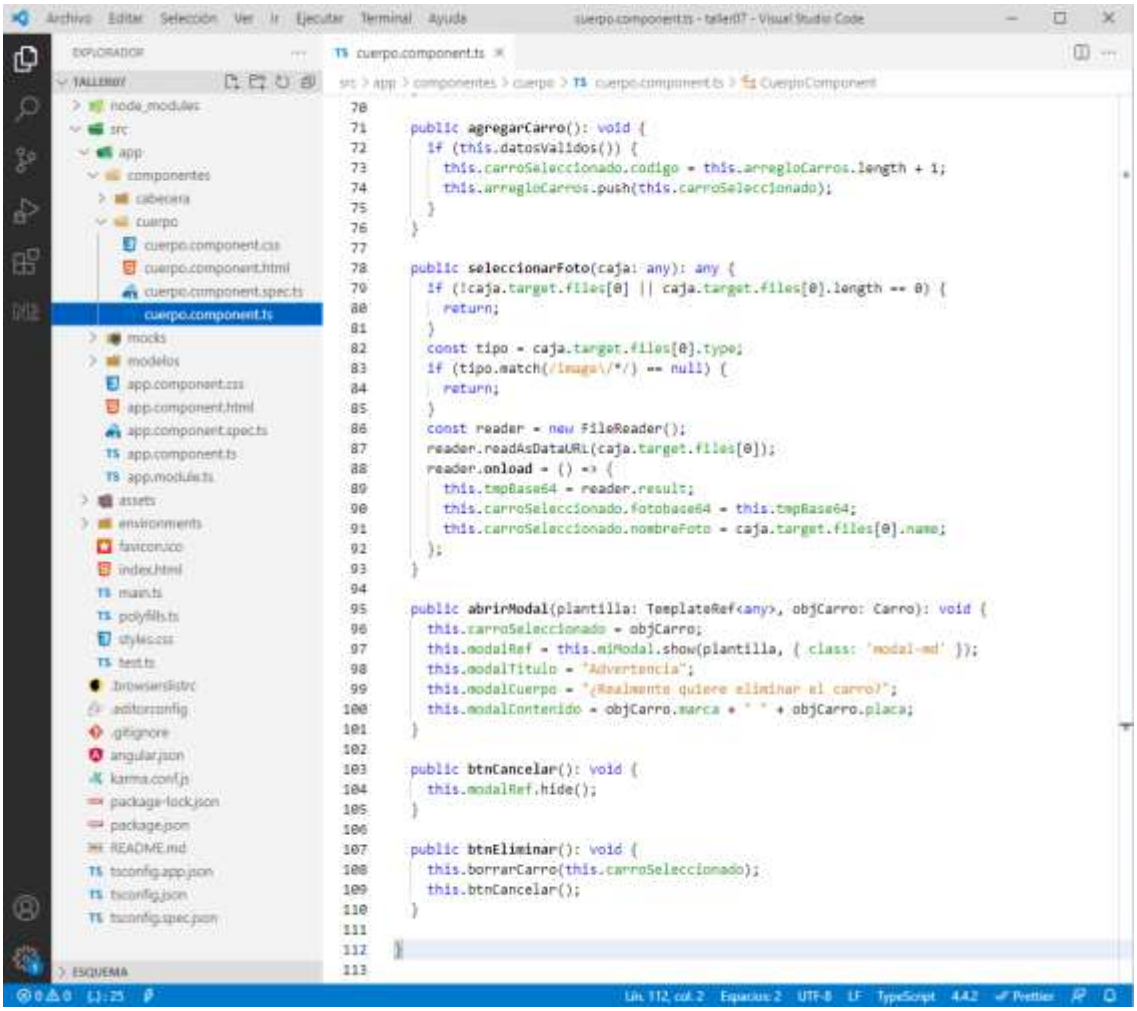
```
public seleccionarCarro(objCarro: Carro): void {
  this.carroSeleccionado = objCarro;
}

public resetearCarro(): void {
  this.carroSeleccionado = new Carro(0, '', '', 0, '', '');
}

public borrarCarro(objCarro: Carro): void {
  this.arregloCarros = this.arregloCarros.filter((miCarro) => miCarro !== objCarro);
  this.resetearCarro();
}

public datosValidos(): boolean {
  if (this.carroSeleccionado.placa == ""
    || this.carroSeleccionado.marca == ""
```

```
      || this.carroSeleccionado.modelo == 0
    ) {
      this.miMensaje.error('No se pueden crear registros<br />con campos vacíos',
       'Advertencia', this.parametrosMensaje);
      return false;
    }
    return true;
  }
```

Tabla 35. Métodos básicos para la creación del CRUD

A continuación, en la Imagen 136, se presentan los métodos para adicionar carros, cargar la foto de un carro, permitir las asignación de la imagen en base64 a una propiedad y los métodos para abrir la ventana modal con sus botones de cerrar y cancelar.

Imagen 136. Métodos para agregar Carros, seleccionar fotos y abrir modales



Fuente: (Autor, 2021)

El código utilizado en el archivo "**src\app\componentes\cuerpo\cuerpo.component.ts**" se presenta en la Tabla 36.

```
public agregarCarro(): void {
    if (this.datosValidos()) {
      this.carroSeleccionado.codigo = this.arregloCarros.length + 1;
      this.arregloCarros.push(this.carroSeleccionado);
    }
  }

  public seleccionarFoto(caja: any): any {
    if (!caja.target.files[0] || caja.target.files[0].length == 0) {
      return;
    }
    const tipo = caja.target.files[0].type;
    if (tipo.match(/image\/*/) == null) {
      return;
    }
    const reader = new FileReader();
    reader.readAsDataURL(caja.target.files[0]);
    reader.onload = () => {
      this.tmpBase64 = reader.result;
      this.carroSeleccionado.fotobase64 = this.tmpBase64;
      this.carroSeleccionado.nombreFoto = caja.target.files[0].name;
    };
  }

  public abrirModal(plantilla: TemplateRef<any>, objCarro: Carro): void {
    this.carroSeleccionado = objCarro;
    this.modalRef = this.miModal.show(plantilla, { class: 'modal-md' });
    this.modalTitulo = "Advertencia";
    this.modalCuerpo = "¿Realmente quiere eliminar el carro?";
    this.modalContenido = objCarro.marca + " " + objCarro.placa;
  }

  public btnCancelar(): void {
    this.modalRef.hide();
  }

  public btnEliminar(): void {
```

```
    this.borrarCarro(this.carroSeleccionado);
    this.btnCancelar();
  }
```

Tabla 36. Métodos crear carro, seleccionar foto y maneja de la modal

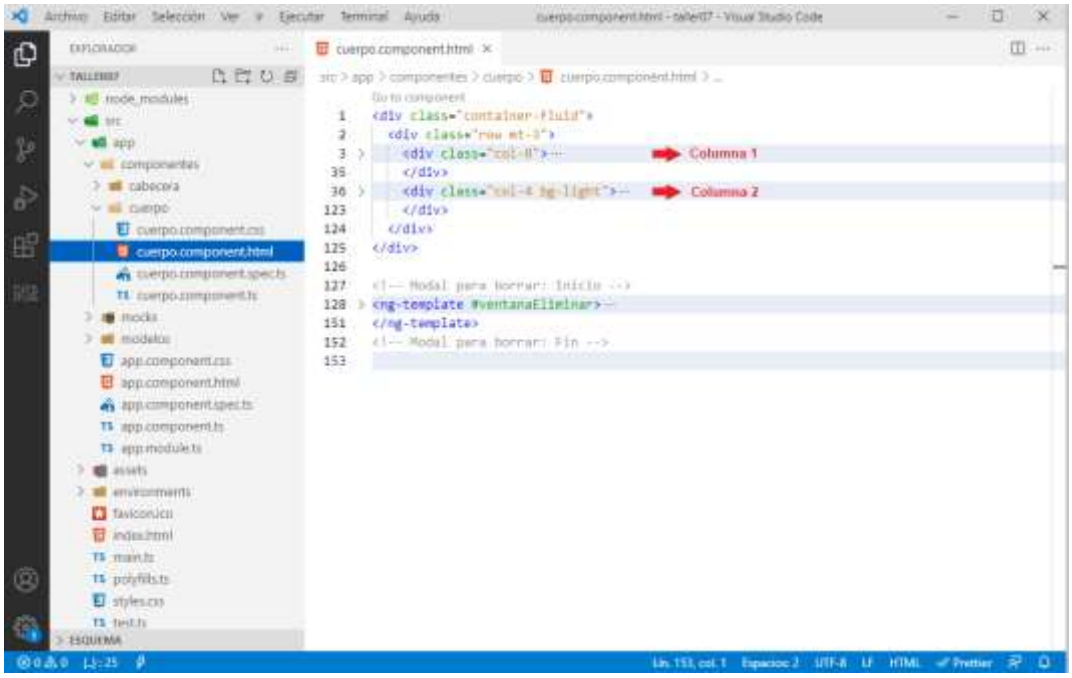
Dado que este taller utiliza imágenes para presentar en la vista, es necesario tener una imagen que se puede presentar en la eventualidad que alguna de las imágenes no pueda ser cargada. La imagen sed debe almacenar en la carpeta "***src\assets***". La Imagen 137 presenta la ruta donde se sugiere se almacene la imagen que se presentará por defecto.

Imagen 137. Imagen por defecto

Fuente: (Autor, 2021)

Para la plantilla del componente cuerpo se utilizará la librería Bootstrap. Todo el contenido HTML estará en un bloque "***div***" utilizando la clase "***container-fluid***", en su interior se definirá un bloque que defina una fila que posteriormente tendrá dos columnas y un maquetado para mostrar una ventana modal para eliminar registros. Dado que el contenido de cada columna es un poco más extenso se presentará más adelante en la tabla Tabla 25. La Imagen 102 presenta el maquetado de la plantilla del componente cuerpo.

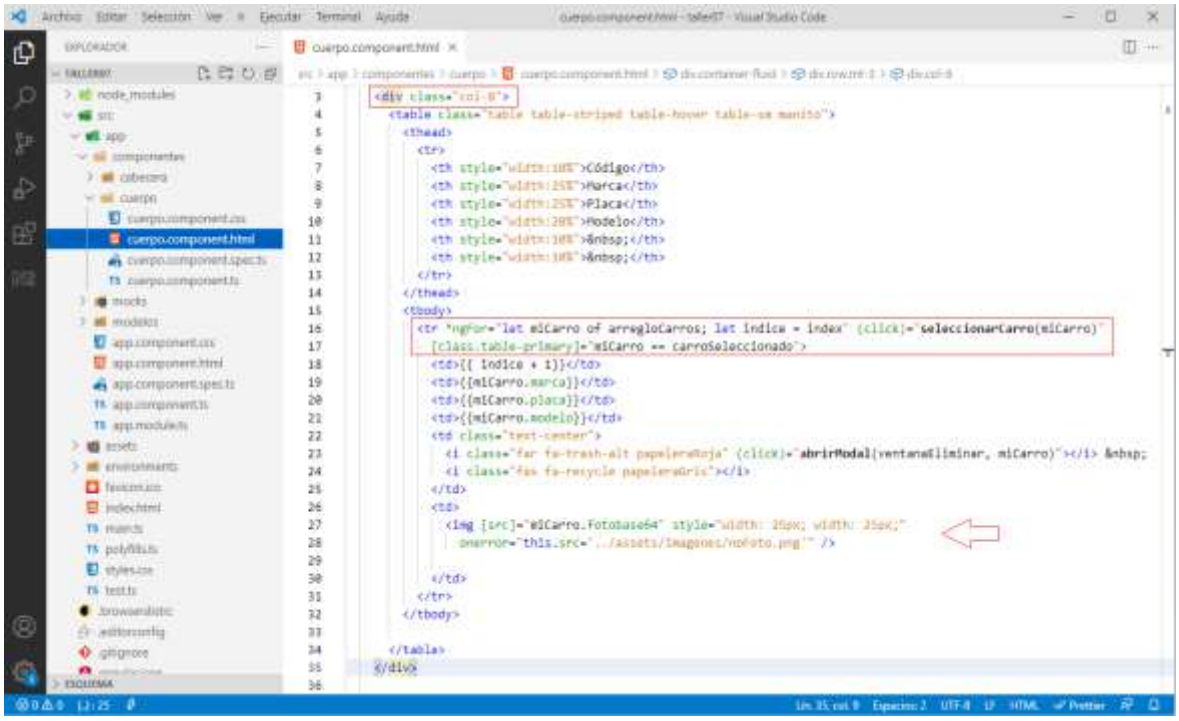
Imagen 138. Taller07, maquetado plantilla cuerpo



Fuente: (Autor, 2021)

La Imagen 139 presenta el contenido de la columna 1, la cual presenta la tabla que carga la información del mock, adicionalmente se tendrán dos iconos para eliminar y editar, adicionalmente una imagen para previsualizar la imagen del Carro.

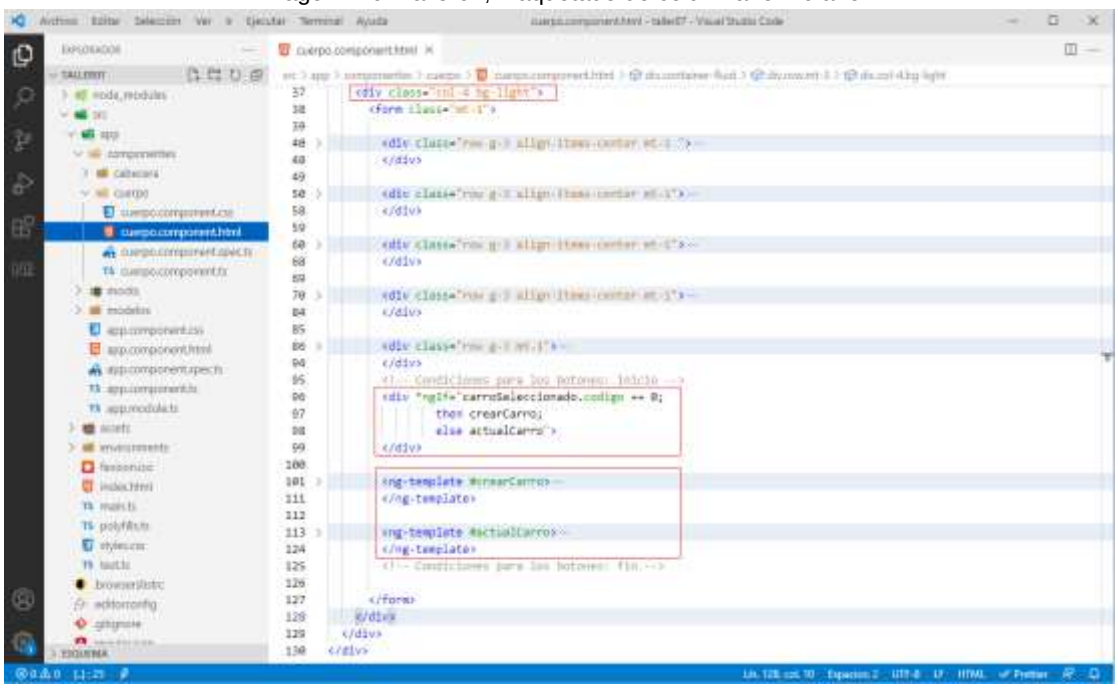
Imagen 139. Taller07, maquetado de columna mostrar información



Fuente: (Autor, 2021)

La Imagen 140 presenta el contenido de la segunda columna, la cual contiene un formulario para crear y editar. Adicionalmente se presenta un campo para agregar la imagen y se tendrán dos "***templates***" para presentar los botones para crear carro y los botones para actualizar carro.

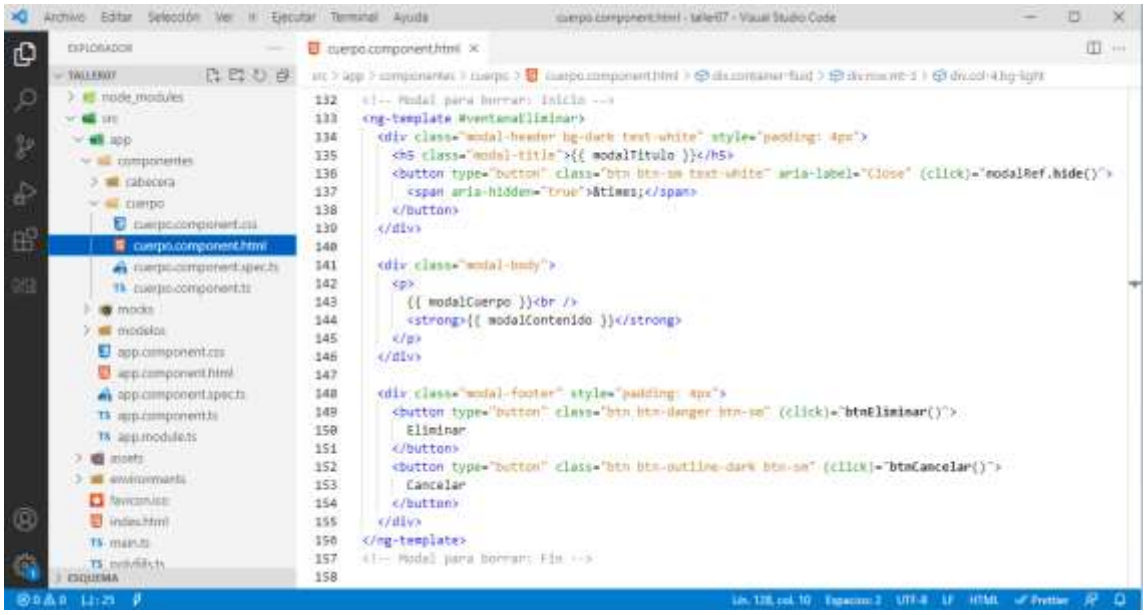
Imagen 140. Taller07, maquetado de columna formulario



Fuente: (Autor, 2021)

Para finalizar se presenta el maquetado de la ventana modal para eliminar carros, la cual corresponde a un "***template***" como se presenta en la Imagen 141.

Imagen 141. Taller07, maquetado de ventana modal Eliminar



Fuente: (Autor, 2021)

Todo el código utilizado en el archivo "**src\app\componentes\cuerpo\cuerpo.component.html**" se presenta en la Tabla 25.

```
<div class="container-fluid">
  <div class="row mt-3">
    <div class="col-8">
      <table class="table table-striped table-hover table-sm manito">
        <thead>
          <tr>
            <th style="width:10%">Código</th>
            <th style="width:25%">Marca</th>
            <th style="width:25%">Placa</th>
            <th style="width:20%">Modelo</th>
            <th style="width:10%"> </th>
            <th style="width:10%"> </th>
          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let miCarro of arregloCarros; let indice = index" (click)="seleccionarCarro(miCarro)"
            [class.table-primary]="miCarro == carroSeleccionado">
            <td>{{ indice + 1}}</td>
            <td>{{miCarro.marca}}</td>
            <td>{{miCarro.placa}}</td>
            <td>{{miCarro.modelo}}</td>
            <td class="text-center">
              <i class="far fa-trash-alt papeleraRoja" (click)="abrirModal(ventanaEliminar, miCarro)"></i>  
              <i class="fas fa-recycle papeleraGris"></i>
            </td>
```

```
          <td>
            <img [src]="miCarro.fotobase64" style="width: 25px; width: 25px;"
              onerror="this.src='../assets/imagenes/noDisponible.png'" />
          </td>
        </tr>
      </tbody>

    </table>
  </div>
  <div class="col-4 bg-light">
    <form class="mt-1">

      <div class="row g-3 align-items-center mt-1 ">
        <div class="col-3">
          <label for="marca" class="text-success">Marca:</label>
        </div>
        <div class="col-8">
          <input type="text" id="marca" name="marca" class="form-control form-control-sm"
            [(ngModel)]="carroSeleccionado.marca" />
        </div>
      </div>

      <div class="row g-3 align-items-center mt-1">
        <div class="col-3">
          <label for="placa" class="text-success">Placa:</label>
        </div>
        <div class="col-8">
          <input type="text" id="placa" name="placa" class="form-control form-control-sm"
            [(ngModel)]="carroSeleccionado.placa" />
        </div>
      </div>

      <div class="row g-3 align-items-center mt-1">
        <div class="col-3">
          <label for="modelo" class="text-success">Modelo:</label>
        </div>
        <div class="col-8">
          <input type="text" id="modelo" name="modelo" class="form-control form-control-sm"
            [(ngModel)]="carroSeleccionado.modelo" />
        </div>
      </div>
      <div class="row g-3 align-items-center mt-1">
        <div class="col-3">
          <label for="foto" class="text-success">Foto:</label>
        </div>
        <div class="col-8">
          <div class="input-group">
            <input type="text" id="foto" name="foto" class="form-control form-control-sm"
```

```
            [(ngModel)]="carroSeleccionado.nombreFoto" />
          <button class="btn btn-primary" (click)="cajaOculta.click()">
            <span class="input-group">+</span>
            <input #cajaOculta type="file" style="display: none;" (change)="seleccionarFoto($event)" />
          </button>
        </div>
      </div>
    </div>

    <div class="row g-3 mt-1">
      <div class="col-3">
      </div>
      <div class="col-8">
        <img class="img-thumbnail" style="width: 200px; height: 100%;"
          onerror="this.src='../assets/imagenes/noDisponible.png'" [src]="carroSeleccionado.fotobase64" />
      </div>

    </div>
    <!-- Condiciones para los botones: inicio -->
    <div *ngIf="carroSeleccionado.codigo == 0;
          then crearCarro;
          else actualCarro">
    </div>
    <ng-template #crearCarro>
      <div class="row g-3 align-items-center mt-1">
        <div class="col-3">
        </div>
        <div class="col-8">
          <div class="d-grid gap-1 col-12">
            <button class="btn btn-primary btn-sm btn-primary" (click)="agregarCarro()">Crear registro</button>
          </div>
        </div>
      </div>
    </ng-template>

    <ng-template #actualCarro>
      <div class="row g-3 align-items-center mt-1">
        <div class="col-3">
        </div>
        <div class="col-8">
          <div class="d-grid gap-1 col-12">
            <button class="btn btn-primary btn-sm btn-success" (click)="resetearCarro()">Actualizar</button>
            <button class="btn btn-primary btn-sm btn-secondary" (click)="resetearCarro()">Cancelar</button>
          </div>
        </div>
      </div>
    </ng-template>
    <!-- Condiciones para los botones: fin -->
```

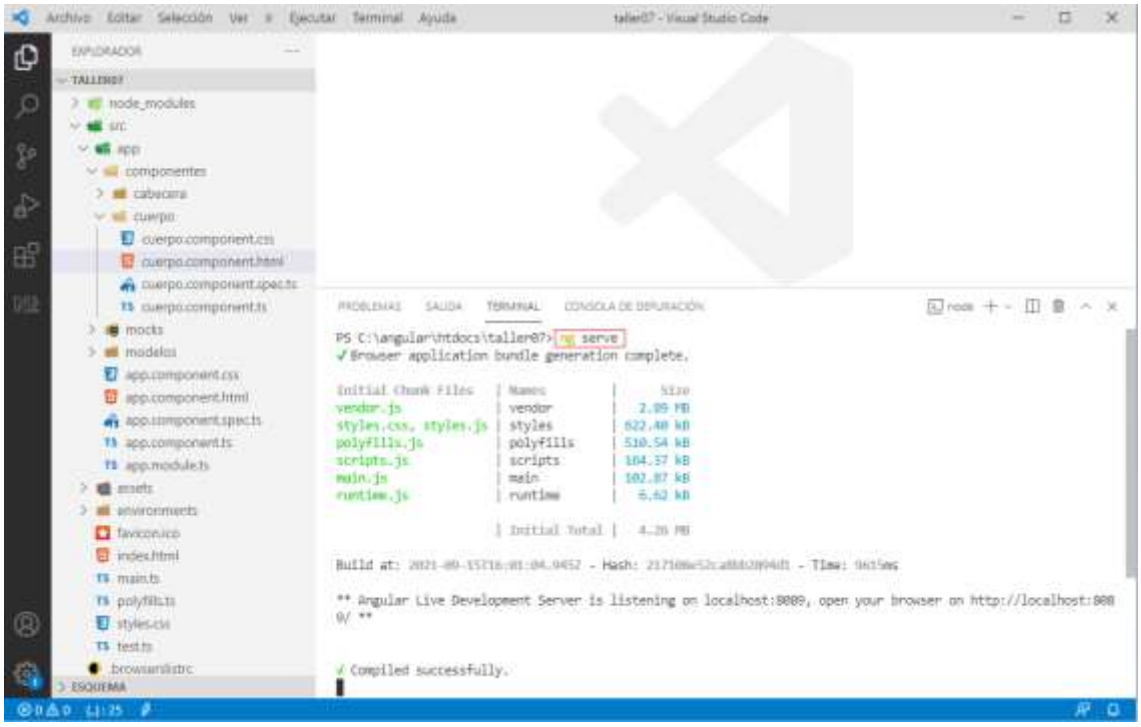
```
      </form>
    </div>
  </div>
</div>
<!-- Modal para borrar: Inicio -->
<ng-template #ventanaEliminar>
  <div class="modal-header bg-dark text-white" style="padding: 4px">
    <h5 class="modal-title">{{ modalTitulo }}</h5>
    <button type="button" class="btn btn-sm text-white" aria-label="Close" (click)="modalRef.hide()">
      <span aria-hidden="true">&times;</span>
    </button>
  </div>

  <div class="modal-body">
    <p>
      {{ modalCuerpo }}<br />
      <strong>{{ modalContenido }}</strong>
    </p>
  </div>

  <div class="modal-footer" style="padding: 4px">
    <button type="button" class="btn btn-danger btn-sm" (click)="btnEliminar()">
      Eliminar
    </button>
    <button type="button" class="btn btn-outline-dark btn-sm" (click)="btnCancelar()">
      Cancelar
    </button>
  </div>
</ng-template>
<!-- Modal para borrar: Fin -->
```

Tabla 37. Taller07, plantilla completa del componente cuerpo

La Imagen 142 presenta la ejecución del comando "***ng serve***" desde la terminal de Visual Estudio Code.

```
ng serve
```

Imagen 142. Ejecución del taller07

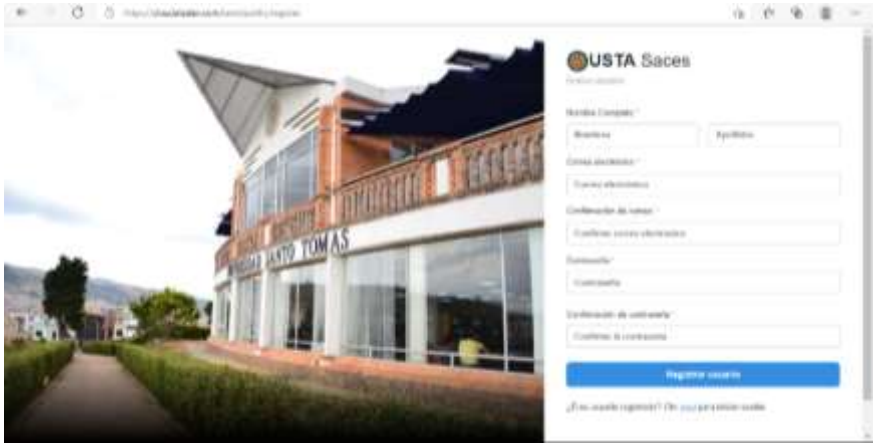


Fuente: (Autor, 2021)

8. Herramienta Software

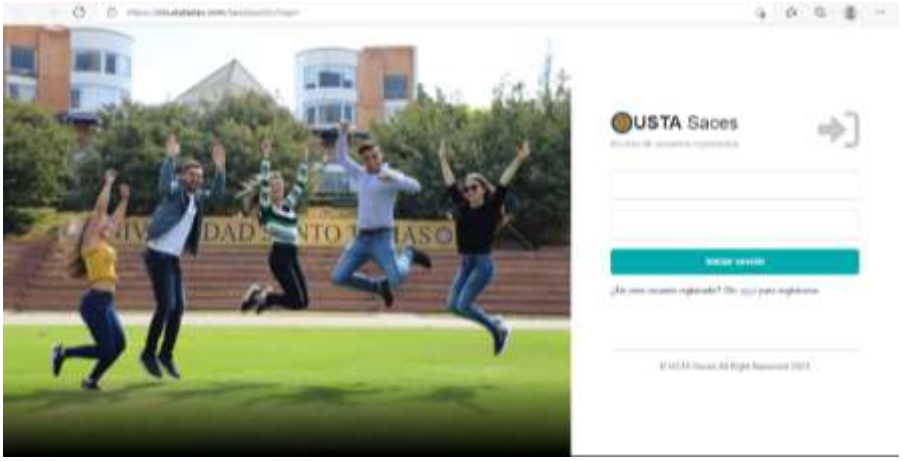
A continuación, se presentan los pantallazos de la aplicación final generada de acuerdo al material de las guías de conocimiento, los pantallazos no corresponden a los talleres de cada una de las guías, por el contrario, son el resultado de aplicar las guías de conocimiento a un proyecto adicional de carácter externo.

Imagen 143. Registro de usuario



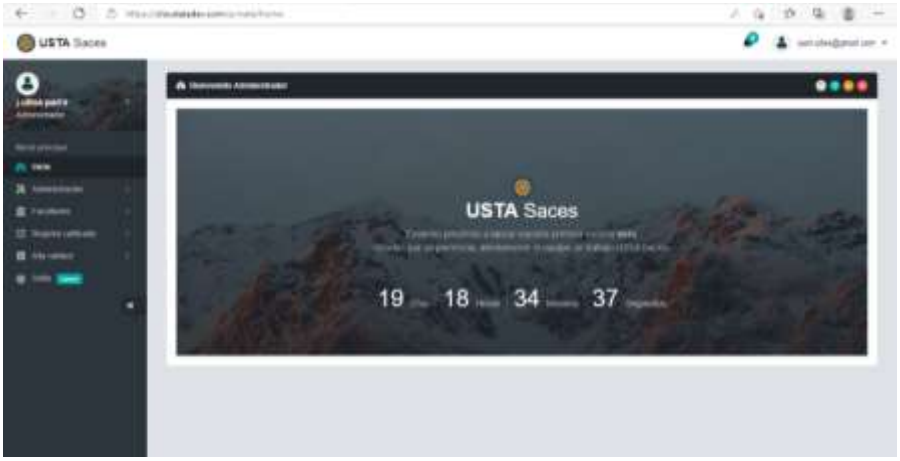
Fuente: (Autor, 2021)

Imagen 144. Inicio de sesión usuario



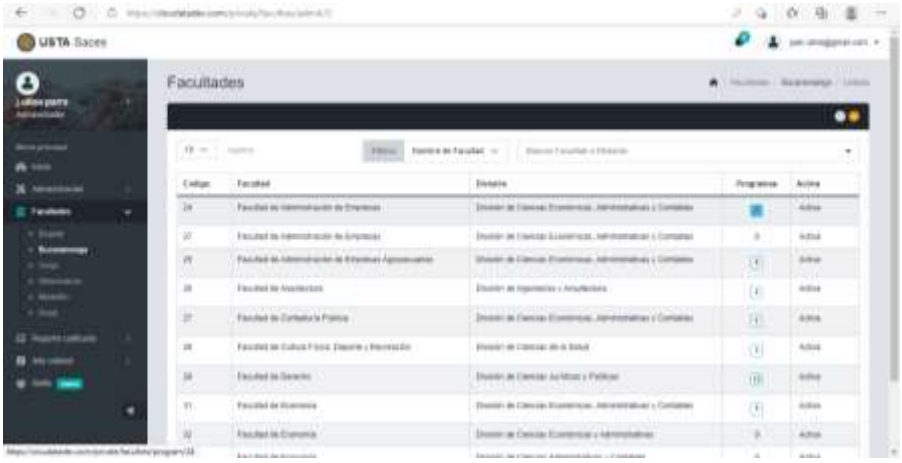
Fuente: (Autor, 2021)

Imagen 145. Bienvenida al sistema



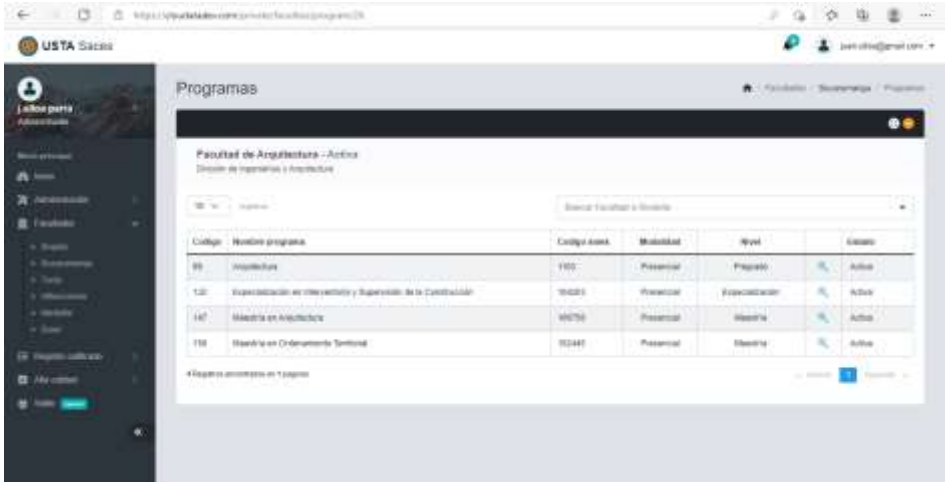
Fuente: (Autor, 2021)

Imagen 146. Sección de facultades



Fuente: (Autor, 2021)

Imagen 147. Sección de Programas

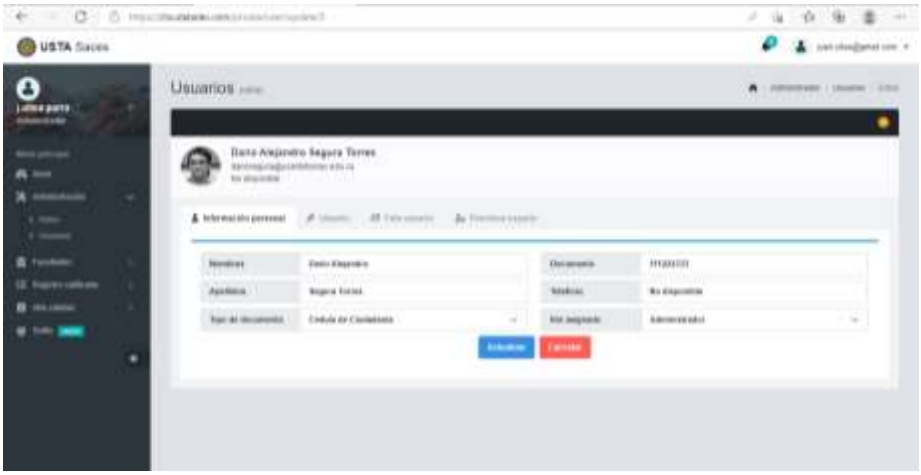


Fuente: (Autor, 2021)

Imagen 148. Roles con ventanas modales

Fuente: (Autor, 2021)

Imagen 149. Edición de usuario



Fuente: (Autor, 2021)

9. **Conclusiones**

1. Después de validar el proyecto en la universidad Santo Tomas y en Mintic se sugiere que la asignatura Fullstack Se divida en dos materias Fullstack Angular y Fullstack NodeJS.
2. Si no se pueden dividir las asignaturas de la conclusión anterior, se sugiere la creación de dos nuevas asignaturas: Frontend y Backend
3. Es posible que un docente con interés en desarrollo web se haga experto en un framework frontend si utiliza estas guías de conocimiento para su formación.
4. Es 100% viable crear una aplicación profesional con el uso de estas guías de conocimiento.
5. Todo proyecto debe tener una verificación y validación. Este proceso suele durar mucho tiempo, sin embargo, permite refinar el proyecto para entregar proyectos de calidad.
6. Todo estudiante de Ingeniería de sistemas debe invertir su tiempo extra clase en perfeccionar sus habilidades en desarrollo de software.

10. Referencias

López Herrera, P. (2016). *Comparación del desempeño de los Sistemas Gestores de Bases de Datos MySQL y PostgreSQL*. Universidad Autónoma del Estado de México. Recuperado de: http://hdl.handle.net/20.500.11799/62548

Navajas Manuel, S. & Pamplona Sonia, S. (2009). *Evaluación de la facilidad de aprendizaje de frameworks JavaScript: Backbone, Angular y Ember.* Recuperado de: http://hdl.handle.net/20.500.12226/272

Llerena Ocaña, L., Fernández Villacres, E., Viscaino Naranjo, A, & Baño Naranjo, P. (2021). *Frameworks basados en typescript para el desarrollo de aplicaciones web interactivas.* Recuperado de: https://doi.org/10.46377/dilemas.v8i3.2644

Alamilla Hernández, L., Pérez Romero, V., Sosa González, S., & Valentín Rodríguez, J. (2021). *Arquitectura REST pnara el desarrollo de aplicaciones web empresariales*. Revista Electrónica Sobre Tecnología, Educación Y Sociedad. Recuperado de: https://www.ctes.org.mx/index.php/ctes/article/view/748

Cámara Braña, S. (2020). *Functional Reactive Programming in ClientSide web applications.* Universidad Politécnica de Madrid. Recuperado de: https://oa.upm.es/64383/