Abril - 2024

ANGULAR V.16

con TypeScript + node

Andres Felipe Cárdenas Alarcon
Sergio Arley Puerto Moreno
Facultad de Ingeniería de Sistemas

TABLA DE CONTENDIÓ

Lista de imágenes

Lista de tablas

1. Requisitos previos del lector

Para empezar a aprender de Angular, es importante disponer de algunos conocimientos y manejo de algunas de las herramientas básicas que se trabajarán y se usarán a lo largo de este material. En la siguiente lista se presentan los requisitos previos con los que se debe estar familiarizado.

- **Conocimientos básicos de JavaScript:** Angular se basa bajo el lenguaje de programación interpretado JavaScript, por lo hay que tener un cierto conocimiento del lenguaje.
- **Familiaridad con la línea de comandos/terminal:** A lo largo del proceso de instalación y cuando se utiliza Angular, la creación de componentes y otros archivos, se trabajan bajo la línea de comandos (en Windows) o el terminal (en macOS y Linux). La familiaridad con los comandos básicos y la navegación es esencial para empezar a utilizar Angular.
- **Familiaridad con TypeScript (opcional):** Aunque no es estrictamente necesario, tener un conocimiento básico de TypeScript puede ser útil cuando se trabaja con Angular, ya que el framework está construido sobre TypeScript. Para tener más información se puede remitir a la documentación oficial.
- **Node.js y npm instalados**: Angular requiere que tanto Node.js como el Gestor de Paquetes de Node (npm) que estén instalados en su sistema. Si aún no los tiene instalados, a continuación, se presentan los pasos para esta y todas las herramientas que se usan a lo largo de este material.
- **Conocimientos de HTML/CSS:** A lo largo de este material, se trabajará con plantillas HTML y los estilos que provee Bootstrap 5. La familiaridad que el lector tiene con respecto a los conocimientos básicos de las etiquetas HTML y su significado, así como algunos de los conceptos de CSS.

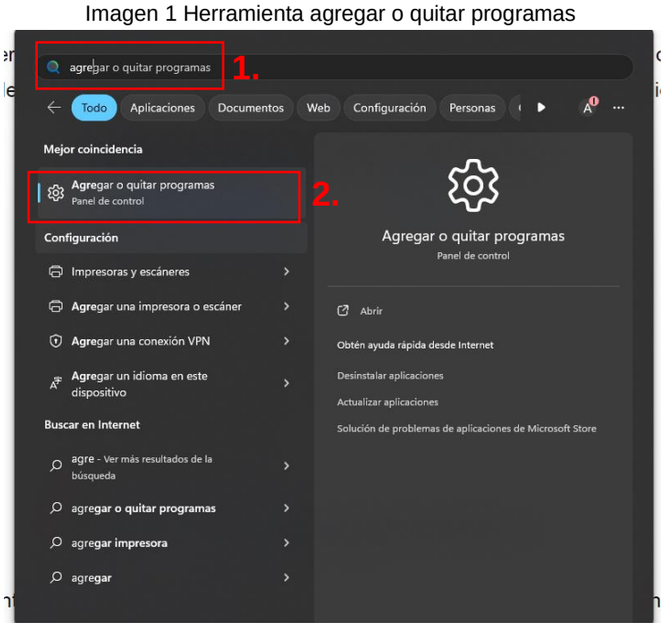
2. Instalación de herramientas

Si ha instalado versiones inferiores a la 16.17.5 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.

2.1. Remover versiones anteriores Node JS

Para remover instalaciones previas de Node JS se utilizará la herramienta agregar o quitar programas de Windows. La Imagen 1 presenta los pasos a seguir para lanzar la aplicación.

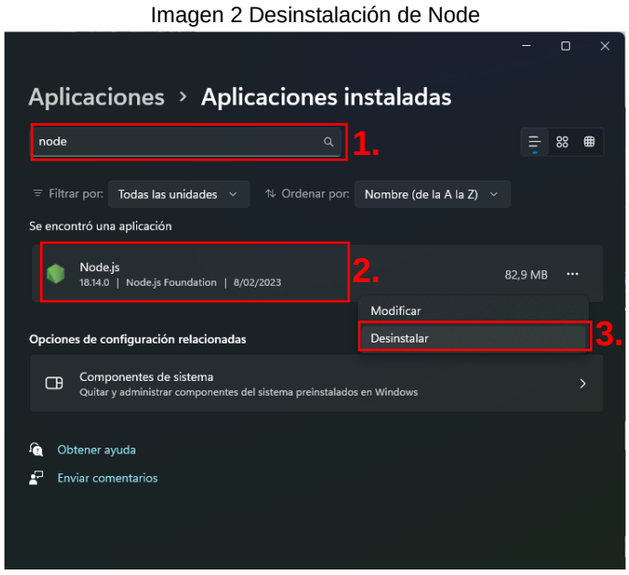
Imagen 1 Herramienta agregar o quitar programas



Fuente Autor.

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 2 presenta la interfaz de la herramienta para eliminar programas.

Imagen 2 Desinstalación de Node

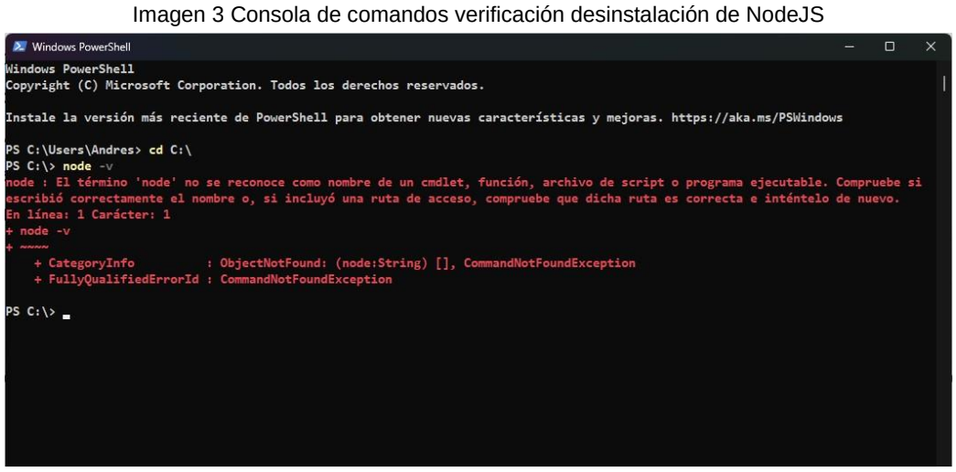


Fuente Autor.

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 3 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 3 Consola de comandos verificación desinstalación de NodeJS

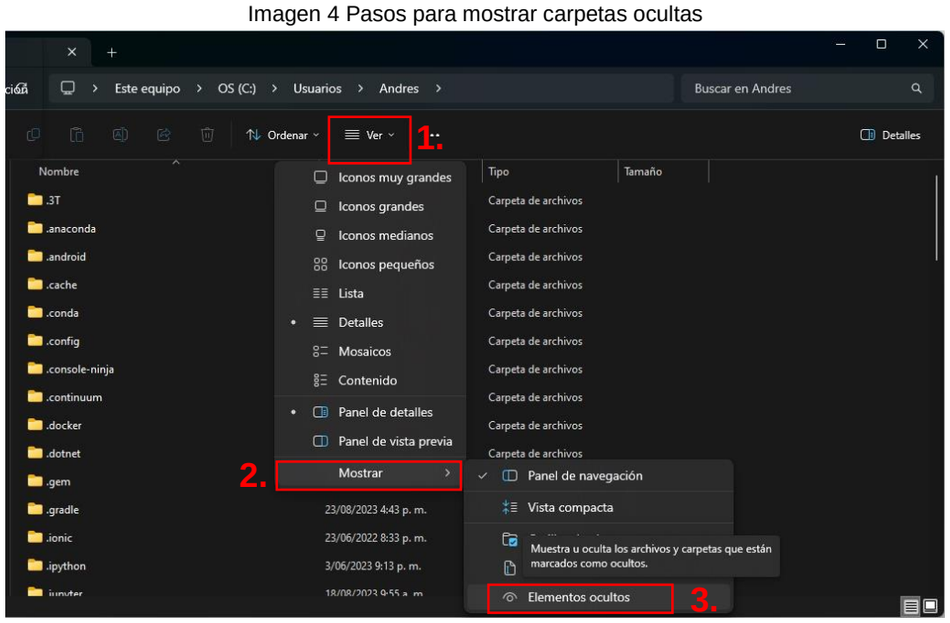


Fuente Autor.

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.

Nota: Pero antes de todo debemos mostrar los elementos y capetas ocultas como se muestra en la Imagen 4.

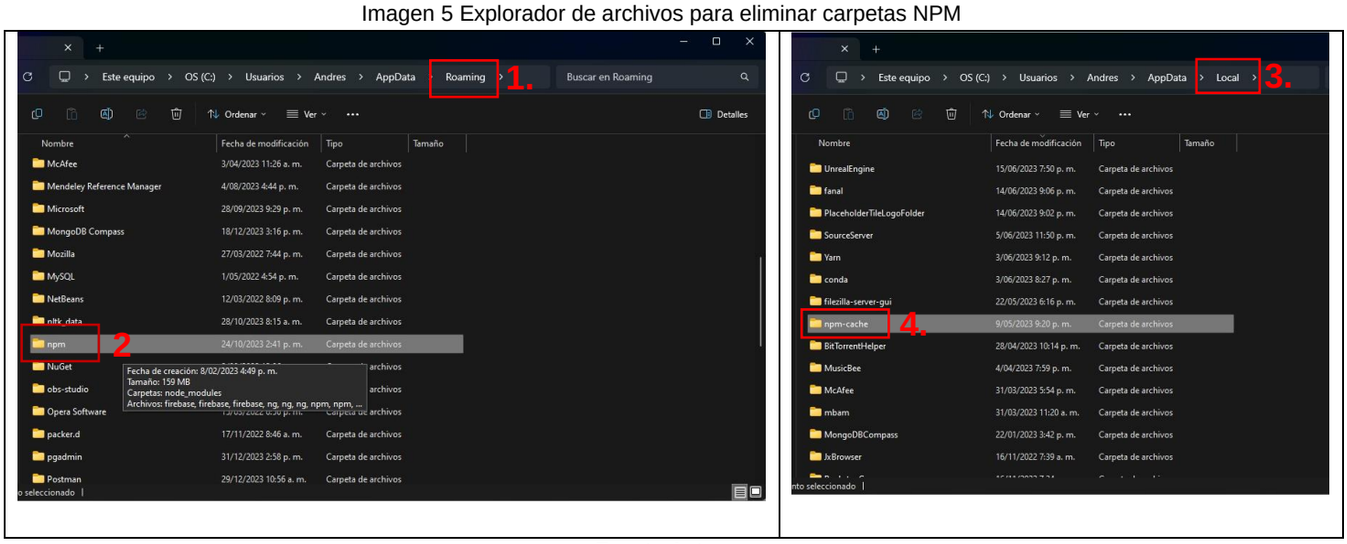
Imagen 4 Pasos para mostrar carpetas ocultas



Fuente Autor.

Esta son las carpetas que debemos eliminar de manera manual.

Imagen 5 Explorador de archivos para eliminar carpetas NPM



Fuente Autor.

2.2. Instalación de Node JS

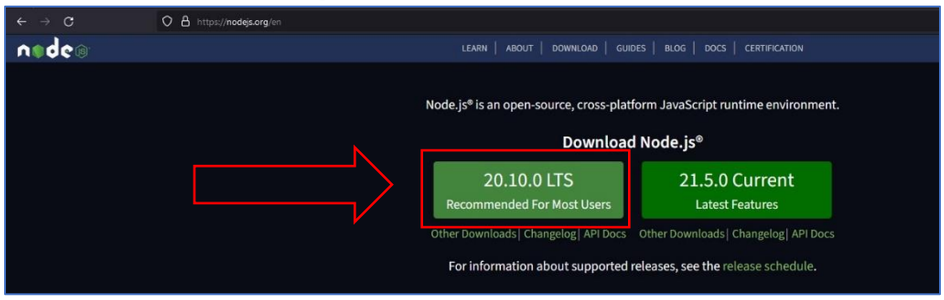
Requisitos para NodeJs: Node.js no requiere una configuración de hardware sofisticada para funcionar; la mayoría de los ordenadores de esta época deberían manejar Node.js de forma eficiente. Incluso los ordenadores más miniaturizados como el BeagleBone o el Arduino YÚN pueden ejecutar Node.js.

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[9].

La potencia de NodeJS 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/ [9]

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



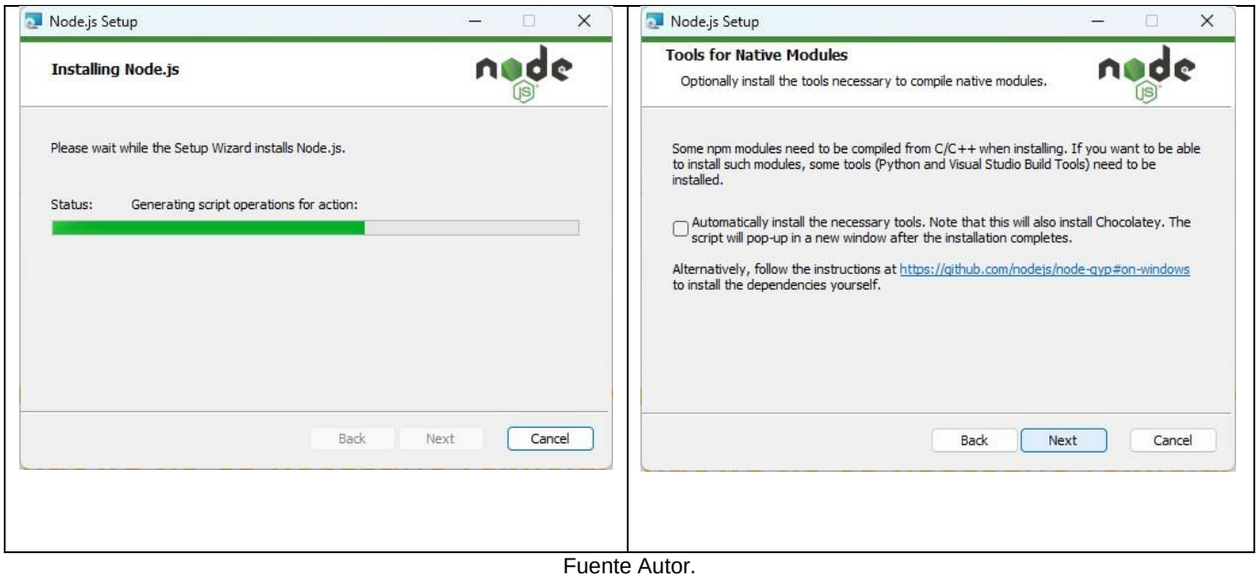
Fuente Autor.

En esta guía se utilizará la versión 20.10.0 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 16.0, 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 20.10.0 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 Autor.

Se recomienda reiniciar el equipo Ya que Windows después de borrar e Instalar NODEJS no actualiza los nuevos cambios.

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 NodeJS y npm

Fuente Autor.

2.3. Instalación de Angular

Requisitos para AngularJs: Para AngularJs es esencial contar con los siguientes requisitos de nuestra maquina:

- Sistema operativo: Windows 10, macOS 10.10 (Yosemite) o posterior, o una distribución reciente de Linux (como Ubuntu 18.04 o posterior)
- Memoria: Al menos 4 GB de RAM
- Espacio de almacenamiento: Al menos 10 GB de espacio libre en disco

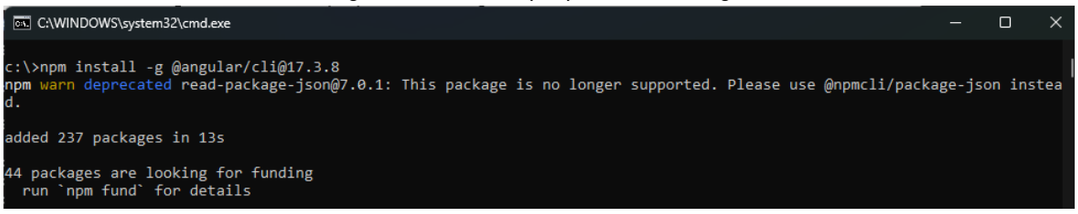
Estos requisitos garantizan que Angular se ejecute sin problemas en nuestro sistema para desarrollar y probar las aplicaciones con eficacia.

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@17.3.8
```

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

Imagen 9. Utilizando npm para instalar Angular



Fuente Autor.

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.

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
```

En caso de tener problemas en la instalación de estas dos herramientas en este Link podrá encontrar una guía de instalación más detallada.

2.4. 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 NodeJS 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.

2.5. Instalación de Visual Studio Code

Requisitos para Visual Studio Code: VS Code es compatible con las siguientes plataformas:

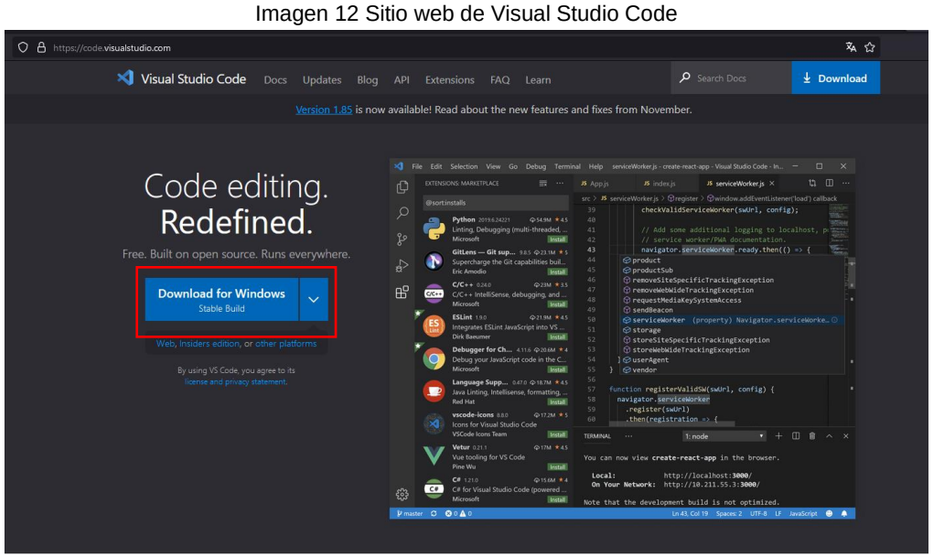
- Windows 10 y 11 (64 bits)
- Versiones de macOS compatibles con actualizaciones de seguridad de Apple. Esta suele ser la última versión y las dos versiones anteriores.
- Linux (Debian): Escritorio Ubuntu 20.04, Debian 10
- Linux (Red Hat): Red Hat Enterprise Linux 8, Fedora 36

Hardware: Visual Studio Code es una descarga pequeña (< 200 MB) y ocupa un espacio en disco menos de 500 MB. VS Code es liviano y debería ejecutarse fácilmente en el hardware actual. Se recomienda:

- Procesador de 1,6 GHz o más rápido
- 1 GB de RAM

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 12 se puede apreciar el sitio web del editor de código seleccionado para el Frontend.

Imagen 12 Sitio web de Visual Studio Code



Fuente Autor.

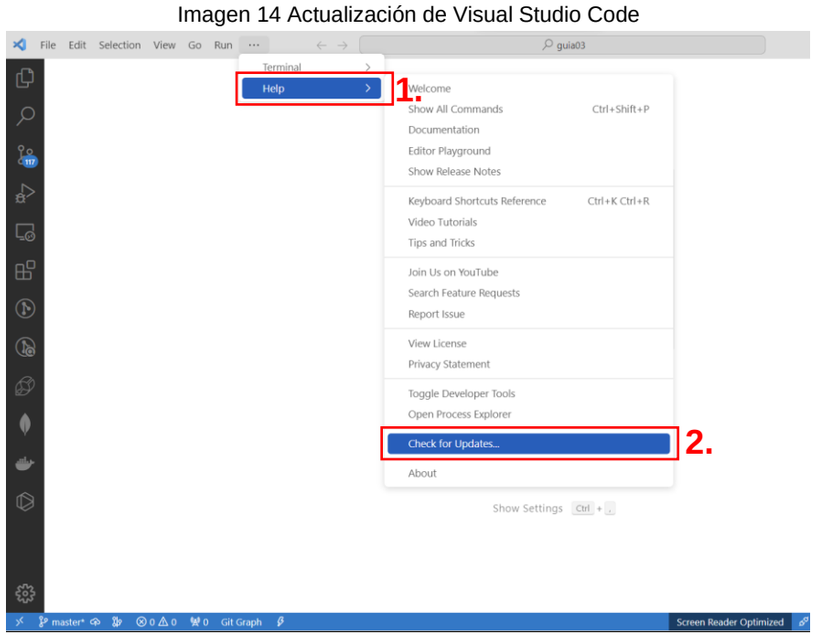
El proceso de instalación es simple, como todo instalador de Windows con varios clics en siguiente, siguiente y finalizar es suficiente. La Imagen 13 presenta la instalación exitosa

Imagen 13 Instalación correcta de Visual Studio Code

Fuente Autor.

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 14 presenta la forma de realizar la actualización en Visual Studio Code y las extensiones más útiles al momento de desarrollar en Angular.

Imagen 14 Actualización de Visual Studio Code

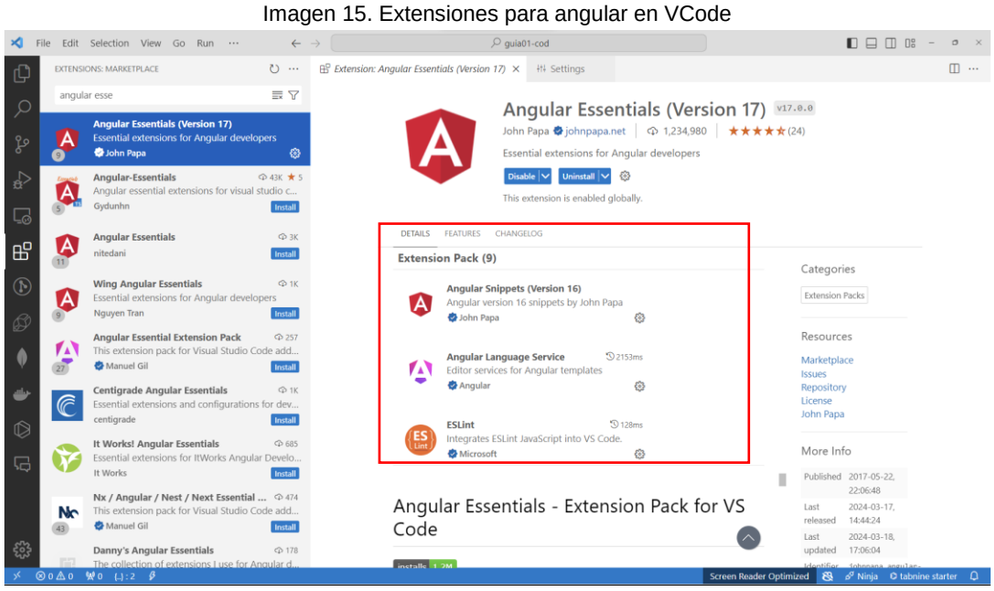


Fuente Autor.

2.6. Extensiones Angular VCode

Como se puede observar en la Imagen 15 Se recomienda instalar el paquete de extensiones llamado "Angular Essentials (Version 17)". Este conjunto de extensiones ha sido diseñado específicamente para potenciar el entorno de desarrollo AngularJS.

Imagen 15. Extensiones para angular en VCode



Fuente: Autor

2.7. Extensión Auto Import

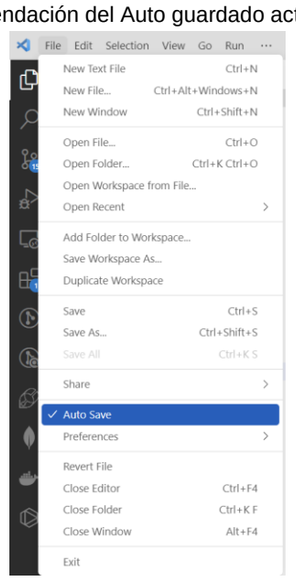
Encuentra, analiza y proporciona automáticamente acciones de código y finalización de código para todas las importaciones disponibles. Funciona con Typecript y TSX.

Imagen 16. Extensión para importación de files

Fuente: Autor

Para optimizar la experiencia de desarrollo en Visual Studio Code, se recomienda activar la función de auto guardado (Auto Save) como se muestra en la Imagen 17.

Imagen 17 Recomendación del Auto guardado activo para Visual Studio Code

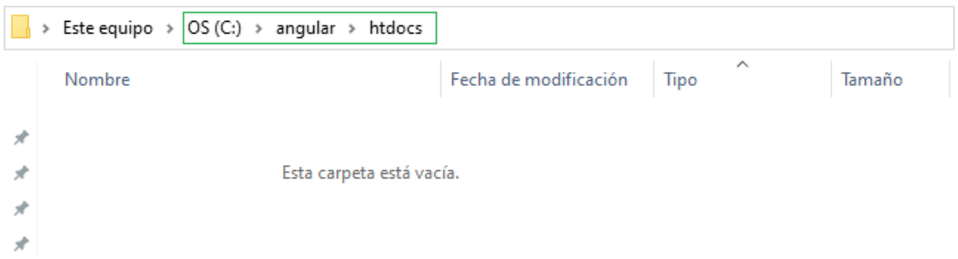


Fuente: Autor

2.8. Estructura de carpetas sugerida

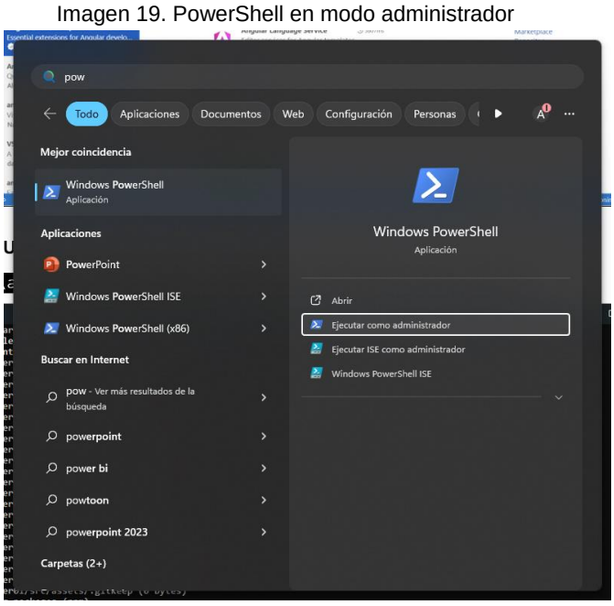
La organización de las carpetas para administrar los proyectos de Angular queda a criterio del desarrollador, sin embargo, a continuación, se propone una estructura de carpetas que permita organizar los proyectos de forma similar a como lo hacen otros lenguajes de programación Web. La Imagen 18 presenta la estructura de carpetas sugerida. La creación de estas carpetas se realiza de manera manual en el explorador de Windows o por la línea de comandos en una terminal.

Imagen 18. Estructura de carpetas



Fuente: Autor

Imagen 19. PowerShell en modo administrador



Fuente: Autor

Imagen 20. Creación de carpetas por comandos

Fuente: Autor

3. 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 hacen del proceso una tarea más eficiente.

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

NodeJS es un entorno de ejecución de JavaScript con una estructura basada en eventos para aplicaciones escalables. NodeJS 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.

3.1. Características de la máquina utilizada en el desarrollo

Ítem	Descripción
Procesador	11th Gen Intel(R) Core (TM) i5-1135G7 @ 2.40GHz
Memoria física total	12 gigas
Espacio disco disponible	480 gigas SSD
Adaptador de pantalla	Intel(R) Iris(R) Xe Graphics

Sistema operativo	Windows 11 version 22H2

Fuente: Autor

4. Repositorios Git con el "*Scaffolding*" de la guía 02

En los siguientes repositorios encontrara la estructura de carpetas, archivos e imágenes que se necesitaran para desarrollar el proyecto.

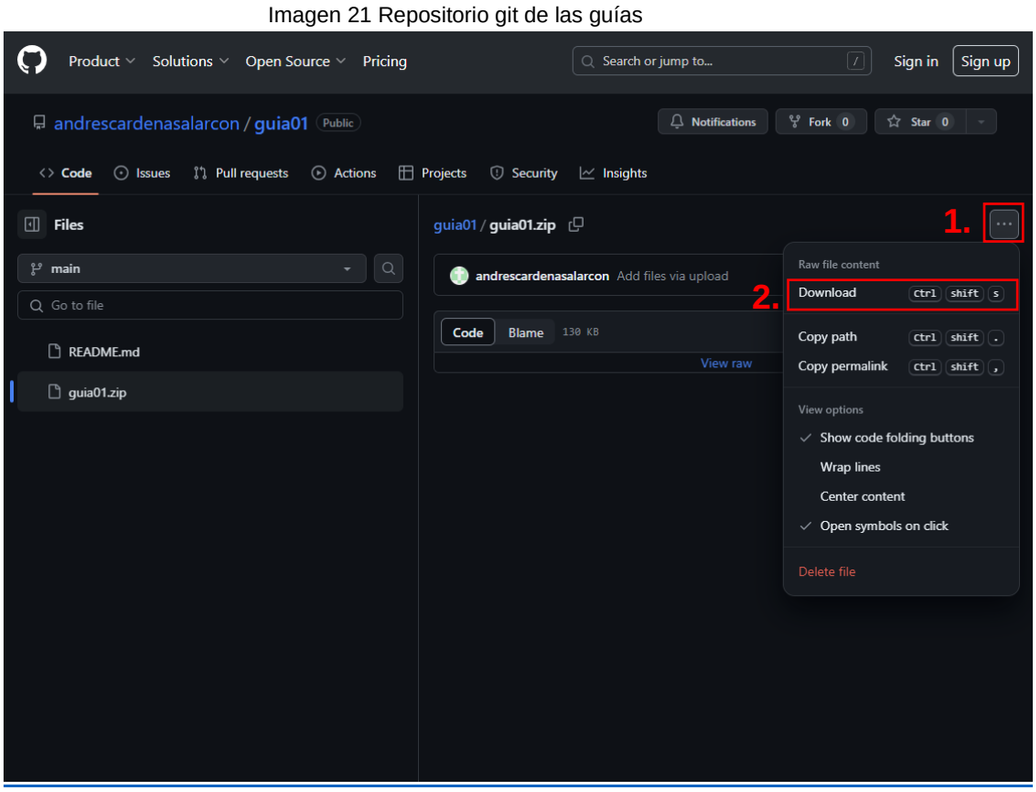
https://github.com/andrescardenasalarcon/guia01/blob/main/guia01.zip
https://github.com/andrescardenasalarcon/guia02/blob/main/guia02.zip
https://github.com/andrescardenasalarcon/guia03/blob/main/guia03.zip

El siguiente proceso aplica para cualquier repositorio, accediendo al enlace de la guía a trabajar.

Pasos:

1. Descargar el proyecto a trabajar como se muestra en la Imagen 21.

Imagen 21 Repositorio git de las guías



Fuente: Autor

2. Descomprimir el archivo zip de la guía, accedemos a la carpeta que abriremos en nuestro editor de código VCode.
3. En la terminal de nuestro proyecto en VCode ejecutaremos el comando "npm i", esto para descargar las librerías node_modules como se muestra en la Imagen 22.

Imagen 22 Instalación de node_modules

Fuente: Autor

4. Empezar a aprender de AngularJs desde este punto [**4.3 Apertura del proyecto en VisualCode**] ✌.

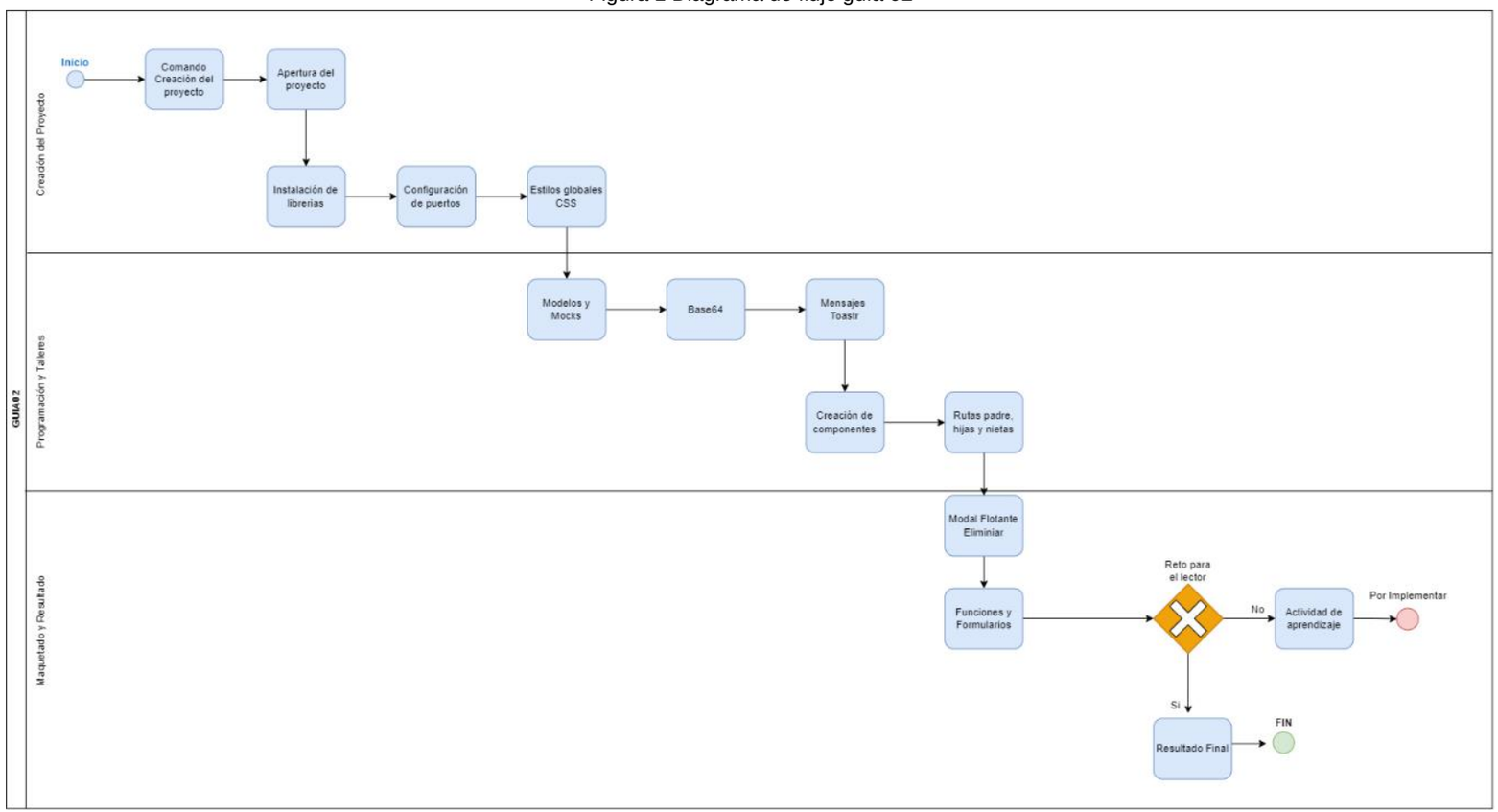
5. Guía 02

Construcción de Talleres Frontend Guía 02	
Problemática	Una empresa fabricante de computadoras ha expresado su interés en desarrollar una aplicación web que le permita crear y configurar computadoras, con distintos periféricos. La aplicación debe permitir la configuración de componentes, incluyendo un monitor con especificaciones de marca y tamaño, así como un ratón y un teclado, ambos con especificaciones idénticas de dispositivo de entrada y marca. En este nuevo caso, se pide que se agregaren compradores y las facturas de venta (las “órdenes de venta” en donde se detalla las características del computador comprado).
Entradas	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.
Proceso	• 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 entre los modelos relacionados. • Taller implementación de ventana de alertas toastr. • Taller implementación de una modal para eliminar.
Resultados	• Imágenes adicionadas a guías de conocimiento

Fuente: Autor

En la Figura 1 se muestra el diagrama de flujo con todos los pasos y las temáticas que se van a abordar a lo largo de esta guía.

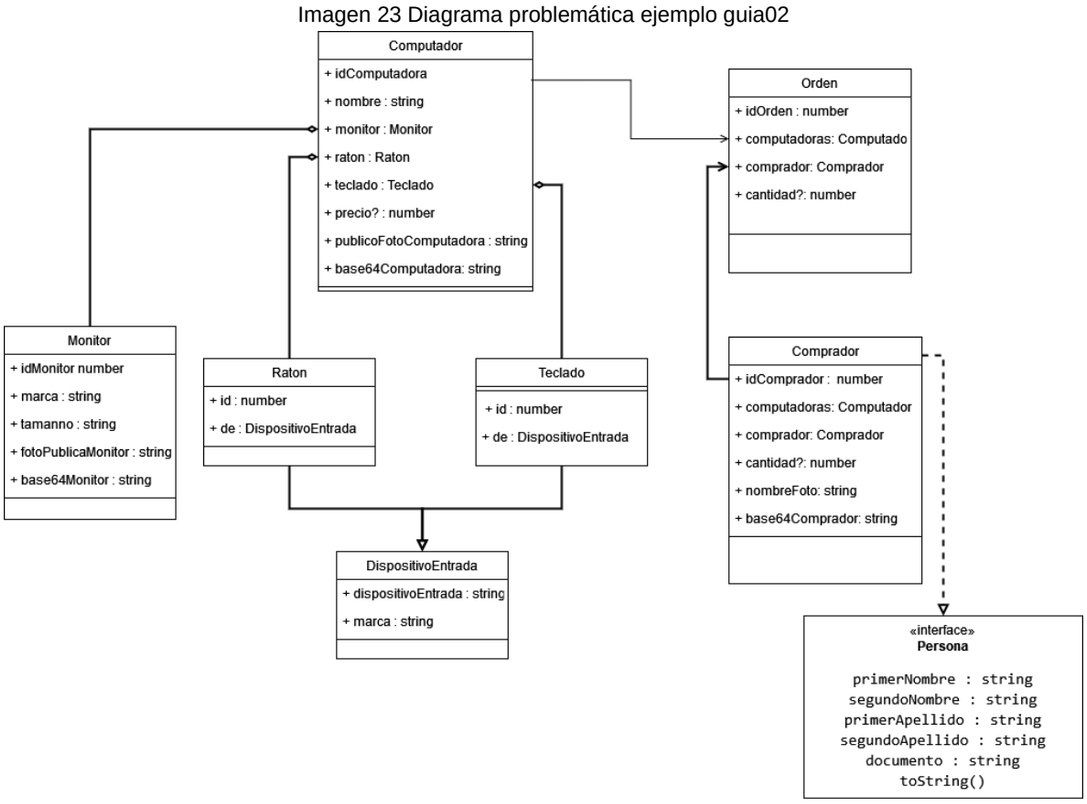
Figura 1 Diagrama de flujo guía 02



Fuente: Autor

Ver la Imagen 23 se ve un diagrama UML con las clases que se van a trabajar a manera de ejemplo para esta guía.

Imagen 23 Diagrama problemática ejemplo guia02



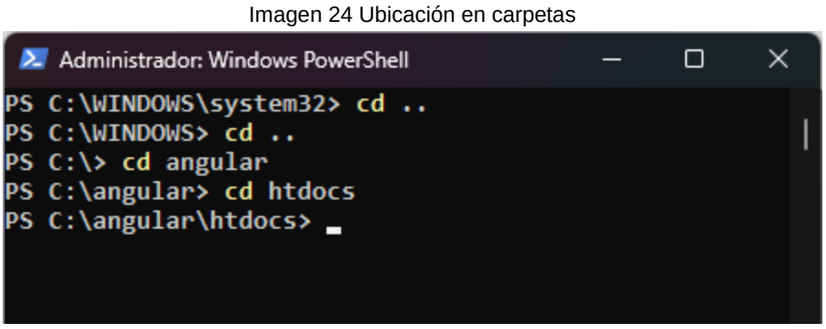
Fuente: Autor

Se **recomienda** descargar y hacer uso del repositorio de GitHub, para el desarrollo de esta guía, ya que ahí se encontrará código y material ya preestablecido que ayudará en el proceso de aprendizaje e implementación de los temas propuestos**. 3. Repositorios Git con el "*Scaffolding*" de la guía 02**

5.1. Ubicación del proyecto para la "Guia02"

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 24 presenta los pasos para cambiar de carpeta.

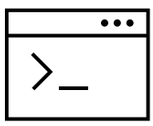
Imagen 24 Ubicación en carpetas



Fuente: Autor

5.2. Creación de un proyecto nuevo

Para crear un proyecto desde cero, mediante el CLI de angular, se sugiere ejecutar el siguiente comando para crear los proyectos.



En una carpeta de ejemplo como "**c:\angular\htdocs**" se debe ejecutar el comando:

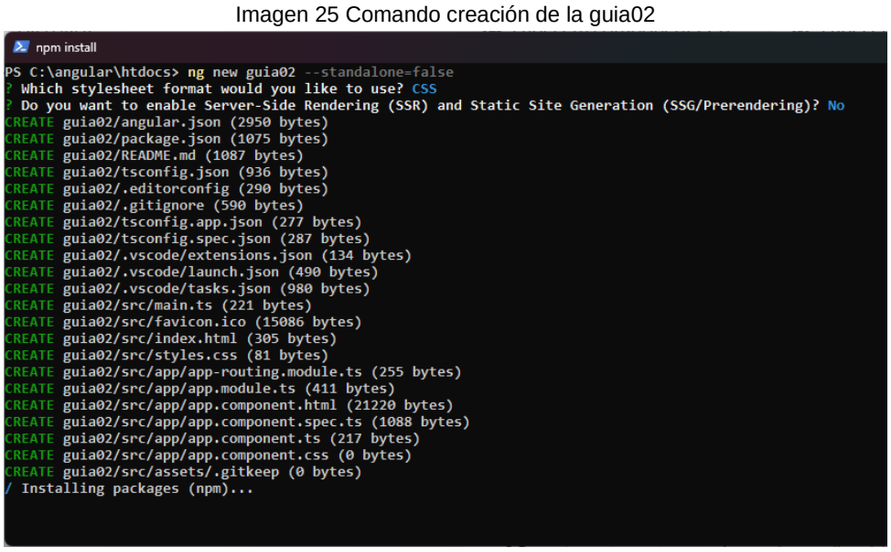
```
C:\angular\htdocs> ng new guia02 --standalone=false
```

Este es el nuevo comando (**en las últimas versiones de angular v17**) para poder crear un proyecto, esto por la estructura que se maneja en este material, se le adiciona el parámetro *--stanalone=false* indica que se debe crear un componente independiente (stanalone component) en lugar de un componente regular.

El comando creará la estructura básica de los archivos, conocida como "**Scaffolding**". La Imagen 25 presenta la ejecución del comando para crear un proyecto en Angular. Al ejecutar el comando *ANGULAR CLI* le hará dos preguntas:

*1) ¿Qué tipo de hoja de estilo se desea utilizar?, se sugiere **CSS**,*
*2) ¿Quiere habilitar el Server-Side Rendering (SSR) y los Sitios de Generación estática (SSG/Prerendering), se sugiere **NO***
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 25 Comando creación de la guia02

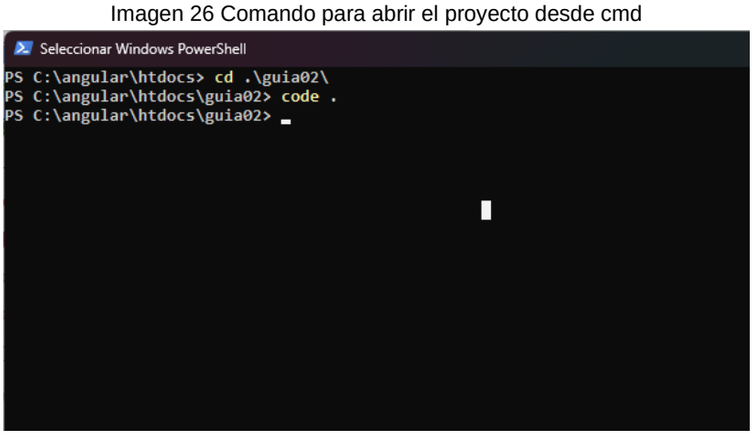


Fuente: Autor

5.3. Apertura del proyecto en VisualCode

También podemos abrir nuestros proyectos creados, accediendo a la carpeta y posteriormente escribiendo "***code .***" como se visualiza en la Imagen 26.

Imagen 26 Comando para abrir el proyecto desde cmd

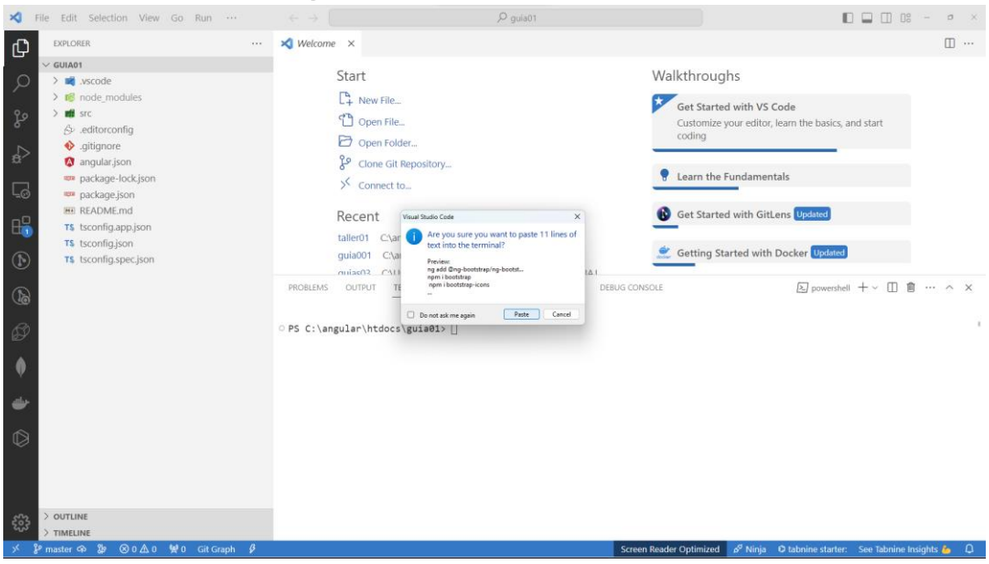


Fuente: Autor

5.4. Instalación de librerías para la guia02

La siguiente Imagen 27 presenta el momento en donde se ejecutan las librerías necesarias para estas guías.

Imagen 27 Selección e instalación de librerías

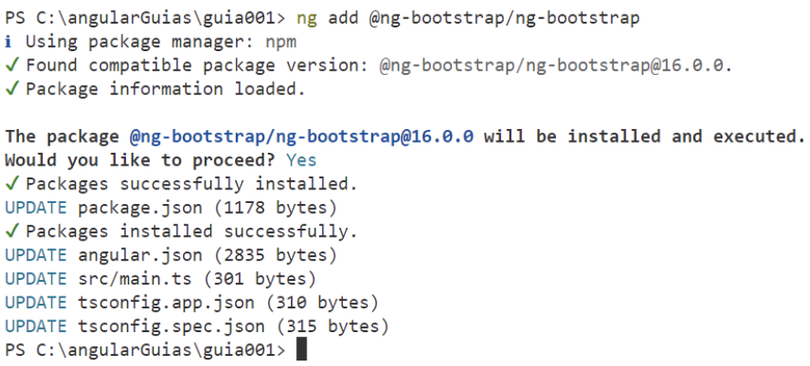


Fuente: Autor

```
ng add @ng-bootstrap/ng-bootstrap
```

El comando ng add @ng-bootstrap/ng-bootstrap se utiliza para agregar la biblioteca ng-bootstrap a un proyecto de Angular. ng-bootstrap es una biblioteca que proporciona componentes de interfaz de usuario basados en Bootstrap para su uso en aplicaciones de Angular.

Imagen 28 Instalación de ng-boostrap

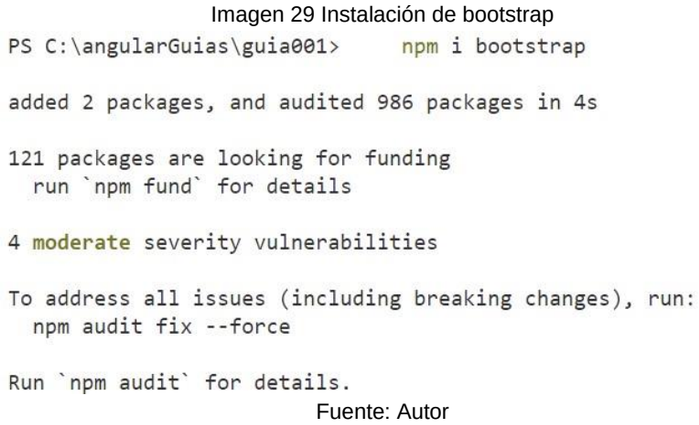


Fuente: Autor

`npm i bootstrap`

El comando npm install bootstrap se utiliza para instalar la biblioteca Bootstrap en un proyecto de Node.js. Bootstrap es un framework de front-end que proporciona componentes y estilos predefinidos para facilitar el desarrollo web.

Imagen 29 Instalación de bootstrap

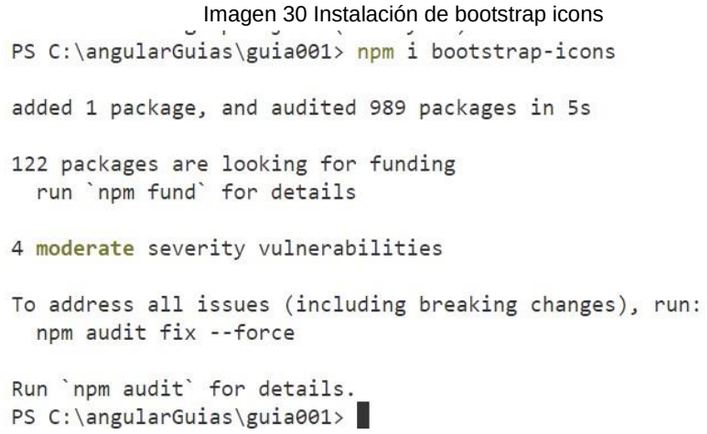


Fuente: Autor

`npm i bootstrap-icons`

El comando npm i bootstrap-icons se utiliza para instalar la biblioteca de iconos Bootstrap Icons en un proyecto. Bootstrap Icons es una biblioteca de iconos SVG de código abierto y oficial para Bootstrap, que cuenta con más de 2,000 iconos disponibles.

Imagen 30 Instalación de bootstrap icons

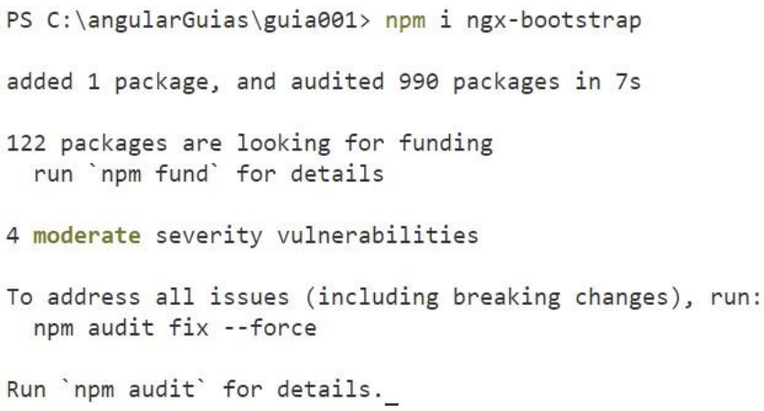


Fuente: Autor

`npm i ngx-bootstrap`

El comando npm i ngx-bootstrap se utiliza para instalar la biblioteca ngx-bootstrap en un proyecto de Angular. ngx-bootstrap es una biblioteca que proporciona componentes de Bootstrap predefinidos y optimizados para su uso en aplicaciones Angular.

Imagen 31 Instalación de ngx-bootstrap

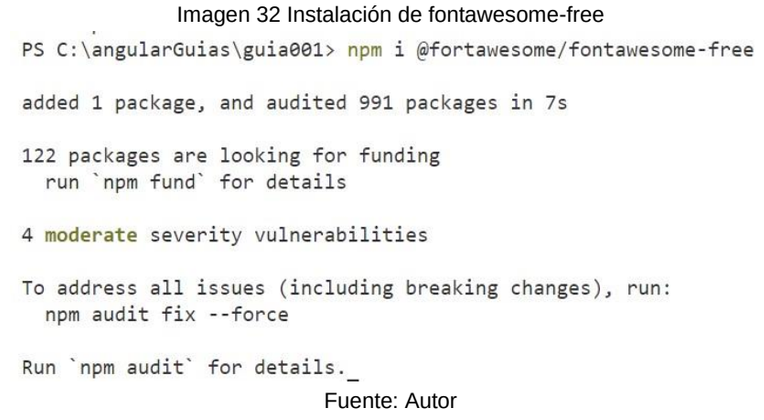


Fuente: Autor

`npm i @fortawesome/fontawesome-free`

El comando npm i @fortawesome/fontawesome-free se utiliza para instalar la biblioteca FontAwesome en un proyecto. FontAwesome es una biblioteca de iconos que proporciona una amplia gama de iconos escalables que se pueden utilizar en aplicaciones web.

Imagen 32 Instalación de fontawesome-free

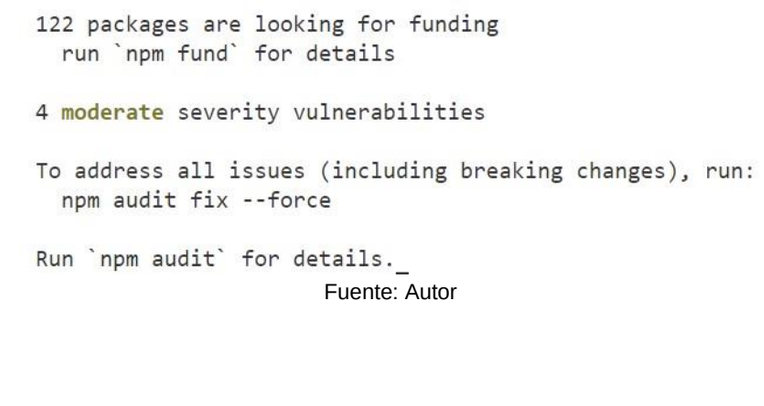


Fuente: Autor

`npm i ngx-toastr`

El comando "***ngx-toastr***" es una librería con componentes que se utiliza para el manejo de las alertas.

Imagen 33 Instalación de ngx-toastr

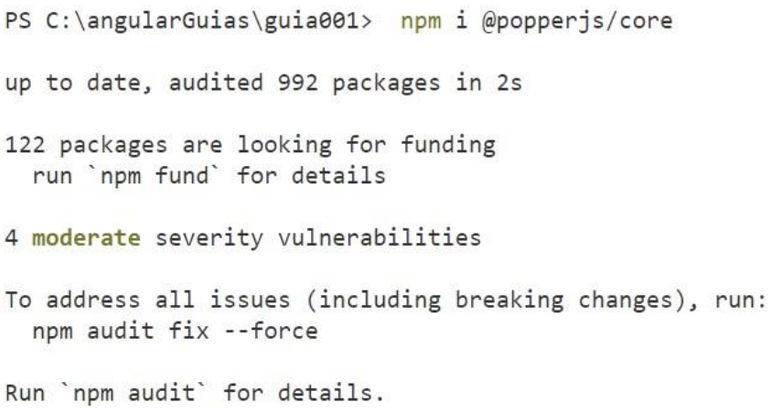


Fuente: Autor

`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.

Imagen 34 Instalación de PopperJs

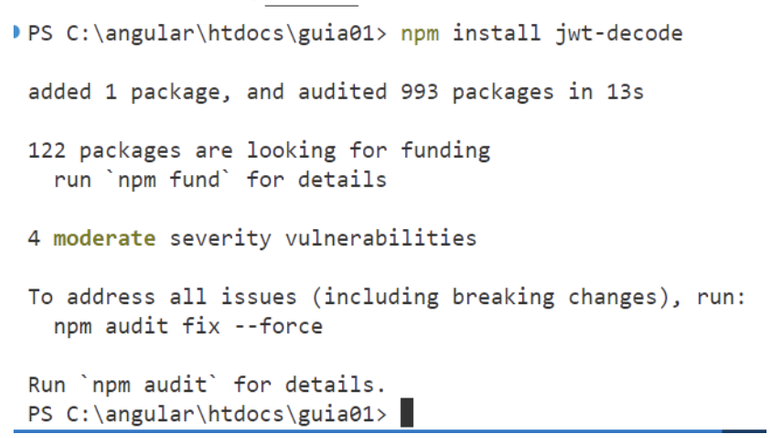


Fuente: Autor

`npm i jwt-decode`

El comando npm i jwt-decode se utiliza para instalar la biblioteca `jwt-decode` en un proyecto. Esta biblioteca permite decodificar tokens JWT (JSON Web Tokens) en aplicaciones JavaScript.

Imagen 35 Instalación de jwt-decode



Fuente: Autor

`npm i crypto-js`

`npm i --save-dev @types/crypto-js`

La librería crypto-js es una biblioteca de JavaScript que proporciona implementaciones de algoritmos criptográficos estándar y seguros.

Imagen 36 Instalación de crypto-js

```
PS C:\angularGuias\guia001> npm install crypto-js

added 1 package, and audited 993 packages in 3s

122 packages are looking for funding
  run `npm fund` for details

4 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Fuente: Autor

Imagen 37 Instalación de @types/crypto-js

```
PS C:\angularGuias\guia001> npm i --save-dev @types/crypto-js

added 1 package, and audited 994 packages in 4s

122 packages are looking for funding
  run `npm fund` for details

4 moderate severity vulnerabilities

To address all issues (including breaking changes), run:
  npm audit fix --force

Run `npm audit` for details.
```

Fuente: Autor

5.5. 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 38 Instalación de paquetes en el módulo principal

```
app.module.ts M
src > app > app.module.ts > AppModule
You, 1 second ago | 1 author (You)
import { NgModule } from '@angular/core';
import { BrowserModule } from '@angular/platform-browser';

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

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



You, 1 second ago | 1 author (You)
@NgModule({
  declarations: [
    AppComponent,
        You, 1 second ago • Uncommitted changes
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,
    NgbModule,
    BrowserModule,
    AppRoutingModule,
    FormsModule,
    ModalModule.forRoot(),
    ToastrModule.forRoot(),
    BrowserAnimationsModule,
  ],
  providers: [],
  bootstrap: [AppComponent]
})
export class AppModule { }
```

Fuente: Autor

Para efectos prácticos se comparte el código en la Tabla 1 para copiarlo y pegarlo en el archivo "***src\app\app.module.ts***"

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

import { AppRoutingModule } from './app-routing.module';
import { AppComponent } from './app.component';

import { NgbModule } from '@ng-bootstrap/ng-bootstrap';
import { FormsModule } from '@angular/forms';
import { ModalModule } from 'ngx-bootstrap/modal';
import { ToastrModule } from 'ngx-toastr';
import { BrowserAnimationsModule } from '@angular/platform-browser/animations';
@NgModule({
  declarations: [
    AppComponent,
  ],
  imports: [
    BrowserModule,
    AppRoutingModule,

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

Tabla 1 Importaciones en app.module.ts

	Un proceso que **NO** es automático es la inclusión de los archivos fuentes o librerias en el proyecto de desarrollo. Se debe editar el archivo "angular.json" e incluir las instrucciones presentadas en la Imagen 39.

Imagen 39 Inclusión de JavaScripts en angulas.json

Fuente: Autor

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

```
      "styles": [
            "node_modules/bootstrap/dist/css/bootstrap.min.css",
            "node_modules/bootstrap-icons/font/bootstrap-icons.css",
            "node_modules/@fortawesome/fontawesome-free/css/all.css",
            "node_modules/ngx-toastr/toastr.css",
            "src/styles.css"
          ],
      "scripts": [
            "node_modules/bootstrap/dist/js/bootstrap.bundle.min.js",
            "node_modules/@popperjs/core/dist/umd/popper-base.min.js"
          ]
```

Tabla 2 Inclusión de JavaScripts en angulas.json

5.6. Asignación de puertos

Para asignar el puerto vamos a ir al archivo "**angular.json**" y en la sección servidor vamos a agregar dos líneas Imagen 40 presenta el cambio de puerto se debe que tener en cuenta.

Imagen 40 Asignación de puertos

Fuente: Autor

En la Imagen 41 se encuentra el bloque de código que se debe agregar para configurar los puertos en modo producción y desarrollo.

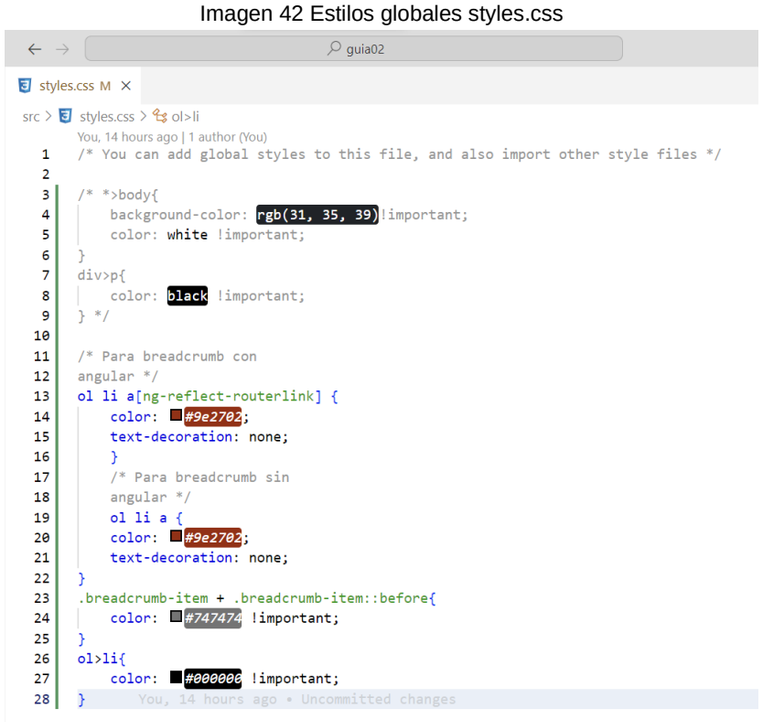
Imagen 41 Asignación de puertos

Fuente: Autor

5.7. 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.css**”. La Imagen 42 Estilos globales styles.css presenta 4 estilos sencillos que pueden ser utilizados en cualquier componente.

Imagen 42 Estilos globales styles.css



Fuente: Autor

Para efectos prácticos, en la Tabla 3 se presenta el código para copiarlo y pegarlo en el archivo “***src\styles.css***”

```
.chequeo {
    font-size: 20px;
    color: #9e2702;
}
/* Para breadcrumb con
angular */
ol li a[ng-reflect-routerlink] {
    color: #9e2702;
    text-decoration: none;
    }
    /* Para breadcrumb sin
    angular */
    ol li a {
    color: #9e2702;
    text-decoration: none;
}
.breadcrumb-item + .breadcrumb-item::before{
    color: #747474 !important;
}
ol>li{
    color: #000000 !important;
}
```

Tabla 3 Estilos globales styles.css

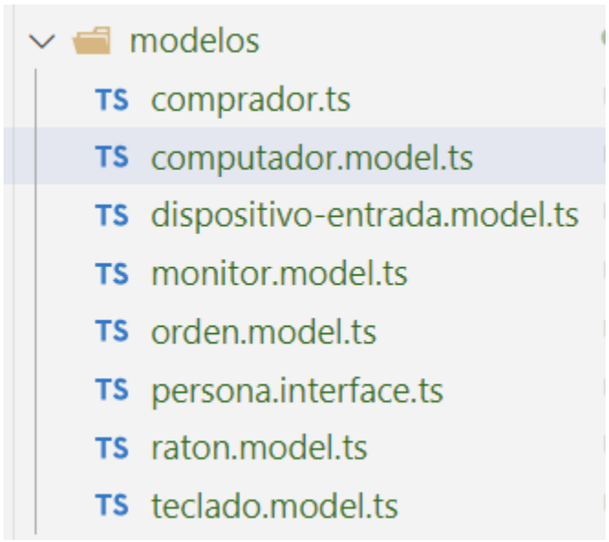
5.8. Modelos

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 Studio Code.

```
ng g class modelos/Comprador --type=model --skip-tests=true
ng g class modelos/Computador --type=model --skip-tests=true
ng g class modelos/DispositivoEntrada --type=model --skip-tests=true
ng g class modelos/Monitor --type=model --skip-tests=true
ng g class modelos/Raton --type=model --skip-tests=true
ng g class modelos/Teclado --type=model --skip-tests=true
ng g class modelos/Orden --type=model --skip-tests=true
ng g interface modelos/persona --type=interface
```

La Imagen 43 presenta la terminal con los comandos a ser ejecutados, la estructura de la carpeta modelos

Imagen 43 Resultado creación de los modelos



Fuente: Autor

En el archivo “**src\app\modelos**” se definen todos los modelos creados a los que posteriormente le asignaremos propiedades que serán inicializadas en el constructor. La Imagen 44 presenta la clase “**DispositivoEntrada**” con sus propiedades y la inicialización de variables en el constructor.

Imagen 44 Clase Dispositivo-entrada de la carpeta modelos

Fuente: Autor

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

```
export class DispositivoEntrada {
    public dispositivoEntrada: string;
    public marca: string;

    constructor(DispositivoEntrada: string, Marca: string) {
        this.dispositivoEntrada = DispositivoEntrada;
        this.marca = Marca;
    }
/* La función `toString` devuelve una representación de cadena de la
marca del objeto y el dispositivo de entrada.*/
    public toString(): string {
        return `Marca: ${this.marca}, Dispositivo de Entrada:
${this.dispositivoEntrada}`;
    }
}
```

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

```
import { DispositivoEntrada } from "./dispositivo-entrada.model";

export class Raton extends DispositivoEntrada {
    public id: number;

    constructor(id: number, DispositivoEntrada: string, Marca: string) {
        super(DispositivoEntrada, Marca);
        this.id = id;
    }

    public override toString(): string {
        return super.toString()
    }
}
```

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

```
import { DispositivoEntrada } from "./dispositivo-entrada.model";

export class Teclado extends DispositivoEntrada {
    public id: number;

    constructor(id: number, DispositivoEntrada: string, Marca: string) {
        super(DispositivoEntrada, Marca);
        this.id = id;
    }

    public override toString(): string {
        return super.toString()
    }
}
```

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

```
export class Monitor{
    public idMonitor: number;
    public marca : string;
    public tamanno : number;
    public fotoPublicaMonitor: string;
    public base64Monitor: string;

    constructor(idRaton: number, marca: string, tamanno: number, fpm: string,
base64: string) {
        this.idMonitor = idRaton;
        this.marca = marca;
        this.tamanno = tamanno;
        this.fotoPublicaMonitor = fpm;
        this.base64Monitor = base64;
    }

    public toString(): string {
        return `Marca: ${this.marca}, Tamaño: ${this.tamanno}''`;
    }
}
```

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

```
import { Monitor } from "./monitor.model";
import { Raton } from "./raton.model";
import { Teclado } from "./teclado.model";

export class Computador {
    public idComputador: number;
    public nombre: string;
    public monitor: Monitor;
    public raton: Raton;
    public teclado: Teclado;
    public publicoFotoComputador: string;
    public base64Computador: string;
    public precio?: number;

    constructor(idComputador: number, nombre: string, monitor: Monitor, raton: Raton,
teclado: Teclado, pubFoto: string, base64: string, precio?: number) {
        this.idComputador = idComputador;
        this.nombre = nombre;
        this.monitor = monitor;
        this.raton = raton;
        this.teclado = teclado;
        this.publicoFotoComputador = pubFoto;
        this.base64Computador = base64;
        if (typeof precio !== "undefined") {
            this.precio = precio;
        }
        this.precio = 1000;
      }

    public getPrice?(): number {
        if (typeof this.precio !== "undefined") {
            return this.precio;
        }
        return 1000;
    }

    public toString(): string {
        return `Nombre: ${this.nombre}, Monitor: ${this.monitor.marca}, Raton:
${this.raton.marca}, Teclado: ${this.teclado.marca}`;
    }
}
```

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

```
/* Define una interfaz llamada `Persona` en TypeScript. Esta interfaz
especifica la estructura de un objeto que debe tener las siguientes
propiedades: */
export interface Persona {
    primerNombre: string;
    segundoNombre: string;
    primerApellido: string;
    segundoApellido: string;
    documento: string;

    toString(): string;
}
```

¡¡¡Bonus!!! En el siguiente blog enseñan acerca de qué es una interfaz y cómo usarlas mediante ejemplos "***Interfaces en Typescript***".

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

```
import { Persona } from "./persona.interface";

export class Comprador implements Persona {
    public idComprador: number;
    public primerNombre: string;
    public segundoNombre: string;
    public primerApellido: string;
    public segundoApellido: string;
    public documento: string;
    public nombreFoto: string;
    public base64Comprador: string;


    constructor(cod: number, primerNombre: string, segundoNombre:
string, primerApellido: string, segundoApellido: string,
        documento: string, nombreFoto: string, fotobase64: string) {
        this.idComprador = cod;
        this.primerNombre = primerNombre;
        this.segundoNombre = segundoNombre;
        this.primerApellido = primerApellido;
        this.segundoApellido = segundoApellido;
        this.documento = documento;
        this.nombreFoto = nombreFoto;
        this.base64Comprador = fotobase64;

    }

    public fullName(): string {
        return `${this.primerNombre} ${this.segundoNombre}
${this.primerApellido} ${this.segundoApellido}`
    }
}
```

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

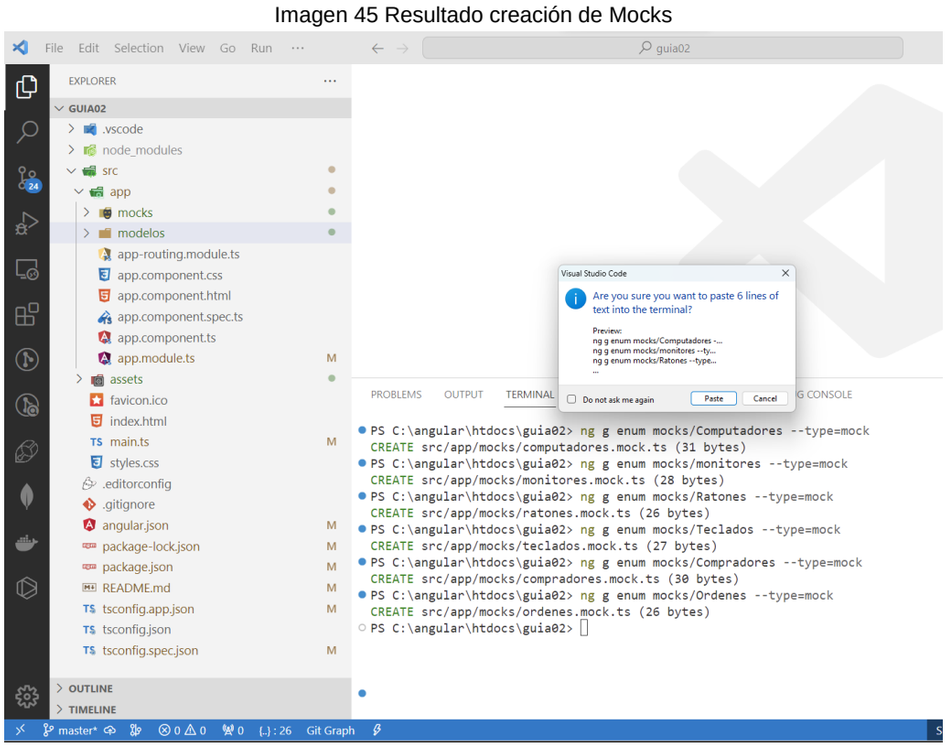
```
/* La clase `Orden` representa un pedido que contiene información sobre
el comprador, las computadoras y cantidad, con un método para calcular el
precio total incluyendo impuestos. */
import { Comprador } from "./comprador.model";
import { Computador } from "./computador.model";

export class Orden {
    public idOrden: number;
    public computadoras: Computador;
    public comprador: Comprador;
    public cantidad: number;

    constructor(idOrden: number, computadoras: Computador, comprador:
Comprador, cantidad?: number) {
        this.idOrden = idOrden;
        this.computadoras = computadoras;
        this.comprador = comprador;
        if (typeof cantidad !== "undefined") {
            this.cantidad = cantidad;
        } else {
            this.cantidad = 1;
        }
    }

/**@returns El método `getTotal` devuelve el precio total de las
computadoras, incluido el impuesto (IVA), * multiplicado por la cantidad
de computadoras, formateado con dos decimales como una cadena.*/
    public getTotal(): string {
        const IVA = 0.19;
        const precioPc: number = this.computadoras.getPrice!();
        return (((precioPc * (IVA + 1)) * this.cantidad)).toFixed(2);
    }
}
```

5.9. Mocks

Con los modelos definidos se procede a generar los mock de prueba. El mock contiene la información para realizar pruebas en el Frontend. La Imagen 45 presenta la ejecución de los comandos en la terminal, la estructura de datos propuestos y el contenido de archivos para cada mock creado con los siguientes comandos:

```
ng g enum mocks/Computadores --type=mock
ng g enum mocks/monitores --type=mock
ng g enum mocks/Ratones --type=mock
ng g enum mocks/Teclados --type=mock
ng g enum mocks/Compradores --type=mock
ng g enum mocks/Ordenes --type=mock
```

Imagen 45 Resultado creación de Mocks



Fuente: Autor

	El mock creado con la “Angular CLI” genera una enum, sin embargo, para los datos de prueba es suficiente con generar un **arreglo** de datos que permita su utilización CRUD en los componentes.

Para efectos prácticos, en la Tabla 4 se comparte el código para copiarlo y pegarlo en el archivo “***src\app\mocks\teclados.mock.ts***”

```
import { Teclado } from "../modelos/teclado.model";

export const ARRAY_TECLADOS: Array<Teclado> = [
    new Teclado(1, "", "Seleccione un teclado"),
    new Teclado(2, "Cable", "Asus"),
    new Teclado(3, "Inalambrico", "Razer"),
    new Teclado(4, "Cable", "Lenovo"),
    new Teclado(5, "Cable - Inalambrico", "Acer"),
    new Teclado(6, "Cable - BT", "Samnsung"),
    new Teclado(7, "Bluetooth", "Razer"),
    new Teclado(8, "USB - Inalambrico", "RedDragon"),
];
```

Tabla 4 Modelo mocks teclado

Para efectos prácticos, en la Tabla 5 se comparte el código para copiarlo y pegarlo en el archivo "***src\app\mocks\ratones.mock.ts***"

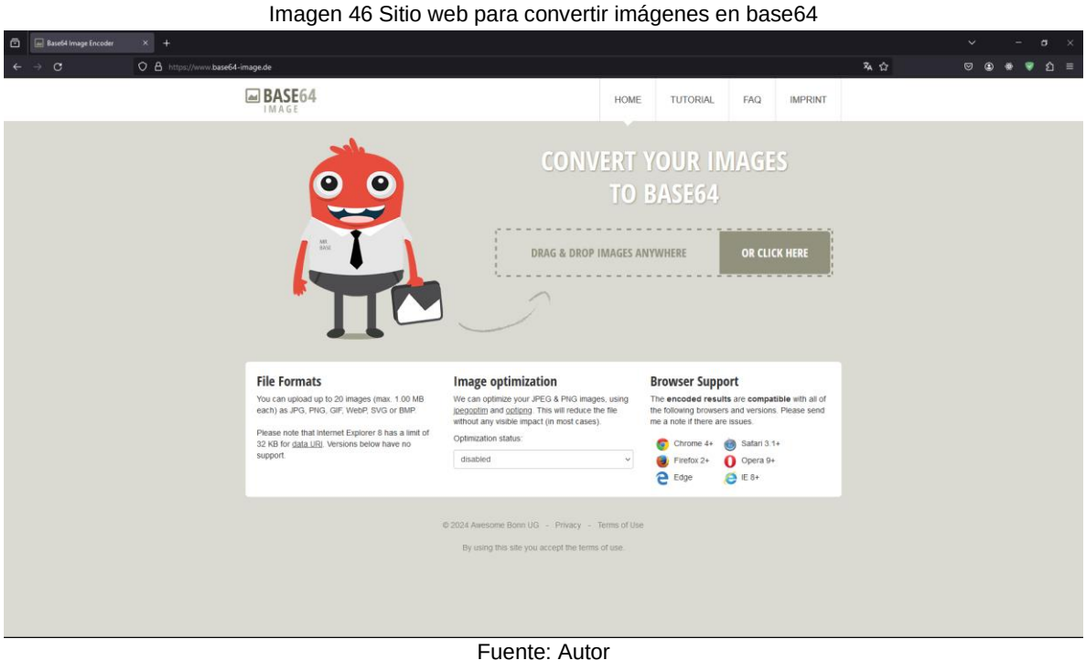
```
import { Raton } from "../modelos/raton.model";

export const ARRAY_RATONES: Array<Raton> = [
    new Raton(1, "", "Seleccione un ratón"),
    new Raton(2, "Cable", " Hewlett-Packard"),
    new Raton(3, "Bluetooth", "Lenovo"),
    new Raton(4, "Cable - Bluetooth", "RedDragon"),
    new Raton(5, "Inalambrico USB", " Hewlett-Packard"),
    new Raton(6, "Bluetooth", "Acer"),
    new Raton(7, "Cable - Bluetooth", "Generico"),
    new Raton(8, "Cable", "Acer"),
]
```

Tabla 5 Modelo mocks ratón

5.10. Codificación Base64 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 46 presenta el sitio web utilizado para procesar las imágenes del taller.

Imagen 46 Sitio web para convertir imágenes en base64

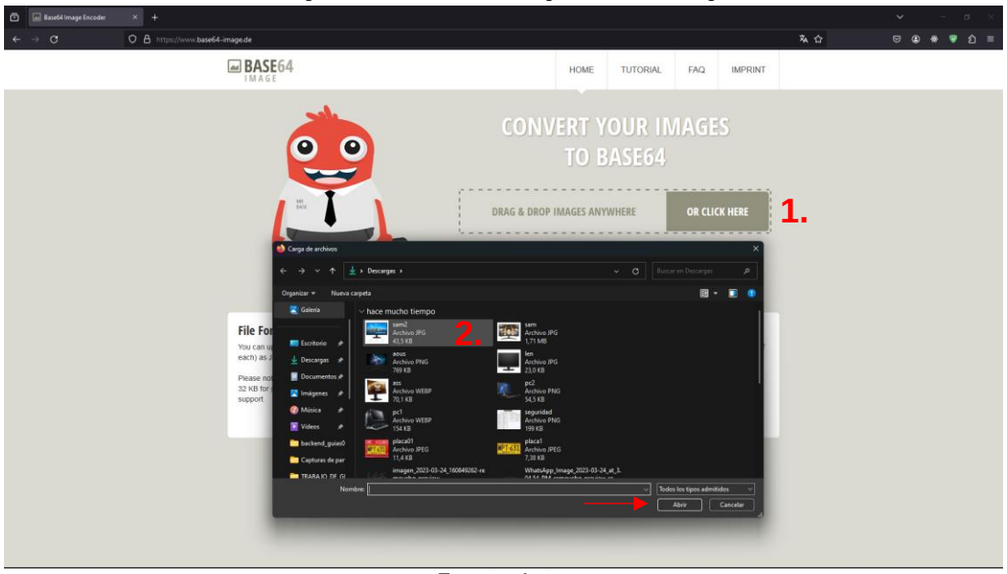


Fuente: Autor

Con el buscador de su preferencia se sugiere que busque una imagen representativa de monitores, computadores y personas para el desarrollo del taller. En este taller la imagen a descargar se almacenará en la carpeta descargas.

Imagen 47 presenta la forma cómo se debe utilizar el sitio web sugerido para obtener la imagen codificada en base 64.

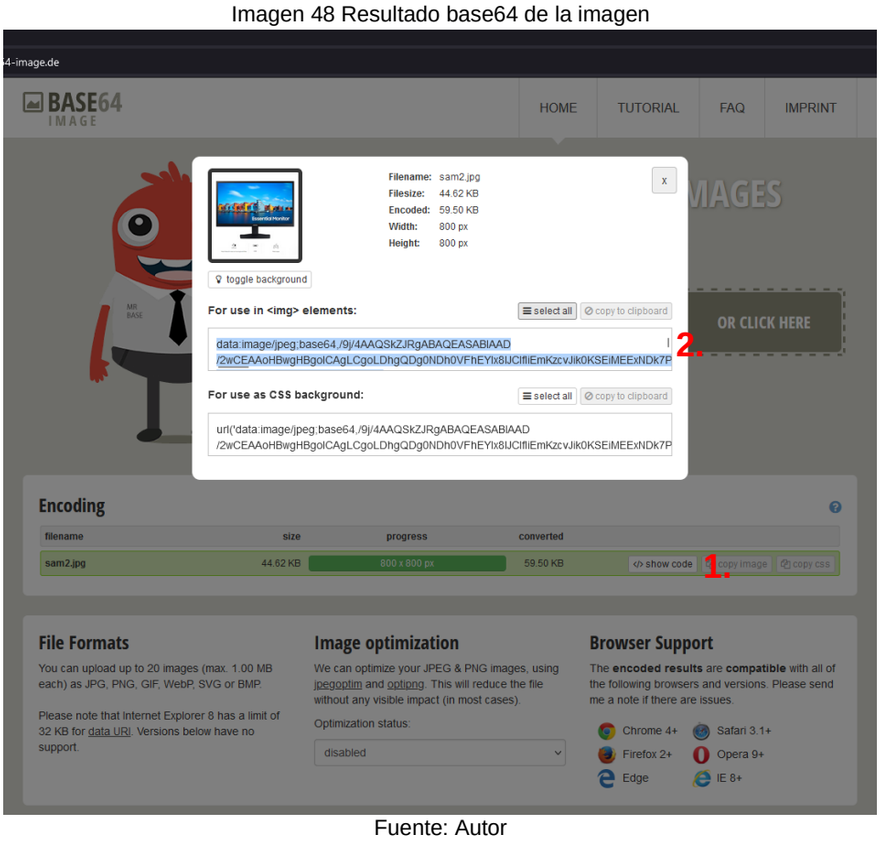
Imagen 47 Selección de imágenes de descargas



Fuente: Autor

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 48 presenta el resultado de la codificación de la imagen de prueba.

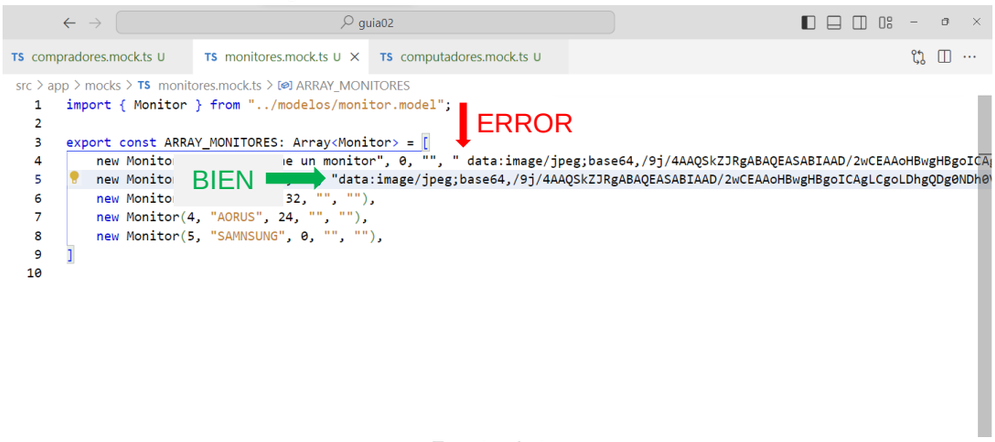
Imagen 48 Resultado base64 de la imagen



Fuente: Autor

Seleccionado el **base64**, se procede a copiar la codificación en el archivo "***src\app\mocks\monitor.mock.ts***" en el **último par de comillas**, así requerido en el modelo como base64. Es clave asegurarse en qué parte del mock se debe agregar la imagen codificada. La Imagen 49 presenta la inclusión de la imagen codificada en el mock.

Imagen 49 Inserción del base 64 de forma correcta



Fuente: Autor

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.

```
import { Monitor } from "../modelos/monitor.model";

export const ARRAY_MONITORES: Array<Monitor> = [
    new Monitor(1, "Seleccione un monitor", 0, "", ""),
    new Monitor(2, "ASUS", 34, "", ""),
    new Monitor(3, "LENOVO", 32, "", ""),
    new Monitor(4, "AORUS", 24, "", ""),
    new Monitor(5, "SAMNSUNG", 0, "", ""),
]
```

También se deben seleccionar imágenes **base64** para el modelo **Computador**, se procede a copiar la codificación en el archivo "***src\app\mocks\computadores.mock.ts***" en el **último par de comillas**, así requerido en el modelo como base64. Es clave asegurarse en qué parte del mock se debe agregar la imagen codificada.

```
import { Computador } from "../modelos/computador.model";
import { ARRAY_MONITORES } from "./monitores.mock";
import { ARRAY_RATONES } from "./ratones.mock";
import { ARRAY_TECLADOS } from "./teclados.mock";

export const ARREGLO_PC: Array<Computador> = [
    new Computador(1, "Gaming", ARRAY_MONITORES[1], ARRAY_RATONES[2],
ARRAY_TECLADOS[2], "", ""),
    new Computador(2, "Oficina", ARRAY_MONITORES[3], ARRAY_RATONES[1],
ARRAY_TECLADOS[7], "", ""),
    new Computador(3, "Empresarial", ARRAY_MONITORES[4], ARRAY_RATONES[6],
ARRAY_TECLADOS[5], "", ""),
    new Computador(4, "Básico", ARRAY_MONITORES[2], ARRAY_RATONES[5],
ARRAY_TECLADOS[4], "", ""),

];
```

También se deben seleccionar imágenes **base64** para el modelo **Comprador**, se procede a copiar la codificación en el archivo "***src\app\mocks\compradores.mock.ts***" en el **último par** de **comillas**, así requerido en el modelo como base64. Es clave asegurarse en qué parte del mock se debe agregar la imagen codificada.

```
import { Comprador } from "../modelos/comprador.model";

export const ARRAY_COMPRADORES: Array<Comprador> = [
    new Comprador(1, "Andres", "Felipe", "Cárdenas", "Alarcon", "10246862", "", ""),
    new Comprador(2, "Juan", "Pablo", "Robles", "Arias", "102487625", "", ""),
    new Comprador(3, "Luis", "Felipe", "Chaparro", "Hurtado", "100245854", "", ""),
    new Comprador(4, "Santiago", "Andrés", "Arias", "Rojo", "40476533", "", ""),
    new Comprador(6, "Brian", "Hernando", "Rodríguez", "Rodríguez", "40476533", "",
""),
    new Comprador(7, "Juan", "Sebastián", "Fonseca", "Fonseca", "40476533", "", ""),

];
```

Finalmente, en el Mock para las Ordenes "***src\app\mocks\ordenes.mock.ts***" **NO** es requerida la inclusión de imágenes en formato Base64. Se deja al final por el hecho de que esta clase es dependiente de clases padres "computador y compradores".

```
import { Orden } from "../modelos/orden.model";
import { ARRAY_COMPRADORES } from "./compradores.mock";
import { ARREGLO_PC } from "./computadores.mock";

export const ARRAY_ORDENES: Array<Orden> = [
    new Orden(0, ARREGLO_PC[0], ARRAY_COMPRADORES[0]),
    new Orden(1, ARREGLO_PC[2], ARRAY_COMPRADORES[0], 2),
    new Orden(2, ARREGLO_PC[3], ARRAY_COMPRADORES[1], 1),
    new Orden(3, ARREGLO_PC[1], ARRAY_COMPRADORES[3], 3),
];
```

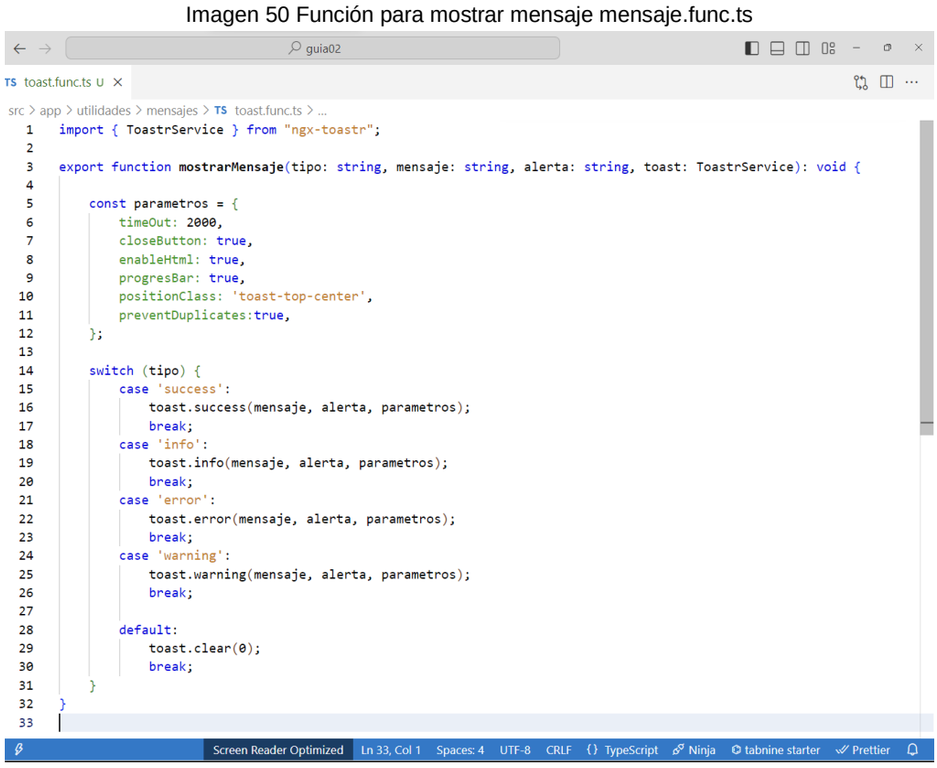
5.11. Mensajes Toastr

Las ventanas toastr son alertas que se mostraran al usuario sobre acciones, información o errores dentro de nuestra aplicación. El comando para crear la funcionalidad de estas alertas es el siguiente:

```
ng g enum utilidades/mensajes/toast --type=func
```

Este comando creará una serie de carpetas dentro de nuestro proyecto "**app/utilidades/mensajes/toast**" como se muestra en la Imagen 50 allí incluiremos la lógica respectiva de esta librería.

Imagen 50 Función para mostrar mensaje mensaje.func.ts



Fuente: Autor

En la Tabla 6 se adjunta el código con la funcionalidad necesaria para mostrar las alertas "**src/app/utilidades/mensajes**"

```
import { ToastrService } from "ngx-toastr";

export function mostrarMensaje(tipo: string, mensaje: string, alerta: string, toast:
ToastrService): void {

    const parametros = {
        timeOut: 2000,
        closeButton: true,
        enableHtml: true,
        progresBar: true,
        positionClass: 'toast-top-center',
        preventDuplicates:true,
    };

    switch (tipo) {
        case 'success':
            toast.success(mensaje, alerta, parametros);
            break;
        case 'info':
            toast.info(mensaje, alerta, parametros);
            break;
        case 'error':
            toast.error(mensaje, alerta, parametros);
            break;
        case 'warning':
            toast.warning(mensaje, alerta, parametros);
            break;

        default:
            toast.clear(0);
            break;
    }
}
```

Tabla 6 Función presentar mensajes

El código en la Tabla 6 para las alertas funciona tomando una serie de parámetros como el tipo de mensaje ('success', 'info', 'error', 'warning'), el mensaje a mostrar, la alerta asociada y el servicio ToastrService. Seguidamente se configuran opciones comunes de visualización de la notificación, como el tiempo de duración, la presencia de un botón de cierre, y la posición en la pantalla.

5.12. Creación de componentes

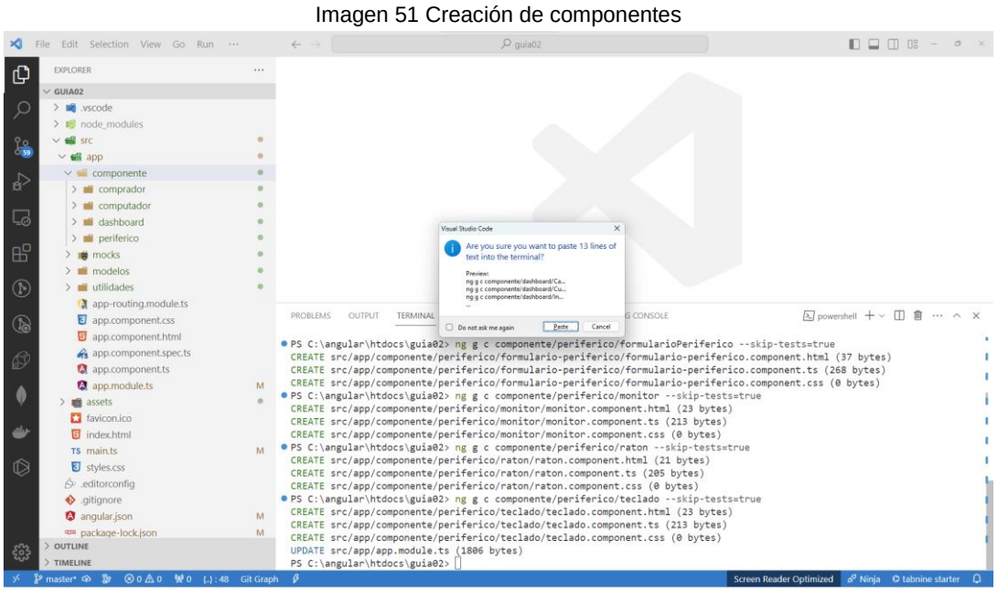
Los componentes son parte esencial de angular, comprender que un componente puede ser una parte de una página, un menú, un pie de página o el mismo contenido, es algo que el desarrollador podrá entender luego de realizar varios ejercicios. Este taller propone la creación de 5 componentes, los cuales distribuiremos la lógica de cada uno de nuestros modelos.

```
ng g c componente/comprador --skip-tests=true
ng g c componente/computador/formularioComputador --skip-tests=true
ng g c componente/computador/tablaComputador --skip-tests=true
ng g c componente/orden --skip-tests=true
ng g c componente/periferico/formularioPeriferico --skip-tests=true
ng g c componente/periferico/monitor --skip-tests=true
ng g c componente/periferico/raton --skip-tests=true
ng g c componente/periferico/teclado --skip-tests=true
ng g c componente/noEncontrado --skip-tests=true
```

De la misma forma en la que se crearon los componentes para nuestros modelos, se procede a ejecutar 3 comandos más, estos necesarios agregar una Cabecera “**header**” un cuerpo “**body**” y una componente bienvenida o “**Inicio**”

```
ng g c dashboard/Cabecera --skip-tests=true
ng g c dashboard/Cuerpo --skip-tests=true
ng g c dashboard/Inicio --skip-tests=true
```

Imagen 51 Creación de componentes



Fuente: Autor

5.13. Globals inits

Ahora, como una buena práctica en esa misma carpeta de “**app/utilidades**” se creará un archivo con las funciones inicializadoras para cada uno de nuestro modelo, esto para limpiar y hacer más entendible nuestro código.

```
ng g enum utilidades/globals/inits --type=func
```

Como se muestra en la Imagen 52 en la ruta “**app/utilidades/globals/inits**” allí incluiremos las instancias necesarias que llamaremos posteriormente.

Imagen 52 Función inicialización de clases

Fuente: Autor

En la siguiente Tabla 7 se adjunta el código con todas las inicializaciones necesarias con las que posteriormente se llamaran en cada uno de nuestros componentes “**app/utilidades/globals/inits**”.

```
import { Comprador } from "../../modelos/comprador.model";
import { Computador } from "../../modelos/computador.model";
import { Monitor } from "../../modelos/monitor.model";
import { Orden } from "../../modelos/orden.model";
import { Raton } from "../../modelos/raton.model";
import { Teclado } from "../../modelos/teclado.model";

export function inicializarMonitor(): Monitor {
    return new Monitor(0, "", 0, "", "");
}

export function inicializarRaton(): Raton {
    return new Raton(0, "", "");
}
export function inicializarTeclado(): Teclado {
    return new Teclado(0, "", "");
}

export function inicializarPc(): Computador {
    return new Computador(0, "", inicializarMonitor(), inicializarRaton(),
inicializarTeclado(), "", "");
}

export function inicializarComprador(): Comprador {
    return new Comprador(0, "", "", "", "", "", "", "");
}

export function inicializarOrden(): Orden {
    return new Orden(0, inicializarPc(), inicializarComprador());
}
```

Tabla 7 Función inicialización de clases

5.14. Etiqueta Router-outlet

La etiqueta para plantillas HTML “**router-outlet**” se utiliza para presentar componentes hijos, esto quiere decir que este taller utilizará la plantilla del componente principal para presentar todos los componentes que se deseen cargar en la aplicación.

En la Tabla 8 se presenta el contenido del archivo “**src\app\app.component.html**”, el cual presenta el código HTML por defecto al crear el componente, todo este código se debe cambiar la etiqueta “router-outlet”.

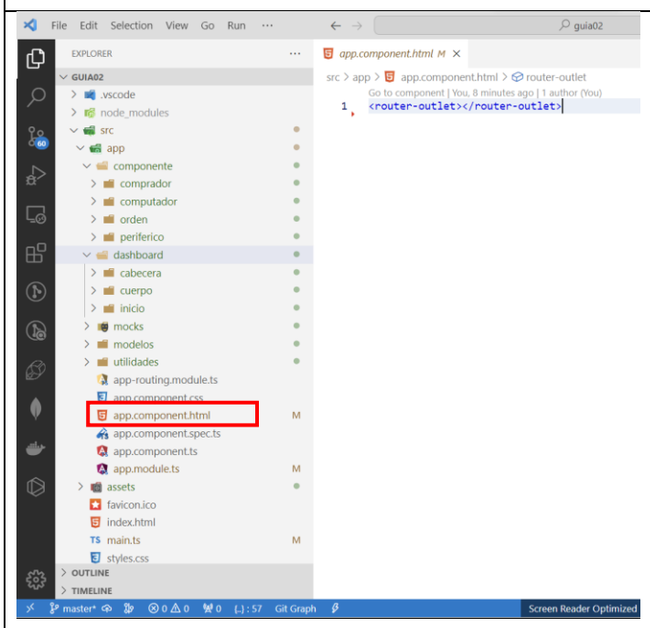
Imagen	Código reutilizable
	`<router-outlet></router-outlet>`

Tabla 8 router-outlet app.component.html

Por defecto, todos los nuevos componentes son hijos del componente “AppComponent”. El objetivo fundamental de la etiqueta “**router-outlet**” es presentar la información sin necesidad de recargar la página, tan solo actualizando los elementos hijos que se desean actualizar.

5.15. Componente Cabecera

En nuestra “**cabecera.component.ts**” previamente creado por nuestro comando, agregaremos un título como se muestra en la Imagen 53.

Imagen 53 Titulo para la cabecera

guia02

cabecera.component.ts U

src > app > dashboard > cabecera > cabecera.component.ts > CabeceraComponent

```
1  import { Component } from '@angular/core';
2
3  @Component({
4    selector: 'app-cabecera',
5    templateUrl: './cabecera.component.html',
6    styleUrl: './cabecera.component.css'
7  })
8  export class CabeceraComponent {
9    public titulo: string;
10   constructor() {
11     this.titulo = 'Rutas padre, hijas y nietas';
12   }
13 }
```

Fuente: Autor

Para efectos prácticos, en la Tabla 9 se comparte el código para copiarlo y pegarlo en el archivo “**src\app\dashboard\cabecera\cabecera.component.ts**”

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

@Component({
  selector: 'app-cabecera',
  templateUrl: './cabecera.component.html',
  styleUrl: './cabecera.component.css'
})
export class CabeceraComponent {
  public titulo: string;
  constructor() {
    this.titulo = 'Rutas padre, hijas y nietas';
  }
}
```

Tabla 9 Titulo para la cabecera

Para efectos prácticos, en la Tabla 10 se comparte el código para copiarlo y pegarlo en el archivo “**app/dashboard/cabecera/cabecera.component.html**”

```
<nav class="navbar navbar-expand-lg navbar-dark bg-dark sticky-top">
  <div class="container">

    <a class="navbar-brand" [routerLink]="['/root']">
      <img
        src="../../../assets/images/angular.png"
        alt="logo"
        style="max-width: 150px"
      />
    </a>
    <i><a class="navbar-text text-decoration-none"
[routerLink]="['/root']">{{ titulo }}</a></i>

    <!-- Collapsed button -->
    <button
      class="navbar-toggler"
      type="button"
      data-bs-toggle="collapse"
      data-bs-target="#navbarSupportedContent"
      aria-controls="navbarSupportedContent"
      aria-expanded="false"
      aria-label="Toggle navigation"
    >
      <span class="navbar-toggler-icon"></span>
    </button>
    <!-- Navbar items -->
    <div
      class="collapse navbar-collapse justify-content-center"
      id="navbarSupportedContent"
    >
      <ul class="navbar-nav">
        <li class="nav-item ml-5">
          <a class="nav-link" [routerLink]="['/root']"
routerLinkActive="active"
            >Inicio</a
          >
        </li>
        <li class="nav-item ml-5">
          <a
            class="nav-link"
            [routerLink]="['/dash/computer']"
            routerLinkActive="active"
            >Computadoras</a
          >
        </li>
```

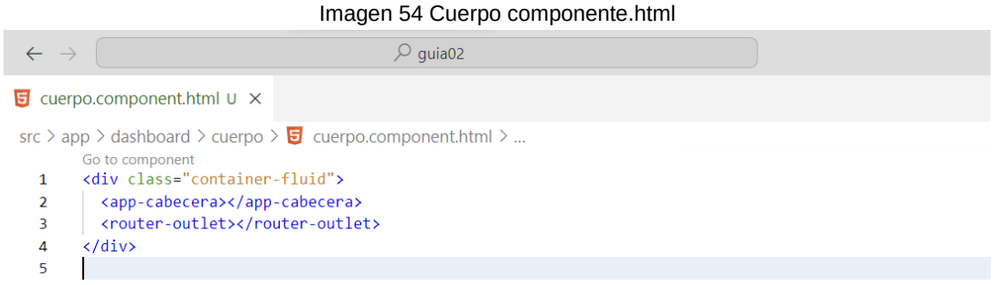
```
        <li class="nav-item ml-5">
          <a
            class="nav-link"
            [routerLink]="['/dash/buyer']"
            routerLinkActive="active"
            >Compradores</a
          >
        </li>
        <!-- Dropdown menu -->
        <li class="nav-item ml-5 dropdown">
          <button
            class="nav-link btn btn-dark dropdown-toggle"
            data-bs-toggle="dropdown"
            aria-expanded="false"
          >
            Perifericos
          </button>
          <!-- Container dropdown menu -->
          <ul class="dropdown-menu dropdown-menu-dark">
            <li>
              <a
                class="dropdown-item"
                [routerLink]="['/dash/monitor']"
                routerLinkActive="active"
                >Monitor</a
              >
            </li>
            <li>
              <a
                class="dropdown-item"
                [routerLink]="['/dash/mouse']"
                routerLinkActive="active"
                >Ratón</a
              >
            </li>
            <li>
              <a
                class="dropdown-item"
                [routerLink]="['/dash/keyboard']"
                routerLinkActive="active"
                >Teclado</a
              >
            </li>
          </ul>
        </li>
      </ul>
    </div>
  </div>
</nav>
```

Tabla 10 Plantilla HTML para la cabecera

`[routerLink]`	Directiva de Angular que se utiliza para crear enlaces de navegación en una aplicación. Se le asigna un valor que especifica la ruta a la que se debe navegar cuando se hace clic en el enlace.

Ahora nos posicionamos en nuestro “**app/dashboard/cuerpo/cuerpo.component.html**” y allí realizaremos un div como se muestra en la Imagen 54 y dentro llamaremos a la componente cabecera, previamente creado, maquetado y que se mantendrá fijado, seguidamente agregaremos `<router-outlet></router-outlet>` este nos renderizará los componentes hijos que llamemos de ahora en adelante.

Imagen 54 Cuerpo componente.html



Fuente: Autor

Tabla 11 muestra el llamado de la componente cabecera y los componentes hijos que se cargarán más adelante.

```
<div class="container-fluid">
  <app-cabecera></app-cabecera>
  <router-outlet></router-outlet>
</div>
```

Tabla 11 Plantilla cuerpo.html

5.16. Componente Inicio

Este componente marca el inicio del proyecto, permite al desarrollador comprender el manejo de una ruta padre “dash” y el redireccionamiento hacía un componente por defecto. En este caso Angular buscará la ruta padre “dash” en el archivo “src\app\app-routing.module.ts” (más adelante se presentará el código para las **rutas**) una vez la encuentra verifica los hijos en este caso la especificada como “/computer” cargará el componente respectivo.

Imagen 55 Componente inicio y router-link

Fuente: Autor

Tabla 12 muestra el maquetado HTML de nuestro inicio, para efectos prácticos se comparte el código para copiarlo y pegarlo en el archivo “**app/dashboard/inicio/inicio.component.html**”

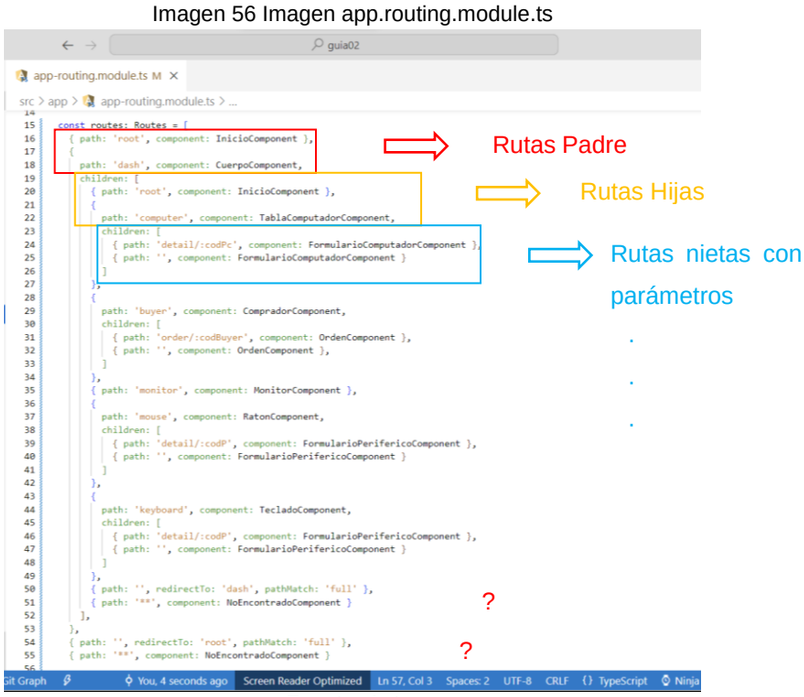
```
<div class="container mt-5">
  <div class="row justify-content-center">
    <div class="col-md-6">
      <div class="p-5 text-white bg-dark rounded-3 bordeado">
        <h1 class="display-6 fw-bold">Navegación Guía02</h1>
        <p style="font-size: 20px">
          <i class="fa fa-check chequeo"></i> Facultad Ingeniería de
          Sistemas
          <br />
          <i class="fa fa-check chequeo"></i> Universidad Santo Tomás
        </p>
        <div class="d-grid">
          <button
            class="btn btn-primary btn-lg"
            type="button"
            [routerLink]="['/dash/computer']"
          >
            Ingresar al taller
          </button>
        </div>
      </div>
    </div>
  </div>
</div>
```

Tabla 12 Plantilla HTML componente inicio.html

5.17. Rutas de tipo padre, hijas y nietas

Las rutas permiten cargar un componente en una etiqueta de tipo “router-outlet” a través de un identificador que en lo posible debería ser único para evitar confusiones. El archivo principal para el manejo de rutas en este taller es “**src\app\app-routing.module.ts**” sin embargo pueden existir muchos más archivos de este tipo en un proyecto. La Imagen 27 presenta el contenido del archivo de rutas completo.

Imagen 56 Imagen app.routing.module.ts



Fuente: Autor

En la Tabla 13 presenta el contenido del “**src\app\app-routing.module.ts**” el cual contiene el ejemplo de rutas de tipo padre, hijas y nietas con parámetros

```
import { NgModule } from '@angular/core';
import { RouterModule, Routes } from '@angular/router';
import { CuerpoComponent } from './dashboard/cuerpo/cuerpo.component';
import { InicioComponent } from './dashboard/inicio/inicio.component';
import { TablaComputadorComponent } from './componente/computador/tabla-computador/tabla-computador.component';
import { FormularioComputadorComponent } from './componente/computador/formulario-computador/formulario-computador.component';
import { CompradorComponent } from './componente/comprador/comprador.component';
import { OrdenComponent } from './componente/orden/orden.component';
import { MonitorComponent } from './componente/periferico/monitor/monitor.component';
import { RatonComponent } from './componente/periferico/raton/raton.component';
import {FormularioPerifericoComponent} from './componente/periferico/formulario-periferico/formulario-periferico.component';
import { TecladoComponent } from './componente/periferico/teclado/teclado.component';
import { NoEncontradoComponent } from './componente/no-encontrado/no-encontrado.component';

const routes: Routes = [
  { path: 'root', component: InicioComponent }, //Ruta raiz '/'
  {
    path: 'dash', component: CuerpoComponent,
    children: [

      { path: 'root', component: InicioComponent }, //Ruta raiz '/' + la cabecera

      {
        path: 'computer', component: TablaComputadorComponent, //Ruta /computador
        children: [ //Rutas hijas del componente computador/detail/:parametro
          { path: 'detail/:codPc', component: FormularioComputadorComponent },
          { path: '', component: FormularioComputadorComponent }
        ]
      },
      {
        path: 'buyer', component: CompradorComponent,
        children: [
          { path: 'order/:codBuyer', component: OrdenComponent },
          { path: '', component: OrdenComponent },
        ]
      },

      { path: 'monitor', component: MonitorComponent },
      {
        path: 'mouse', component: RatonComponent,
        children: [
          { path: 'detail/:codP', component: FormularioPerifericoComponent },
          { path: '', component: FormularioPerifericoComponent }
        ]
      },

      {
        path: 'keyboard', component: TecladoComponent,
        children: [
          { path: 'detail/:codP', component: FormularioPerifericoComponent },
          { path: '', component: FormularioPerifericoComponent }
        ]
      },

      { path: '', redirectTo: 'dash', pathMatch: 'full' },
      { path: '**', component: NoEncontradoComponent }

    ],
  },

  { path: '', redirectTo: 'root', pathMatch: 'full' },
  { path: '**', component: NoEncontradoComponent }

];

@NgModule({
  imports: [RouterModule.forRoot(routes)],
  exports: [RouterModule]
})
export class AppRoutingModule { }
```

Tabla 13 Contenido de src\app\app-routing.module.ts

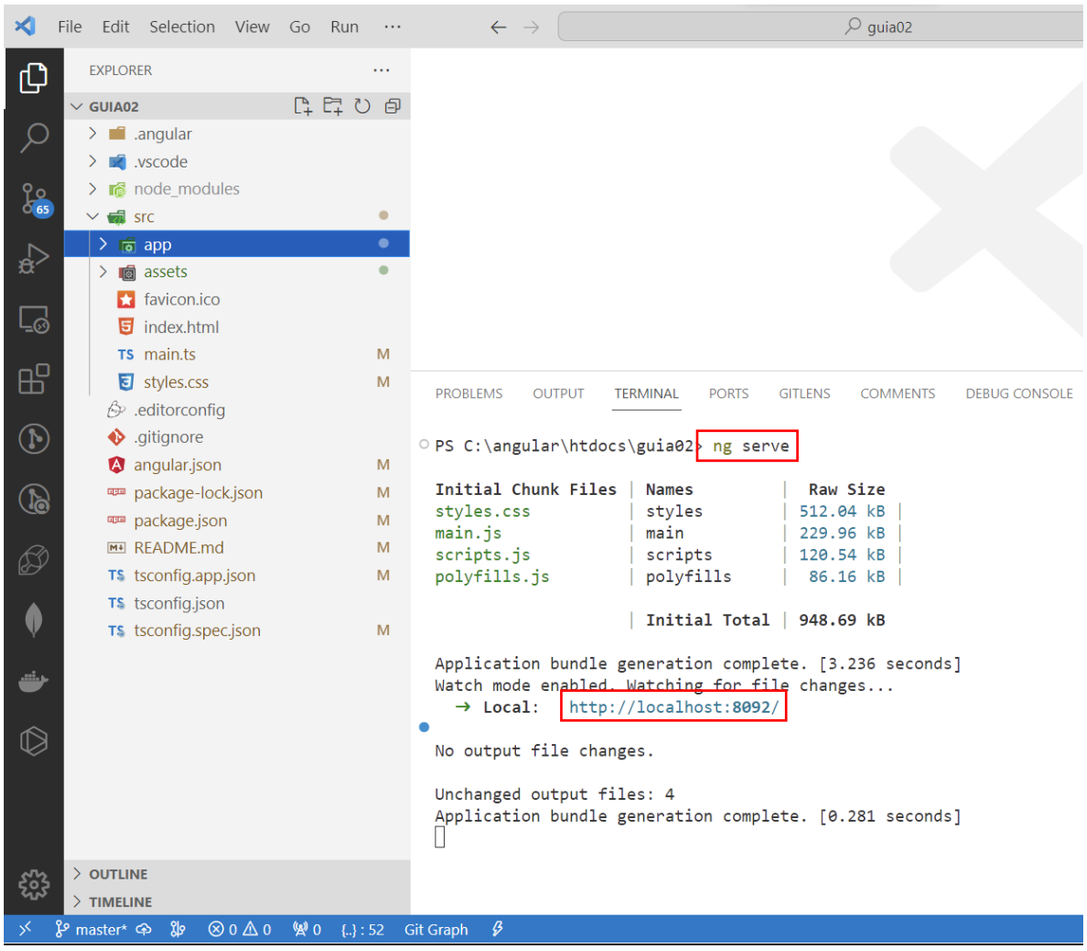
¡¡¡Bonus!!! En el siguiente blog, se presenta una explicación mas a detalle de todo lo relacionado al manejo del archivo de **rutas**, la directiva ***<router-outlet>*** y la etiqueta ***routerLink "Cómo manejar el router de Angular".***

Para verificar el funcionamiento del taller hasta este punto, se sugiere ejecutar en la terminal de visual Studio Code el proyecto con el siguiente comando.

```
ng serve
```

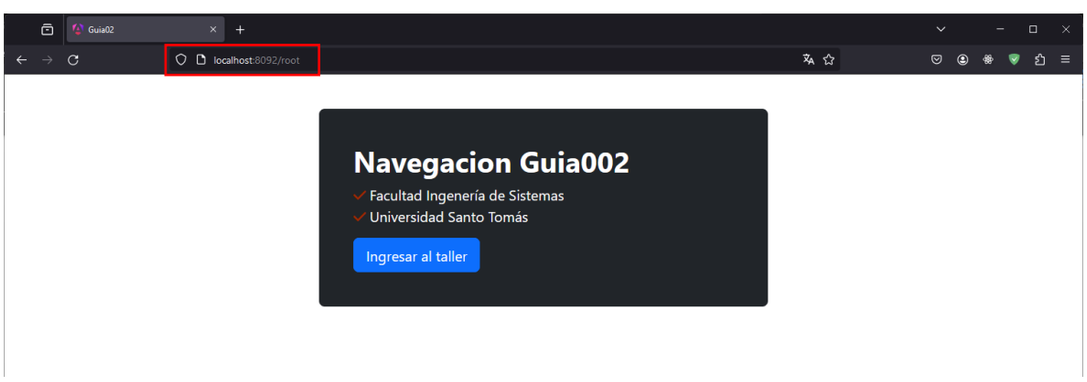
La Imagen 57 presenta la terminal después de la ejecución de comande "ng serve"

Imagen 57 Revisión de avances con ng serve



Fuente: Autor

En el navegador Web de su preferencia podrá visualizar el proyecto.



Fuente: Autor

5.18. Componente Tabla Computador

El componente que presentará funcionalidades de tipo CRUD (Create, Read, Update y Delete) será el componente “TablaComputador”. A continuación, se hará la gestión de la clase del componente. La Imagen 58 presenta el contenido del archivo “src\app\componente\computador\tabla-computador\tabla-computador.component.ts”

Imagen 58 Tabla computador y su lógica respectiva

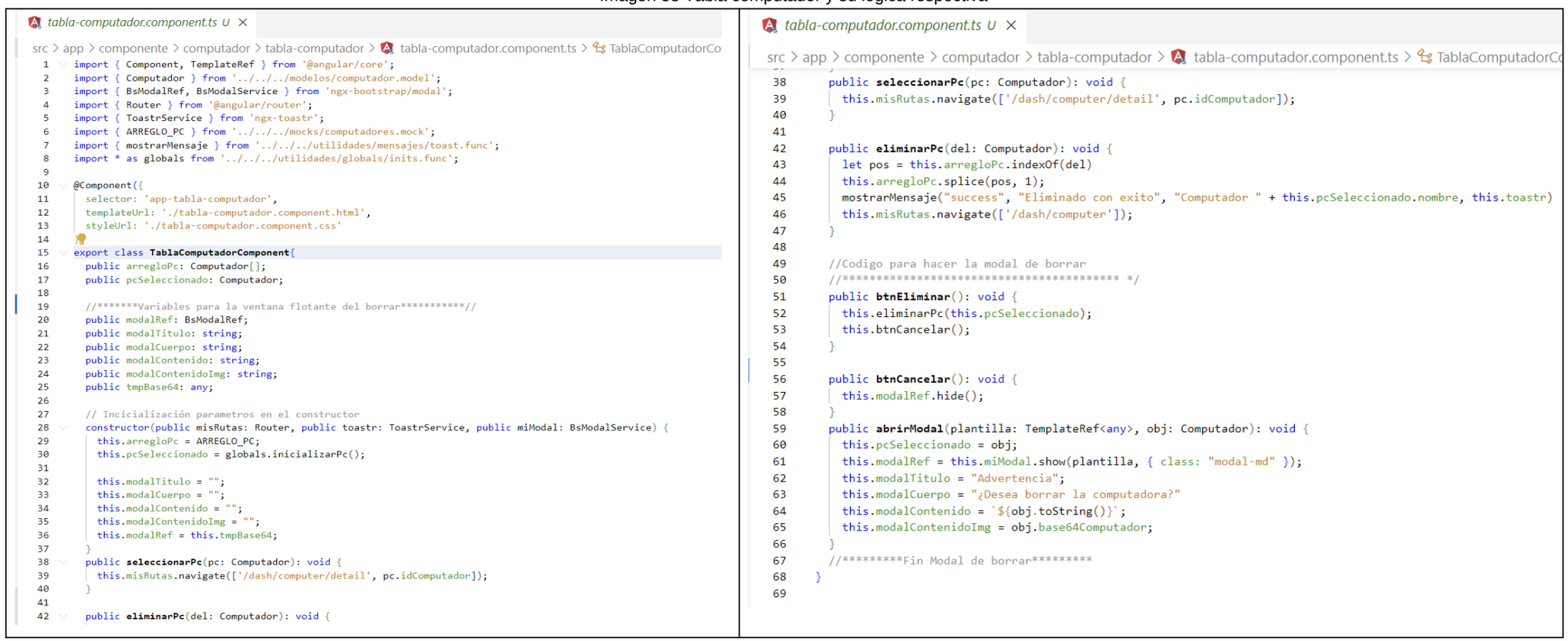


tabla-computador.component.ts U

src > app > componente > computador > tabla-computador > tabla-computador.component.ts > TablaComputadorCo

```ts
import { Component, TemplateRef } from '@angular/core';
import { Computador } from '../../../modelos/computador.model';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';
import { Router } from '@angular/router';
import { ToastrService } from 'ngx-toastr';
import { ARREGLO_PC } from '../../../mocks/computadores.mock';
import { mostrarMensaje } from '../../../utilidades/mensajes/toast.func';
import * as globals from '../../../utilidades/globals/inits.func';

@Component({
  selector: 'app-tabla-computador',
  templateUrl: './tabla-computador.component.html',
  styleUrl: './tabla-computador.component.css'
})
export class TablaComputadorComponent{
  public arregloPc: Computador[];
  public pcSeleccionado: Computador;

  //*******Variables para la ventana flotante del borrar**********//
  public modalRef: BsModalRef;
  public modalTitulo: string;
  public modalCuerpo: string;
  public modalContenido: string;
  public modalContenidoImg: string;
  public tmpBase64: any;

  // Incicialización parametros en el constructor
  constructor(public misRutas: Router, public toastr: ToastrService, public miModal: BsModalService) {
    this.arregloPc = ARREGLO_PC;
    this.pcSeleccionado = globals.inicializarPc();

    this.modalTitulo = "";
    this.modalCuerpo = "";
    this.modalContenido = "";
    this.modalContenidoImg = "";
    this.modalRef = this.tmpBase64;
  }
  public seleccionarPc(pc: Computador): void {
    this.misRutas.navigate(['/dash/computer/detail', pc.idComputador]);
  }

  public eliminarPc(del: Computador): void {
```

tabla-computador.component.ts U

src > app > componente > computador > tabla-computador > tabla-computador.component.ts > TablaComputadorCo

```ts
  public seleccionarPc(pc: Computador): void {
    this.misRutas.navigate(['/dash/computer/detail', pc.idComputador]);
  }

  public eliminarPc(del: Computador): void {
    let pos = this.arregloPc.indexOf(del)
    this.arregloPc.splice(pos, 1);
    mostrarMensaje("success", "Eliminado con exito", "Computador " + this.pcSeleccionado.nombre, this.toastr)
    this.misRutas.navigate(['/dash/computer']);
  }

  //Codigo para hacer la modal de borrar
  //**************************************** */
  public btnEliminar(): void {
    this.eliminarPc(this.pcSeleccionado);
    this.btnCancelar();
  }

  public btnCancelar(): void {
    this.modalRef.hide();
  }
  public abrirModal(plantilla: TemplateRef<any>, obj: Computador): void {
    this.pcSeleccionado = obj;
    this.modalRef = this.miModal.show(plantilla, { class: "modal-md" });
    this.modalTitulo = "Advertencia";
    this.modalCuerpo = "¿Desea borrar la computadora?"
    this.modalContenido = `${obj.toString()}`;
    this.modalContenidoImg = obj.base64Computador;
  }
  //*********Fin Modal de borrar*********
}
```

Fuente: Autor

La Tabla 14 se presenta el código completo de la clase del componente “TablaComputador” que corresponde al archivo “**src\app\componente\computador\tabla-computador\tabla-computador.component.ts**”

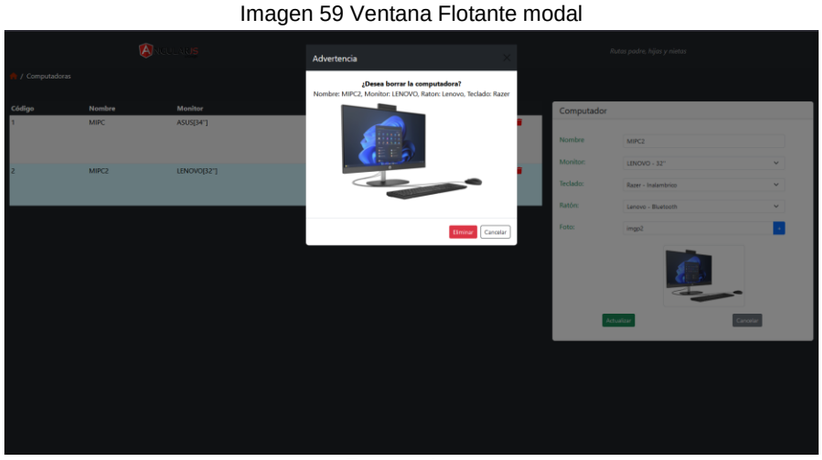
```
import { Component, TemplateRef } from '@angular/core';
import { Computador } from '../../../modelos/computador.model';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';
import { Router } from '@angular/router';
import { ToastrService } from 'ngx-toastr';
import { ARREGLO_PC } from '../../../mocks/computadores.mock';
import { mostrarMensaje } from '../../../utilidades/mensajes/toast.func';
import * as globals from '../../../utilidades/globals/inits.func';

@Component({
  selector: 'app-tabla-computador',
  templateUrl: './tabla-computador.component.html',
  styleUrl: './tabla-computador.component.css'
})

export class TablaComputadorComponent{
  public arregloPc: Computador[];
  public pcSeleccionado: Computador;
  //*******Variables para la ventana flotante del borrar**********//
  public modalRef: BsModalRef;
  public modalTitulo: string;
  public modalCuerpo: string;
  public modalContenido: string;
  public modalContenidoImg: string;
  public tmpBase64: any;

  // Inicialización parámetros en el constructor
  constructor(public misRutas: Router, public toastr: ToastrService, public miModal:
BsModalService) {
    this.arregloPc = ARREGLO_PC;
    this.pcSeleccionado = globals.inicializarPc();

    this.modalTitulo = "";
    this.modalCuerpo = "";
    this.modalContenido = "";
    this.modalContenidoImg = "";
    this.modalRef = this.tmpBase64;
  }
  public seleccionarPc(pc: Computador): void {
    this.misRutas.navigate(['/dash/computer/detail', pc.idComputador]);
  }

  public eliminarPc(del: Computador): void {
    let pos = this.arregloPc.indexOf(del)
    this.arregloPc.splice(pos, 1);
    mostrarMensaje("success", "Eliminado con exito", "Computador " +
this.pcSeleccionado.nombre, this.toastr)
    this.misRutas.navigate(['/dash/computer']);
  }
  //Código para hacer la modal de borrar
  //************************************** */

  //*********Fin Modal de borrar*********
}
```

Tabla 14 Lógica TablaComputadorComponente.ts

5.19. Modal Flotante para eliminar

En la Imagen 59 ilustra un ejemplo de una ventana flotante, en este caso para confirmar la eliminación de un objeto en específico de “**src\app\componente\computador\tabla-computador\tabla-computador.component.ts**”.

Imagen 59 Ventana Flotante modal



Fuente: Autor

La Tabla 15 se presenta el contenido que debe ser agregado entre los **comentarios** `//*********Código para hacer la modal de borrar***` para el funcionamiento de la modal emergente de la “**tabla-computador.component.ts**”.

```
  //*********Código para hacer la modal de borrar*********
  public btnEliminar(): void {
    this.eliminarPc(this.pcSeleccionado);
    this.btnCancelar();
  }

  public btnCancelar(): void {
    this.modalRef.hide();
  }
 /*El parámetro `plantilla` en la función `abrirModal` es de tipo
`TemplateRef<any>`. Esto significa que es una referencia a una
plantilla que se puede usar para representar contenido * dentro del
modal.* @param {Computador} obj - El parámetro `obj` en la función
`abrirModal` parece representar un * objeto de tipo `Computador`. */

  public abrirModal(plantilla: TemplateRef<any>, obj: Computador): void {
    this.pcSeleccionado = obj;
    this.modalRef = this.miModal.show(plantilla, { class: "modal-md" });
    this.modalTitulo = "Advertencia";
    this.modalCuerpo = "¿Desea borrar la computadora?"
    this.modalContenido = `${obj.nombre} $${obj.precio}`;
    this.modalContenidoImg = obj.base64Computador;
  }
  //*********Fin Modal de borrar*********
```

Tabla 15 Modal para borrar

La Tabla 16 presenta el contenido de la plantilla de componente computador, la cual está ubicada en el archivo “**src\app\componente\computador\tabla-computador\tabla-computador.component.html**”

```
<nav class="pt-1" aria-label="breadcrumb">
    <ol class="breadcrumb">
      <li class="breadcrumb-item">
        <a [routerLink]="['/dash/root']" routerLinkActive="router-link-active">
          <i class="fa fa-home"></i>
        </a>
      </li>
      <li class="breadcrumb-item active"[routerLink]="['../computer']" aria-current="page">
        <a href="javascript:void(0)">Computadoras</a>
      </li>
    </ol>
 </nav>
<div class="row mt-lg-5">
  <div class="col-lg-8">
      <!-- TABLA COMPUTADORES -->
    <div class="table-responsive">
      <table class="table table-striped table-hover table-sm my-hover">
          <thead class="bg-dark text-white">
            <tr class="table-dark">
              <th scope="col" style="width: 10px">Código</th>
              <th scope="col" style="width: 30px">Nombre</th>
              <th scope="col" style="width: 15px">Monitor</th>
              <th scope="col" style="width: 15px">Ratón</th>
              <th scope="col" style="width: 20px">Teclado</th>
              <th scope="col" style="width: 20px">Foto</th>
              <th scope="col" style="width: 10px"> </th>
            </tr>
          </thead>
          <tbody>
            <tr
              *ngFor="let miPc of arregloPc"
              (click)="seleccionarPc(miPc)"
              [class.table-info]="miPc == pcSeleccionado"
            >
              <td>{{ miPc.idComputador }}</td>
              <td>{{ miPc.nombre }}</td>
              <td>{{ miPc.monitor.marca }}[{{ miPc.monitor.tamanno }}'']</td>
              <td>{{ miPc.raton.marca }}</td>
              <td>{{ miPc.teclado.marca }}</td>
              <td>
                <img
                onerror="this.src='../assets/images/noFoto.png'"
                [src]="miPc.base64Computador"
                class="img-thumbnail"
                />
              </td>
              <td><i class="fa-solid fa-trash fa-fade" style="color: #ff0000;"
(click)="abrirModal(ventanaModalEliminar, miPc)"></i></td>
            </tr>
          </tbody>
        </table>
    </div>
  </div>
    <div class="col-lg-4">
      <router-outlet></router-outlet>
    </div>
  </div>
```

Tabla 16 Plantilla HTML de la tabla de un computador

5.19.1. Modal Flotante para eliminar en la plantilla HTML

La Tabla 17 Presenta el código que corresponde al maquetado para la visualización de la modal de borrado y esta debe ser **agregada al final** del archivo "***src\app\componente\computador\tabla-computador\tabla-computador.component.html***"

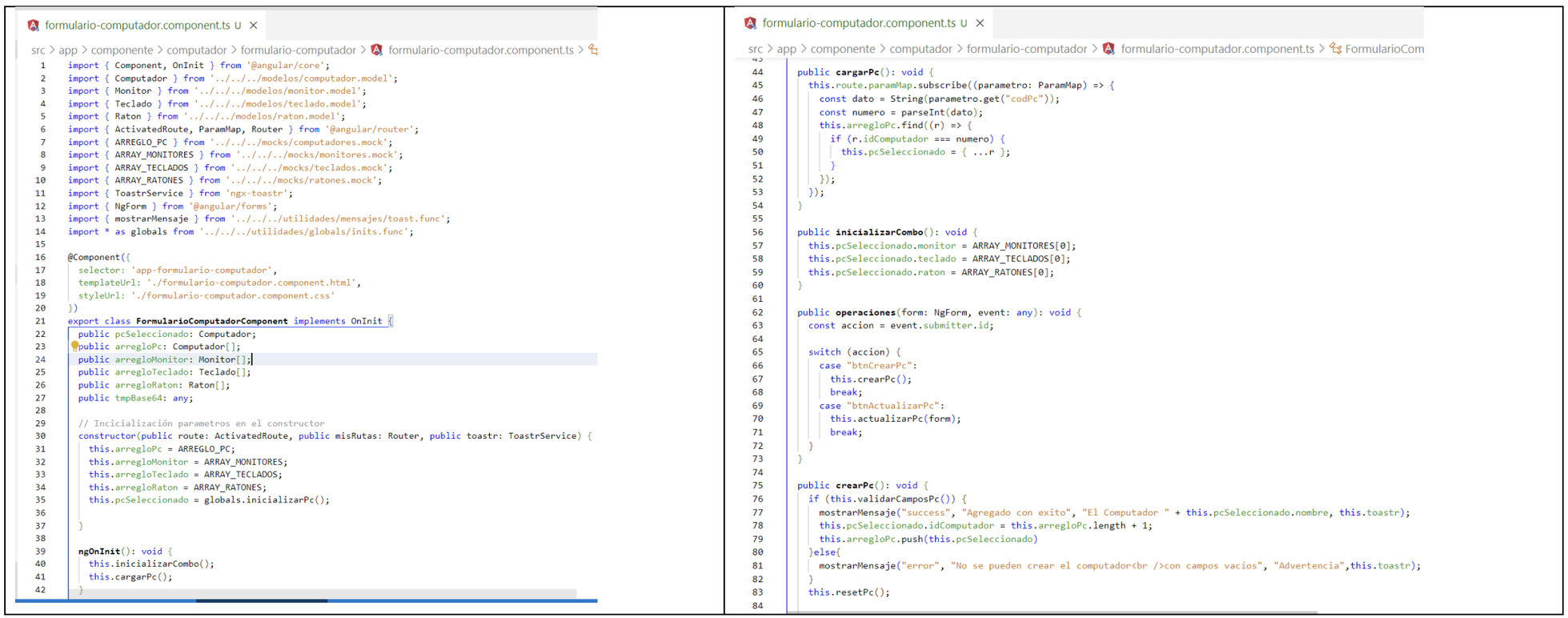
```
 <!--TARJETA MODAL CONFIRMACIÓN AL ELIMINAR -->
  <ng-template #ventanaModalEliminar>
    <div class="modal-header bg-dark text-white"style="border-radius: 2px" >
      <h5 class="modal-title">{{modalTitulo}}</h5>
      <button class="btn btn-close text-white" type="button"
(click)="modalRef.hide()" > </button>
    </div>
    <div class="modal-body d-flex justify-content-center">
      <p class="text-center">
        <strong>{{modalCuerpo}}</strong> <br/>
        {{modalContenido}}
        <img
              [src]="'data:image/png;base64' + modalContenidoImg"
              onerror="this.src='../assets/images/noFoto.png'"
              class="img-fluid"
              style="
                height: 250px;
                width: auto;
                display: block;
                margin-left: auto;
                margin-right: auto;
              "
            />
      </p>
    </div>
    <div class="modal-footer">
      <button class="btn btn-danger btn-sm" type="button"
(click)="btnEliminar()">Eliminar</button>
      <button class="btn btn-outline-dark btn-sm" type="button"
(click)="btnCancelar()">Cancelar</button>
    </div>
  </ng-template>
```

Tabla 17 Plantilla HTML de la modal de borrado

5.20. Componente Formulario Computador

El componente que presentará funcionalidades de tipo CRUD (Create, Read, Update y Delete) será el componente "FormularioComputador". A continuación, se hará la gestión de la clase del componente. La Imagen 60 presenta el contenido del archivo ***"src\app\componente\computador\formulario-computador\formulario-computador.component.ts"***

Imagen 60 Formulario CURD para la clase computador



formulario-computador.component.ts U

src > app > componente > computador > formulario-computador > formulario-computador.component.ts

```ts
import { Component, OnInit } from '@angular/core';
import { Computador } from '../../../modelos/computador.model';
import { Monitor } from '../../../modelos/monitor.model';
import { Teclado } from '../../../modelos/teclado.model';
import { Raton } from '../../../modelos/raton.model';
import { ActivatedRoute, ParamMap, Router } from '@angular/router';
import { ARREGLO_PC } from '../../../mocks/computadores.mock';
import { ARRAY_MONITORES } from '../../../mocks/monitores.mock';
import { ARRAY_TECLADOS } from '../../../mocks/teclados.mock';
import { ARRAY_RATONES } from '../../../mocks/ratones.mock';
import { ToastrService } from 'ngx-toastr';
import { NgForm } from '@angular/forms';
import { mostrarMensaje } from '../../../utilidades/mensajes/toast.func';
import * as globals from '../../../utilidades/globals/inits.func';

@Component({
  selector: 'app-formulario-computador',
  templateUrl: './formulario-computador.component.html',
  styleUrl: './formulario-computador.component.css'
})
export class FormularioComputadorComponent implements OnInit {
  public pcSeleccionado: Computador;
  public arregloPc: Computador[];
  public arregloMonitor: Monitor[];
  public arregloTeclado: Teclado[];
  public arregloRaton: Raton[];
  public tmpBase64: any;

  // Inicialización parametros en el constructor
  constructor(public route: ActivatedRoute, public misRutas: Router, public toastr: ToastrService) {
    this.arregloPc = ARREGLO_PC;
    this.arregloMonitor = ARRAY_MONITORES;
    this.arregloTeclado = ARRAY_TECLADOS;
    this.arregloRaton = ARRAY_RATONES;
    this.pcSeleccionado = globals.inicializarPc();

  }

  ngOnInit(): void {
    this.inicializarCombo();
    this.cargarPc();
  }
```

formulario-computador.component.ts U

src > app > componente > computador > formulario-computador > formulario-computador.component.ts > FormularioCom

```ts
public cargarPc(): void {
  this.route.paramMap.subscribe((parametro: ParamMap) => {
    const dato = String(parametro.get("codPc"));
    const numero = parseInt(dato);
    this.arregloPc.find((r) => {
      if (r.idComputador === numero) {
        this.pcSeleccionado = { ...r };
      }
    });
  });
}

public inicializarCombo(): void {
  this.pcSeleccionado.monitor = ARRAY_MONITORES[0];
  this.pcSeleccionado.teclado = ARRAY_TECLADOS[0];
  this.pcSeleccionado.raton = ARRAY_RATONES[0];
}

public operaciones(form: NgForm, event: any): void {
  const accion = event.submitter.id;

  switch (accion) {
    case "btnCrearPc":
      this.crearPc();
      break;
    case "btnActualizarPc":
      this.actualizarPc(form);
      break;
  }
}

public crearPc(): void {
  if (this.validarCamposPc()) {
    mostrarMensaje("success", "Agregado con exito", "El Computador " + this.pcSeleccionado.nombre, this.toastr);
    this.pcSeleccionado.idComputador = this.arregloPc.length + 1;
    this.arregloPc.push(this.pcSeleccionado)
  }else{
    mostrarMensaje("error", "No se pueden crear el computador<br />con campos vacíos", "Advertencia",this.toastr);
  }
  this.resetPc();
```

formulario-computador.component.ts U

src > app > componente > computador > formulario-computador > formulario-computador.component.ts > FormularioComputadorC

```
80      }else{
81        mostrarMensaje("error", "No se pueden crear el computador<br />con campos vacíos", "Advertencia",this.toastr);
82      }
83      this.resetPc();
84
85      this.inicializarCombo();
86    }
87
88    public actualizarPc(form: NgForm): void {
89      if (this.validarCamposPc()) {
90        const index = this.arregloPc.findIndex(pc => pc.idComputador === this.pcSeleccionado.idComputador);
91        if (index !== -1) {
92          this.arregloPc[index] = this.pcSeleccionado = { ...this.pcSeleccionado };
93          mostrarMensaje("success", "Editado con exito", "El Computador " + this.pcSeleccionado.nombre, this.toastr);
94        }
95        this.resetPc();
96        form.resetForm();
97      }else{
98        mostrarMensaje("warning", "No se puede actualizar el computador<br />con campos vacíos", "Advertencia",this.toastr);
99
100     }
101   }
102   //Validación de campos
103   public validarCamposPc(): boolean {
104     var bandera = true;
105     if(this.pcSeleccionado.nombre == "" || this.pcSeleccionado.monitor.idMonitor == 0 || this.pcSeleccionado.raton.id == 0
106     || this.pcSeleccionado.teclado.id == 0 || this.pcSeleccionado.publicoFotoComputador == ""){
107       bandera =  false;
108     }
109     return bandera;
110   }
111
112   public resetPc(): void {
113     this.misRutas.navigateByUrl("/dash/computer");
114     this.pcSeleccionado = globals.inicializarPc();
115   }
116
117   public cancelForm(form: NgForm): void {
118     this.resetPc();
119     form.resetForm();
120     this.inicializarCombo();
121   }
```

```
122
123   // Manejo input de la imagen
124   // ************************************************************************
125   public seleccionarFoto(objeto: any): any {
126     let caja = objeto.target.files[0]
127     if (!caja || caja.length == 0) {
128       return;
129     }
130     if (caja.type.match(/image\/*/) == null) {
131       return;
132     }
133     const reader = new FileReader();
134     reader.readAsDataURL(caja);
135     reader.onload = () => {
136       this.tmpBase64 = reader.result;
137       this.pcSeleccionado.publicoFotoComputador = caja.name;
138       this.pcSeleccionado.base64Computador = this.tmpBase64;
139     };
140   }
141
142
143 }
144
```

Fuente: Autor

La Tabla 18 Presenta el código completo de la clase del componente “Computador” que corresponde al archivo “***src\app\componente\computador\tabla-computador\tabla computador.component.ts***”

```
import { Component, OnInit } from '@angular/core';
import { Computador } from '../../../modelos/computador.model';
import { Monitor } from '../../../modelos/monitor.model';
import { Teclado } from '../../../modelos/teclado.model';
import { Raton } from '../../../modelos/raton.model';
import { ActivatedRoute, ParamMap, Router } from '@angular/router';
import { ARREGLO_PC } from '../../../mocks/computadores.mock';
import { ARRAY_MONITORES } from '../../../mocks/monitores.mock';
import { ARRAY_TECLADOS } from '../../../mocks/teclados.mock';
import { ARRAY_RATONES } from '../../../mocks/ratones.mock';
import { ToastrService } from 'ngx-toastr';
import { NgForm } from '@angular/forms';
import { mostrarMensaje } from '../../../utilidades/mensajes/toast.func';
import * as globals from '../../../utilidades/globals/inits.func';

@Component({
  selector: 'app-formulario-computador',
  templateUrl: './formulario-computador.component.html',
  styleUrl: './formulario-computador.component.css'
})

export class FormularioComputadorComponent implements OnInit {
  public pcSeleccionado: Computador;
  public arregloPc: Computador[];
  public arregloMonitor: Monitor[];
  public arregloTeclado: Teclado[];
  public arregloRaton: Raton[];
  public tmpBase64: any;

  // Inicialización parámetros en el constructor
  constructor(public route: ActivatedRoute, public misRutas: Router, public toastr:
ToastrService) {
    this.arregloPc = ARREGLO_PC;
    this.arregloMonitor = ARRAY_MONITORES;
    this.arregloTeclado = ARRAY_TECLADOS;
    this.arregloRaton = ARRAY_RATONES;
    this.pcSeleccionado = globals.inicializarPc();

  }

  ngOnInit(): void {
    this.inicializarCombo();
    this.cargarPc();
  }
/**La función "cargarPc" recupera un objeto de computadora específico de una matriz
basada en un parámetro * dado.*/
  public cargarPc(): void {
    this.route.paramMap.subscribe((parametro: ParamMap) => {
      const dato = String(parametro.get("codPc"));
      const numero = parseInt(dato);
      this.arregloPc.find((r) => {
        if (r.idComputador === numero) {
          this.pcSeleccionado = { ...r };
        }
      });
    });
  }
  public inicializarCombo(): void {
    this.pcSeleccionado.monitor = ARRAY_MONITORES[0];
    this.pcSeleccionado.teclado = ARRAY_TECLADOS[0];
    this.pcSeleccionado.raton = ARRAY_RATONES[0];
  }

  public operaciones(form: NgForm, event: any): void {
    const accion = event.submitter.id;

    switch (accion) {
```

```
      case "btnCrearPc":
        this.crearPc();
        break;
      case "btnActualizarPc":
        this.actualizarPc(form);
        break;
    }
  }

  public crearPc(): void {
    if (this.validarCamposPc()) {
      mostrarMensaje("success", "Agregado con exito", "El Computador " +
this.pcSeleccionado.nombre, this.toastr);
      this.pcSeleccionado.idComputador = this.arregloPc.length + 1;
      this.arregloPc.push(this.pcSeleccionado)
    }else{
    mostrarMensaje("error", "No se pueden crear el computador<br />con campos vacíos",
"Advertencia",this.toastr);
    }
    this.resetPc();

    this.inicializarCombo();
  }

  public actualizarPc(form: NgForm): void {
    if (this.validarCamposPc()) {
      const index = this.arregloPc.findIndex(pc => pc.idComputador ===
this.pcSeleccionado.idComputador);
      if (index !== -1) {
        this.arregloPc[index] = this.pcSeleccionado = { ...this.pcSeleccionado };
        mostrarMensaje("success", "Editado con exito", "El Computador " +
this.pcSeleccionado.nombre, this.toastr);
      }
      this.resetPc();
      form.resetForm();
    }else{
 mostrarMensaje("warning", "No se puede actualizar el computador<br />con campos vacíos",
"Advertencia",this.toastr);
    }
  }
  /*Validación de campos de los inputs, hasta que sean diferentes de vacío, levanta
una bandera boolean según el caso*/
  public validarCamposPc(): boolean {
    var bandera = true;
    if(this.pcSeleccionado.nombre == "" || this.pcSeleccionado.monitor.idMonitor == 0
|| this.pcSeleccionado.raton.id == 0
    || this.pcSeleccionado.teclado.id == 0 ||
this.pcSeleccionado.publicoFotoComputador == ""){
      bandera =  false;
    }
    return bandera;
  }
  public resetPc(): void {
    this.misRutas.navigateByUrl("/dash/computer");
    this.pcSeleccionado = globals.inicializarPc();
  }
  public cancelForm(form: NgForm): void {
    this.resetPc();
    form.resetForm();
    this.inicializarCombo();
  }
   // Manejo input de la caja para la imagen URL, que se convertirá a base64
  // ************************************************************************
  public seleccionarFoto(objeto: any): any {
    let caja = objeto.target.files[0]
    if (!caja || caja.length == 0) {
      return;
    }
    if (caja.type.match(/image\/*/) == null) {
      return;
    }
  const reader = new FileReader();
```

```
    reader.readAsDataURL(caja);
    reader.onload = () => {
      this.tmpBase64 = reader.result;
      this.pcSeleccionado.publicoFotoComputador = caja.name;
      this.pcSeleccionado.base64Computador = this.tmpBase64;
    };
  }
}
```

Tabla 18 Lógica CURD para el formulario de un computador

¡¡¡Bonus!!! Así como se ha visto en algunos de los anteriores componentes .ts, cierta lógica se ejecuta en una especie de constructor que se usa para hacer llamadas a servicios o al backend, en el siguiente blog se explica a detalle la funcionalidad del **ngOnInit "*Cómo usar el método de ngOnInit de Angular*".**

La Tabla 19 se presenta el archivo del "Formulario Computador" que corresponde al archivo ***"src\app\componente\computador\formulario-computador\formulario computador.component.html"***

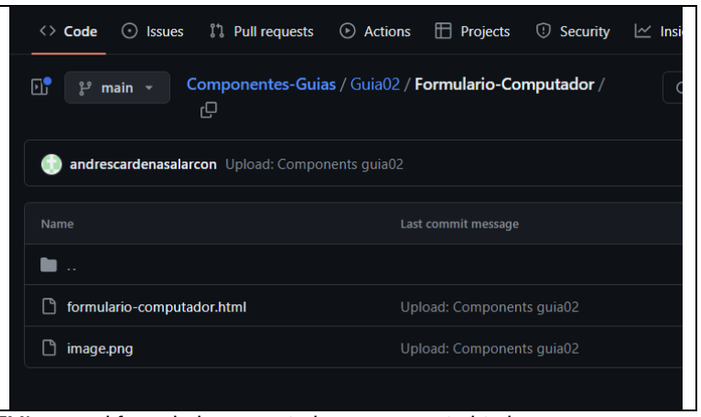
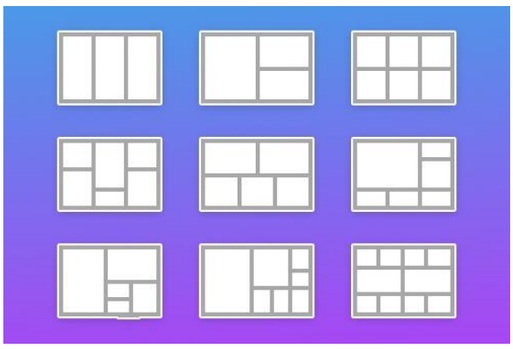
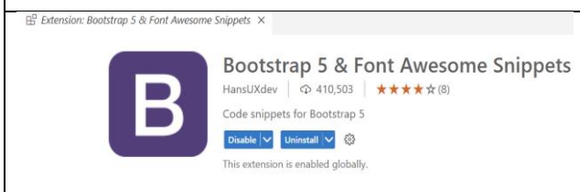
Componente GitHub Formulario-Computador	

Tabla 19 Plantilla HTML para el formulario-computador.componente.html

5.21. ¡¡¡Bonus!!!

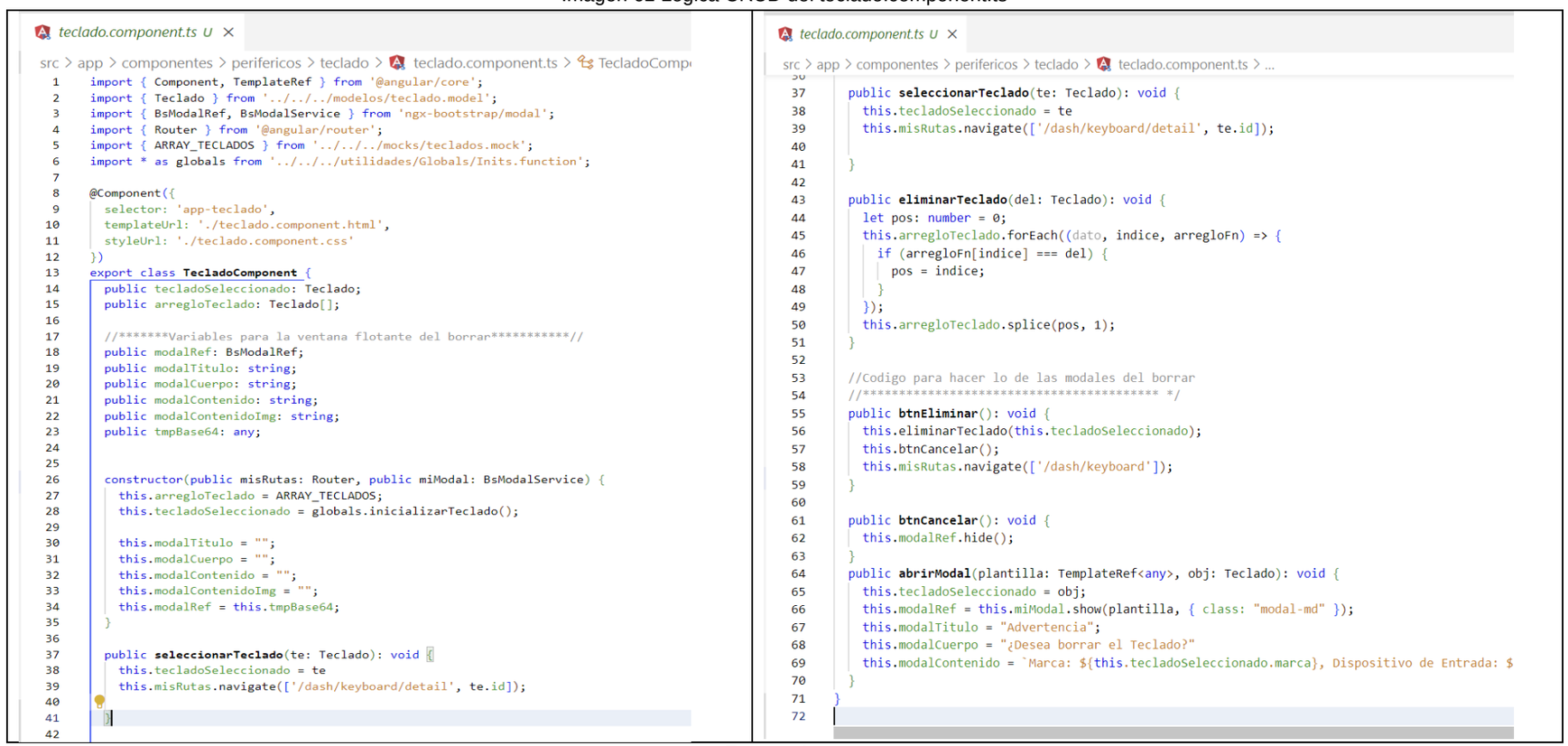
Las siguientes herramientas nos muestra una más fácil de entender o asimilar ciertos conceptos de maquetado HTML mediante los estilos de Boostrap5.

	La página web https://www.net-developer.nl/en/bootstrap-grid-generator.html es un generador de cuadrícula basado en Bootstrap. Su objetivo principal es ayudar a los desarrolladores a crear sitios web responsivos y orientados a dispositivos móviles utilizando el sistema de cuadrícula de Bootstrap.
	La siguiente extensión para VisualCode nos ayuda a mostrar y completar ciertos estilos que encontraríamos en la documentación de Boostrap5.

5.22. Componente Teclado

El componente que presentará funcionalidades de tipo CRUD (Create, Read, Update y Delete) será el componente “TablaTeclado”. A continuación, se hará la gestión de la clase del componente. La Imagen 61 presenta el contenido del archivo “**src\app\componente\periferico\teclado\teclado.component.ts**”

Imagen 61 Lógica CRUD del teclado.component.ts



teclado.component.ts U

src > app > componentes > periféricos > teclado > teclado.component.ts > TecladoComp

```ts
import { Component, TemplateRef } from '@angular/core';
import { Teclado } from '../../../modelos/teclado.model';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';
import { Router } from '@angular/router';
import { ARRAY_TECLADOS } from '../../../mocks/teclados.mock';
import * as globals from '../../../utilidades/Globals/Inits.function';

@Component({
  selector: 'app-teclado',
  templateUrl: './teclado.component.html',
  styleUrl: './teclado.component.css'
})
export class TecladoComponent {
  public tecladoSeleccionado: Teclado;
  public arregloTeclado: Teclado[];

  //*******Variables para la ventana flotante del borrar**********//
  public modalRef: BsModalRef;
  public modalTitulo: string;
  public modalCuerpo: string;
  public modalContenido: string;
  public modalContenidoImg: string;
  public tmpBase64: any;


  constructor(public misRutas: Router, public miModal: BsModalService) {
    this.arregloTeclado = ARRAY_TECLADOS;
    this.tecladoSeleccionado = globals.inicializarTeclado();

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

  public seleccionarTeclado(te: Teclado): void {
    this.tecladoSeleccionado = te
    this.misRutas.navigate(['/dash/keyboard/detail', te.id]);
  }

```

teclado.component.ts U

src > app > componentes > periféricos > teclado > teclado.component.ts > ...

```ts
  public seleccionarTeclado(te: Teclado): void {
    this.tecladoSeleccionado = te
    this.misRutas.navigate(['/dash/keyboard/detail', te.id]);

  }

  public eliminarTeclado(del: Teclado): void {
    let pos: number = 0;
    this.arregloTeclado.forEach((dato, indice, arregloFn) => {
      if (arregloFn[indice] === del) {
        pos = indice;
      }
    });
    this.arregloTeclado.splice(pos, 1);
  }

  //Codigo para hacer lo de las modales del borrar
  //************************************* */
  public btnEliminar(): void {
    this.eliminarTeclado(this.tecladoSeleccionado);
    this.btnCancelar();
    this.misRutas.navigate(['/dash/keyboard']);
  }

  public btnCancelar(): void {
    this.modalRef.hide();
  }
  public abrirModal(plantilla: TemplateRef<any>, obj: Teclado): void {
    this.tecladoSeleccionado = obj;
    this.modalRef = this.miModal.show(plantilla, { class: "modal-md" });
    this.modalTitulo = "Advertencia";
    this.modalCuerpo = "¿Desea borrar el Teclado?"
    this.modalContenido = `Marca: ${this.tecladoSeleccionado.marca}, Dispositivo de Entrada: $
  }
}

```

Fuente: Autor

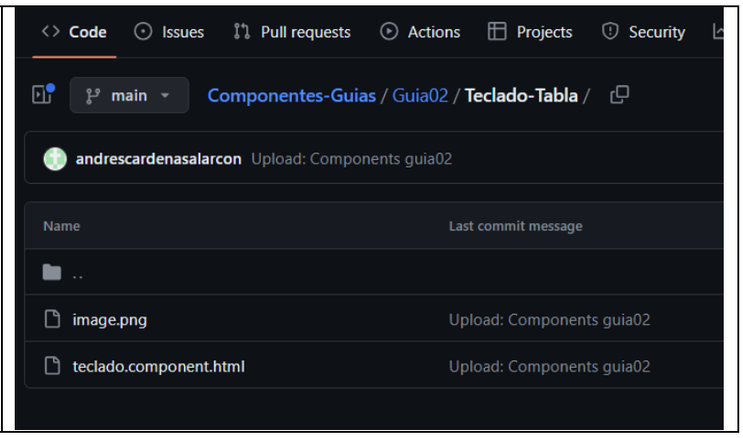
La Tabla 20 se presenta el código completo de la clase del componente “teclado” que corresponde al archivo “**src\app\componente\periferico\teclado\teclado.component.ts**”

```
import { Component, TemplateRef } from '@angular/core';
import { Teclado } from '../../../modelos/teclado.model';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';
import { Router } from '@angular/router';
import { ARRAY_TECLADOS } from '../../../mocks/teclados.mock';
import * as globals from '../../../utilidades/globals/inits.func';

@Component({
  selector: 'app-teclado',
  templateUrl: './teclado.component.html',
  styleUrl: './teclado.component.css'
})

export class TecladoComponent {
  public tecladoSeleccionado: Teclado;
  public arregloTeclado: Teclado[];

  //*******Variables para la ventana flotante del borrar**********//
  public modalRef: BsModalRef;
  public modalTitulo: string;
  public modalCuerpo: string;
  public modalContenido: string;
  public modalContenidoImg: string;
  public tmpBase64: any;


  constructor(public misRutas: Router, public miModal: BsModalService) {
    this.arregloTeclado = ARRAY_TECLADOS;
    this.tecladoSeleccionado = globals.inicializarTeclado();

    this.modalTitulo = "";
    this.modalCuerpo = "";
    this.modalContenido = "";
    this.modalContenidoImg = "";
    this.modalRef = this.tmpBase64;
  }
/**La función "seleccionarTeclado" selecciona un teclado y navega a la página
de detalles.*/
  public seleccionarTeclado(te: Teclado): void {
    this.tecladoSeleccionado = te
    this.misRutas.navigate(['/dash/keyboard/detail', te.id]);
  }

  public eliminarTeclado(del: Teclado): void {
    let pos: number = 0;
    this.arregloTeclado.forEach((dato, indice, arregloFn) => {
      if (arregloFn[indice] === del) {
        pos = indice;
      }
    });
    this.arregloTeclado.splice(pos, 1);
  }

  //Código para hacer la modal del borrar
  //************************************** */
  public btnEliminar(): void {
    this.eliminarTeclado(this.tecladoSeleccionado);
    this.btnCancelar();
    this.misRutas.navigate(['/dash/keyboard']);
}
```

```
  public btnCancelar(): void {
    this.modalRef.hide();
  }
  public abrirModal(plantilla: TemplateRef<any>, obj: Teclado): void {
    this.tecladoSeleccionado = obj;
    this.modalRef = this.miModal.show(plantilla, { class: "modal-md" });
    this.modalTitulo = "Advertencia";
    this.modalCuerpo = "¿Desea borrar el Teclado?"
    this.modalContenido = `Marca: ${this.tecladoSeleccionado.marca}, Dispositivo de
Entrada: ${this.tecladoSeleccionado.dispositivoEntrada}`;
  }
}
```

Tabla 20 Lógica para el teclado.component.ts

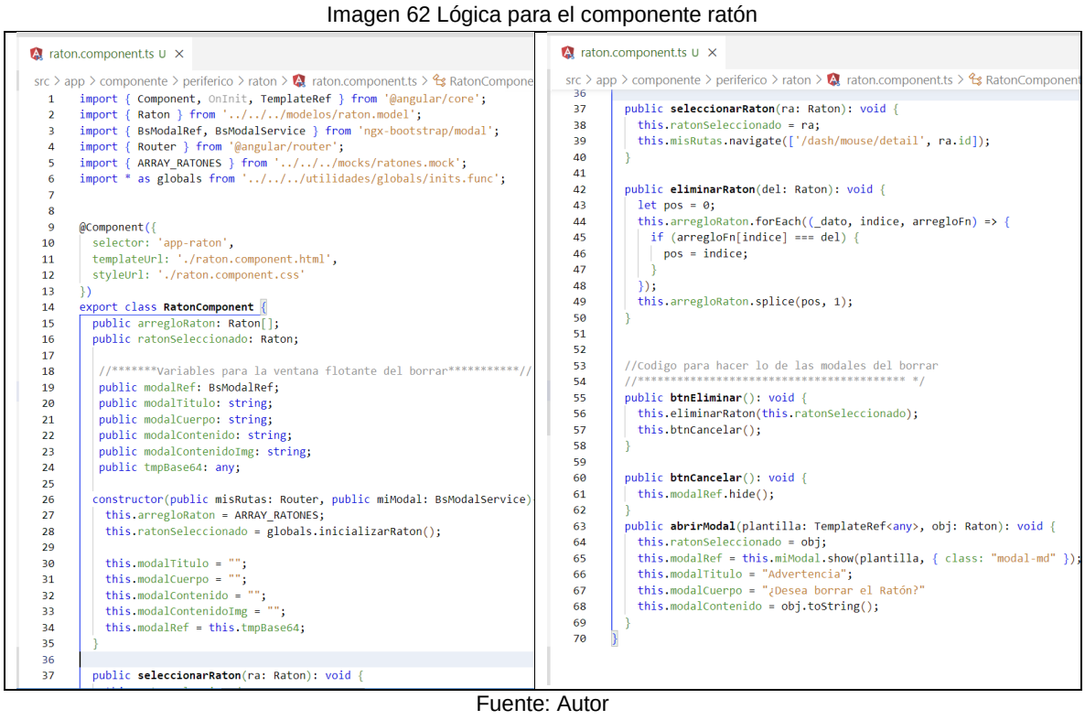
Se presenta el código completo de la clase del componente “teclado” que corresponde al archivo “**src\app\componente\periferico\teclado\teclado.component.html**”

Componente GitHub Tabla-Teclado	

5.23. Componente Ratón

El componente que presentará funcionalidades de tipo CRUD (Create, Read, Update y Delete) será el componente “TablaRaton”. A continuación, se hará la gestión de la clase del componente. La Imagen 62 presenta el contenido del archivo **“src\app\componente\periferico\raton\raton.component.ts**”

Imagen 62 Lógica para el componente ratón



```
import { Component, OnInit, TemplateRef } from '@angular/core';
import { Raton } from '../../../modelos/raton.model';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';
import { Router } from '@angular/router';
import { ARRAY_RATONES } from '../../../mocks/ratones.mock';
import * as globals from '../../../utilidades/globals/inits.func';


@Component({
  selector: 'app-raton',
  templateUrl: './raton.component.html',
  styleUrl: './raton.component.css'
})
export class RatonComponent {
  public arregloRaton: Raton[];
  public ratonSeleccionado: Raton;

  //*******Variables para la ventana flotante del borrar**********//
  public modalRef: BsModalRef;
  public modalTitulo: string;
  public modalCuerpo: string;
  public modalContenido: string;
  public modalContenidoImg: string;
  public tmpBase64: any;

  constructor(public misRutas: Router, public miModal: BsModalService) {
    this.arregloRaton = ARRAY_RATONES;
    this.ratonSeleccionado = globals.inicializarRaton();

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

  public seleccionarRaton(ra: Raton): void {
```

```
  public seleccionarRaton(ra: Raton): void {
    this.ratonSeleccionado = ra;
    this.misRutas.navigate(['/dash/mouse/detail', ra.id]);
  }

  public eliminarRaton(del: Raton): void {
    let pos = 0;
    this.arregloRaton.forEach((_dato, indice, arregloFn) => {
      if (arregloFn[indice] === del) {
        pos = indice;
      }
    });
    this.arregloRaton.splice(pos, 1);
  }


  //Codigo para hacer lo de las modales del borrar
  //************************************** */
  public btnEliminar(): void {
    this.eliminarRaton(this.ratonSeleccionado);
    this.btnCancelar();
  }

  public btnCancelar(): void {
    this.modalRef.hide();
  }
  public abrirModal(plantilla: TemplateRef<any>, obj: Raton): void {
    this.ratonSeleccionado = obj;
    this.modalRef = this.miModal.show(plantilla, { class: "modal-md" });
    this.modalTitulo = "Advertencia";
    this.modalCuerpo = "¿Desea borrar el Ratón?"
    this.modalContenido = obj.toString();
  }
}
```

Fuente: Autor

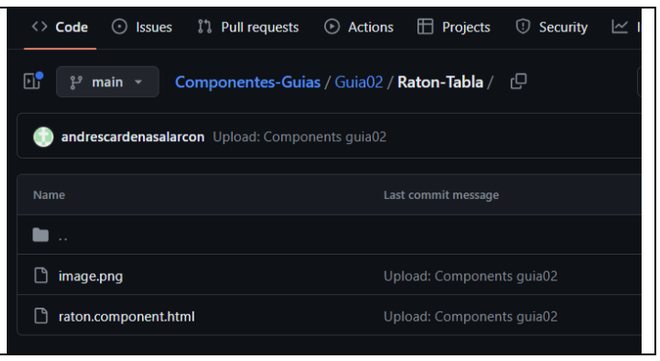
La Tabla 21 se presenta el código completo de la clase del componente “ratón” que corresponde al archivo “**src\app\componente\periferico\raton\raton.component.ts**”

```
import { Component, OnInit, TemplateRef } from '@angular/core';
import { Raton } from '../../../modelos/raton.model';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';
import { Router } from '@angular/router';
import { ARRAY_RATONES } from '../../../mocks/ratones.mock';
import * as globals from '../../../utilidades/globals/inits.func';

@Component({
  selector: 'app-raton',
  templateUrl: './raton.component.html',
  styleUrl: './raton.component.css'
})

export class RatonComponent {
  public arregloRaton: Raton[];
  public ratonSeleccionado: Raton;

   //*******Variables para la ventana flotante del borrar**********//
   public modalRef: BsModalRef;
   public modalTitulo: string;
   public modalCuerpo: string;
   public modalContenido: string;
   public modalContenidoImg: string;
   public tmpBase64: any;

  constructor(public misRutas: Router, public miModal: BsModalService){
    this.arregloRaton = ARRAY_RATONES;
    this.ratonSeleccionado = globals.inicializarRaton();
    this.modalTitulo = "";
    this.modalCuerpo = "";
    this.modalContenido = "";
    this.modalContenidoImg = "";
    this.modalRef = this.tmpBase64;
  }
  public seleccionarRaton(ra: Raton): void {
    this.ratonSeleccionado = ra;
    this.misRutas.navigate(['/dash/mouse/detail', ra.id]);
  }
/*The function iterates over the `arregloTeclado` array to find the index of
the Teclado object that matches the `del`*/
  public eliminarRaton(del: Raton): void {
    let pos = 0;
    this.arregloRaton.forEach((_dato, indice, arregloFn) => {
      if (arregloFn[indice] === del) {
        pos = indice;
      }
    });
    this.arregloRaton.splice(pos, 1);
  }
  //Código para hacer la modal del borrar
  public btnEliminar(): void {
    this.eliminarRaton(this.ratonSeleccionado);
    this.btnCancelar();
  }

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

```
  public abrirModal(plantilla: TemplateRef<any>, obj: Raton): void {
    this.ratonSeleccionado = obj;
    this.modalRef = this.miModal.show(plantilla, { class: "modal-md" });
    this.modalTitulo = "Advertencia";
    this.modalCuerpo = "¿Desea borrar el Ratón?"
    this.modalContenido = obj.toString();
  }
}
```

Tabla 21 Lógica para el ratón.component.ts

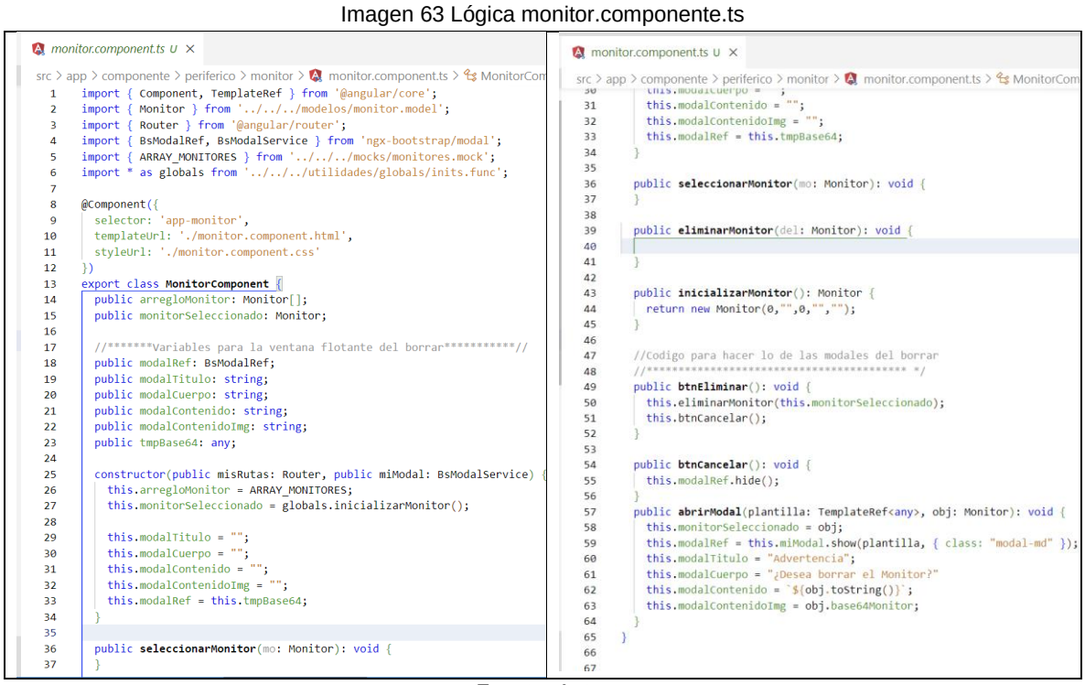
Se presenta el contenido de la plantilla de componente Ratón, la cual está ubicada en el archivo "***src\app\componente\periferico\raton\raton.component.html***"

Componente GitHub Raton-Tabla	

5.24. Componente Monitor

El componente que presentará funcionalidades de tipo CRUD (Create, Read, Update y Delete) será el componente "Monitor". A continuación, se hará la gestión de la clase del componente. La Imagen 63 presenta el contenido del archivo "**src\app\componente\periferico\monitor\monitor.component.ts**"

Imagen 63 Lógica monitor.componente.ts



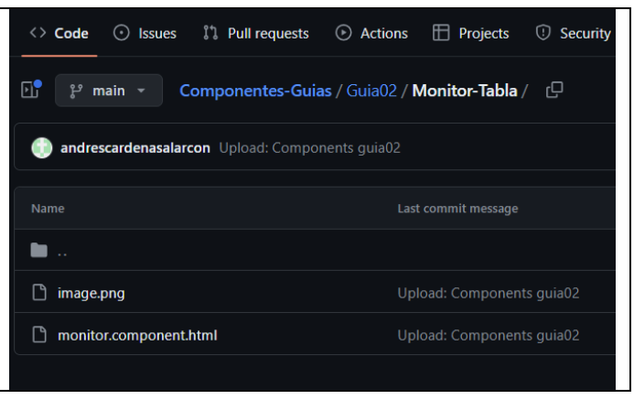
Fuente: Autor

La Tabla 22. Presenta el código completo de la clase del componente "monitor" que corresponde al archivo "**src\app\componente\periferico\monitor\monitor.component.ts**"

```
import { Component, TemplateRef } from '@angular/core';
import { Monitor } from '../../../modelos/monitor.model';
import { Router } from '@angular/router';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';
import { ARRAY_MONITORES } from '../../../mocks/monitores.mock';
import * as globals from '../../../utilidades/globals/inits.func';

@Component({
  selector: 'app-monitor',
  templateUrl: './monitor.component.html',
  styleUrl: './monitor.component.css'
})

export class MonitorComponent{
  public arregloMonitor: Monitor[];
  public monitorSeleccionado: Monitor;

  //*******Variables para la ventana flotante del borrar***********//
  public modalRef: BsModalRef;
  public modalTitulo: string;
  public modalCuerpo: string;
  public modalContenido: string;
  public modalContenidoImg: string;
  public tmpBase64: any;

  constructor(public misRutas: Router, public miModal: BsModalService) {
    this.arregloMonitor = ARRAY_MONITORES;
    this.monitorSeleccionado = globals.inicializarMonitor();

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

  public seleccionarMonitor(mo: Monitor): void {
  }

  public eliminarMonitor(del: Monitor): void {

  }
  //Código para hacer lo de las modales del borrar
  //************************************* */
  public abrirModal(plantilla: TemplateRef<any>, obj: Monitor): void {}
}
```

Tabla 22 Lógica para el monitor.component.ts

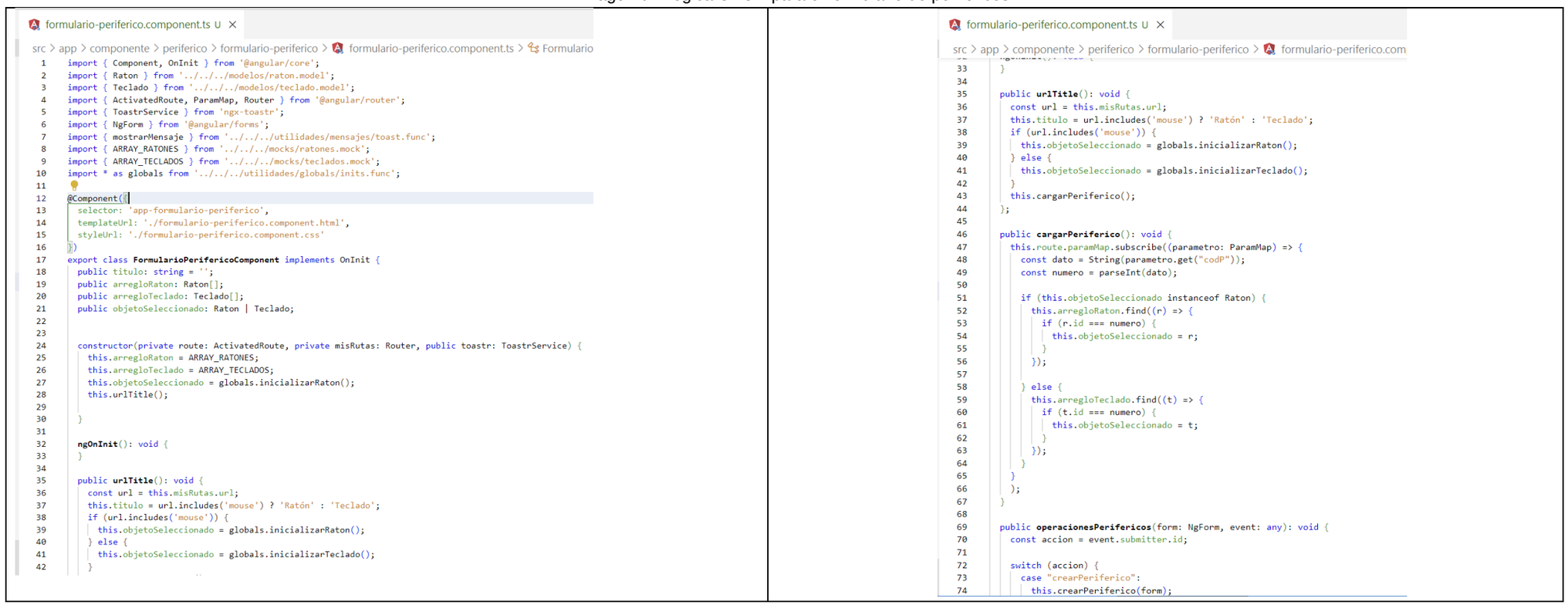
Se presenta el contenido de la plantilla de componente Monitor, la cual está ubicada en el archivo "**src\app\componente\periferico\monitor\monitor.component.html**"

Componente GitHub Monitor-Tabla	

5.25. Componente Formulario Periféricos (Ratón y Teclado)

El componente que presentará funcionalidades de tipo CRUD (Create, Read, Update y Delete) será el componente “Formulario-Periferico”. A continuación, se hará la gestión de la clase del componente. La Imagen 64 presenta el contenido del archivo **“src\app\componente\periferico\formulario-periferico\formulario-periferico.component.ts”**

Imagen 64 Lógica CRUD para el formulario de periféricos



formulario-periferico.component.ts U

src > app > componente > periferico > formulario-periferico > formulario-periferico.component.ts > Formulario

```ts
import { Component, OnInit } from '@angular/core';
import { Raton } from '../../../modelos/raton.model';
import { Teclado } from '../../../modelos/teclado.model';
import { ActivatedRoute, ParamMap, Router } from '@angular/router';
import { ToastrService } from 'ngx-toastr';
import { NgForm } from '@angular/forms';
import { mostrarMensaje } from '../../../utilidades/mensajes/toast.func';
import { ARRAY_RATONES } from '../../../mocks/ratones.mock';
import { ARRAY_TECLADOS } from '../../../mocks/teclados.mock';
import * as globals from '../../../utilidades/globals/inits.func';

@Component({
  selector: 'app-formulario-periferico',
  templateUrl: './formulario-periferico.component.html',
  styleUrl: './formulario-periferico.component.css'
})
export class FormularioPerifericoComponent implements OnInit {
  public titulo: string = '';
  public arregloRaton: Raton[];
  public arregloTeclado: Teclado[];
  public objetoSeleccionado: Raton | Teclado;


  constructor(private route: ActivatedRoute, private misRutas: Router, public toastr: ToastrService) {
    this.arregloRaton = ARRAY_RATONES;
    this.arregloTeclado = ARRAY_TECLADOS;
    this.objetoSeleccionado = globals.inicializarRaton();
    this.urlTitle();

  }

  ngOnInit(): void {
  }

  public urlTitle(): void {
    const url = this.misRutas.url;
    this.titulo = url.includes('mouse') ? 'Ratón' : 'Teclado';
    if (url.includes('mouse')) {
      this.objetoSeleccionado = globals.inicializarRaton();
    } else {
      this.objetoSeleccionado = globals.inicializarTeclado();
    }
```

formulario-periferico.component.ts U

src > app > componente > periferico > formulario-periferico > formulario-periferico.com

```ts
  }

  public urlTitle(): void {
    const url = this.misRutas.url;
    this.titulo = url.includes('mouse') ? 'Ratón' : 'Teclado';
    if (url.includes('mouse')) {
      this.objetoSeleccionado = globals.inicializarRaton();
    } else {
      this.objetoSeleccionado = globals.inicializarTeclado();
    }
    this.cargarPeriferico();
  };

  public cargarPeriferico(): void {
    this.route.paramMap.subscribe((parametro: ParamMap) => {
      const dato = String(parametro.get("codP"));
      const numero = parseInt(dato);

      if (this.objetoSeleccionado instanceof Raton) {
        this.arregloRaton.find((r) => {
          if (r.id === numero) {
            this.objetoSeleccionado = r;
          }
        });

      } else {
        this.arregloTeclado.find((t) => {
          if (t.id === numero) {
            this.objetoSeleccionado = t;
          }
        });
      }
    }
    );
  }

  public operacionesPerifericos(form: NgForm, event: any): void {
    const accion = event.submitter.id;

    switch (accion) {
      case "crearPeriferico":
        this.crearPeriferico(form);
```

formulario-periferico.component.ts U

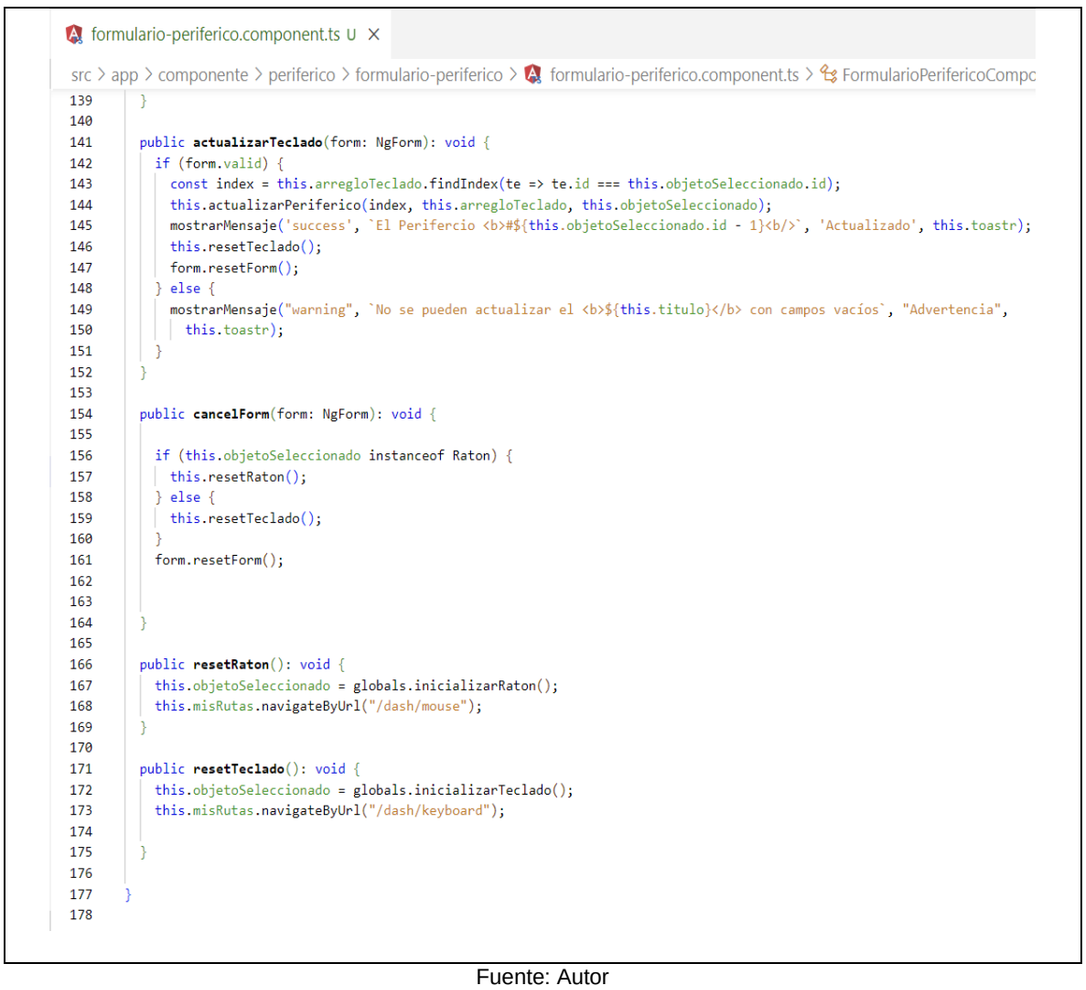
src > app > componente > periferico > formulario-periferico > formulario-periferico.component.ts > FormularioPerifericoC

```
68
69    public operacionesPerifericos(form: NgForm, event: any): void {
70      const accion = event.submitter.id;
71
72      switch (accion) {
73        case "crearPeriferico":
74          this.crearPeriferico(form);
75          break;
76        case "actualizarPeriferico":
77          this.actualizacionPerifericos(form);
78          break;
79      }
80    }
81
82
83    public generarGenerico<A extends Raton | Teclado>(p: A[], obj: A): void {
84      p.push(obj);
85    };
86
87    public crearPeriferico(form: NgForm): void {
88      if (form.valid) {
89        if (this.objetoSeleccionado instanceof Raton) {
90          this.crearRaton();
91        } else {
92          this.crearTeclado();
93        }
94        form.resetForm();
95      } else {
96        mostrarMensaje("error", "No se pueden crear Periferico<br />con campos vacíos", "Advertencia", this.toastr);
97
98      }
99    }
100   public actualizacionPerifericos(form: NgForm): void {
101     if (this.objetoSeleccionado instanceof Raton) {
102       this.actualizarRaton(form);
103     } else {
104       this.actualizarTeclado(form);
105
106     }
107
108   }
109   public crearRaton(): void {
110     this.objetoSeleccionado.id = this.arregloRaton.length + 1;
```

formulario-periferico.component.ts U

src > app > componente > periferico > formulario-periferico > formulario-periferico.component.ts > FormularioPerifericoCompon

```
109   public crearRaton(): void {
110     this.objetoSeleccionado.id = this.arregloRaton.length + 1;
111     this.generarGenerico(this.arregloRaton, this.objetoSeleccionado);
112     mostrarMensaje('success', `Nuevo Periferico de tipo Ratón Creado!`, 'Añadido', this.toastr);
113     this.resetRaton();
114
115   }
116
117   public crearTeclado(): void {
118     this.objetoSeleccionado.id = this.arregloTeclado.length + 1;
119     this.generarGenerico(this.arregloTeclado, this.objetoSeleccionado);
120     mostrarMensaje('success', `Nuevo Periferico de tipo Teclado Creado!`, 'Añadido', this.toastr);
121     this.resetTeclado();
122   }
123
124   public actualizarPeriferico<T extends Raton | Teclado>(index: number, arrayObj: T[], obj: T): void {
125     arrayObj[index] = obj;
126   }
127
128   public actualizarRaton(form: NgForm): void {
129     if (form.valid) {
130       const index = this.arregloRaton.findIndex(ra => ra.id === this.objetoSeleccionado.id);
131       this.actualizarPeriferico(index, this.arregloRaton, this.objetoSeleccionado);
132       mostrarMensaje('success', `El Periferico <b>#${this.objetoSeleccionado.id - 1}<b/>`, 'Actualizado', this.toastr);
133       this.resetRaton();
134       form.resetForm();
135     } else {
136       mostrarMensaje("warning", `No se pueden actualizar el <b>${this.titulo}</b> con campos vacíos`, "Advertencia",
137         this.toastr);
138     }
139   }
140
141   public actualizarTeclado(form: NgForm): void {
142     if (form.valid) {
143       const index = this.arregloTeclado.findIndex(te => te.id === this.objetoSeleccionado.id);
144       this.actualizarPeriferico(index, this.arregloTeclado, this.objetoSeleccionado);
145       mostrarMensaje('success', `El Periferico <b>#${this.objetoSeleccionado.id - 1}<b/>`, 'Actualizado', this.toastr);
146       this.resetTeclado();
147       form.resetForm();
148     } else {
149       mostrarMensaje("warning", `No se pueden actualizar el <b>${this.titulo}</b> con campos vacíos`, "Advertencia",
150         this.toastr);
```



Fuente: Autor

La Tabla 23 Presenta el código completo de la clase del componente “Formulario-Periferico” que corresponde al archivo “**src\app\componente\periferico\formulario-periferico\formulario-periferico.component.ts**”

```
import { Component, OnInit } from '@angular/core';
import { Raton } from '../../../modelos/raton.model';
import { Teclado } from '../../../modelos/teclado.model';
import { ActivatedRoute, ParamMap, Router } from '@angular/router';
import { ToastrService } from 'ngx-toastr';
import { NgForm } from '@angular/forms';
import { mostrarMensaje } from '../../../utilidades/mensajes/toast.func';
import { ARRAY_RATONES } from '../../../mocks/ratones.mock';
import { ARRAY_TECLADOS } from '../../../mocks/teclados.mock';
import * as globals from '../../../utilidades/globals/inits.func';

@Component({
  selector: 'app-formulario-periferico',
  templateUrl: './formulario-periferico.component.html',
  styleUrl: './formulario-periferico.component.css'
})

export class FormularioPerifericoComponent{
  public titulo: string = '';
  public arregloRaton: Raton[];
  public arregloTeclado: Teclado[];
  public objetoSeleccionado: Raton | Teclado; //Objeto Raton ó Teclado


  constructor(private route: ActivatedRoute, private misRutas: Router, public toastr:
ToastrService) {
    this.arregloRaton = ARRAY_RATONES;
    this.arregloTeclado = ARRAY_TECLADOS;
    this.objetoSeleccionado = globals.inicializarRaton();
    this.urlTitle();
  }
```

```
  public urlTitle(): void {
    const url = this.misRutas.url;
    this.titulo = url.includes('mouse') ? 'Ratón' : 'Teclado';
    if (url.includes('mouse')) {
      this.objetoSeleccionado = globals.inicializarRaton();
    } else {
      this.objetoSeleccionado = globals.inicializarTeclado();
    }
    this.cargarPeriferico();
  };
  public cargarPeriferico(): void {
    this.route.paramMap.subscribe((parametro: ParamMap) => {
      const dato = String(parametro.get("codP")); /*--> trae el parámetro que viene
del archivo app.routing */
      const numero = parseInt(dato);
      /* identificamos el tipo la clase del objeto seleccionado mediante el instanceof
Raton o Teclado */
      if (this.objetoSeleccionado instanceof Raton) {
        this.arregloRaton.find((r) => {
          if (r.id === numero) {
            this.objetoSeleccionado = r;
          }
        });

      } else { /* De lo contrario es un objeto de tipo teclado, y lo busca dentro del
array */
        this.arregloTeclado.find((t) => {
          if (t.id === numero) {
            this.objetoSeleccionado = t;
          }
        });
      }
    }
    );
  }
  public operacionesPerifericos(form: NgForm, event: any): void {
    const accion = event.submitter.id;

    switch (accion) {
      case "crearPeriferico":
        this.crearPeriferico(form);
        break;
      case "actualizarPeriferico":
        this.actualizacionPerifericos(form);
        break;
    }
  }
  public crearPeriferico(form: NgForm): void {
    if (form.valid) {
      if (this.objetoSeleccionado instanceof Raton) {
        this.crearRaton();
      } else {
        this.crearTeclado();
      }
      form.resetForm();
    } else {
      mostrarMensaje("error", "No se pueden crear Periféricos<br />con campos vacíos",
"Advertencia", this.toastr);

    }
  }
  public actualizacionPerifericos(form: NgForm): void {
    if (this.objetoSeleccionado instanceof Raton) {
      this.actualizarRaton(form);
    } else {
      this.actualizarTeclado(form);

    }
  } //Continua más código ...
```

El bloque **urlTitle**() de código se encarga de guardar el contenido de la URL de nuestro navegador este lo almacena en una constante url a lo que después hará; 1). Si en la url encuentra la palabra “mouse” asignará a la variable `this.titulo` la palabra “Ratón” de lo contrario asigna como título “Teclado”. 2). De la misma forma que la asignación del título realiza la inicialización del objeto `this.objetoSeleccionado` ya sea de tipo Ratón o Teclado.

El bloque de **cargarPeriferico**() trae el parámetro `"codP"` que en la url de nuestro navegador será el id de unos nuestros periféricos a lo que postreramente convertiremos a tipo number. El `this.objetoSeleccionado` y mediante el método “instanceOf” que nos validará el tipo de Object cargado en memoria, si es de tipo Ratón ejecutará la búsqueda en el `this.arregloRaton` mediante el numero id previamente guardado, en caso contrario buscará en el `this.arregloTeclado`.

5.25.1. Funciones Genéricas Formulario Periferico

Este bloque de código **generarGenerico()** y **actualizarPeriferico()** son funciones genericas, para no tener que crear dos “crear” para raton y teclado y dos “actualizar” hacemos uso de esta funcionalida, a la que le pasamos por parametro el arreglo, el objeto y en el caso de actualizar un index, y con el tipo de datos ingresados interpreta si es un Objeto de tipo Raton o Teclado realizando los metodos de “push” para crear o de asignacion en actualizar.

```
  public generarGenerico<A extends Raton | Teclado>(p: A[], obj: A): void {
    p.push(obj);
  };
  public actualizarPeriferico<T extends Raton | Teclado>(index: number, arrayObj: T[],
obj: T): void {
    arrayObj[index] = obj;
  }

  public crearRaton(): void {
    this.objetoSeleccionado.id = this.arregloRaton.length + 1;
    this.generarGenerico(this.arregloRaton, this.objetoSeleccionado);
    mostrarMensaje('success', `Nuevo Periferico de tipo Ratón Creado!`, 'Añadido',
this.toastr);
    this.resetRaton();

  }

  public crearTeclado(): void {
    this.objetoSeleccionado.id = this.arregloTeclado.length + 1;
    this.generarGenerico(this.arregloTeclado, this.objetoSeleccionado);
    mostrarMensaje('success', `Nuevo Periferico de tipo Teclado Creado!`, 'Añadido',
this.toastr);
    this.resetTeclado();
  }
  public actualizarRaton(form: NgForm): void {
    if (form.valid) {
      const index = this.arregloRaton.findIndex(ra => ra.id ===
this.objetoSeleccionado.id);
      this.actualizarPeriferico(index, this.arregloRaton, this.objetoSeleccionado);
      mostrarMensaje('success', `El Perifercio <b>#${this.objetoSeleccionado.id -
1}<b/>`, 'Actualizado', this.toastr);
      this.resetRaton();
      form.resetForm();
    } else {
      mostrarMensaje("warning", `No se pueden actualizar el <b>${this.titulo}</b> con
campos vacíos`, "Advertencia",
        this.toastr);
    }
  }
```

```
  public actualizarTeclado(form: NgForm): void {
    if (form.valid) {
      const index = this.arregloTeclado.findIndex(te => te.id ===
this.objetoSeleccionado.id);
      this.actualizarPeriferico(index, this.arregloTeclado, this.objetoSeleccionado);
      mostrarMensaje('success', `El Perifercio <b>#${this.objetoSeleccionado.id -
1}<b/>`, 'Actualizado', this.toastr);
      this.resetTeclado();
      form.resetForm();
    } else {
      mostrarMensaje("warning", `No se pueden actualizar el <b>${this.titulo}</b> con
campos vacíos`, "Advertencia",
        this.toastr);
    }
  }

  public cancelForm(form: NgForm): void {

    if (this.objetoSeleccionado instanceof Raton) {
      this.resetRaton();
    } else {
      this.resetTeclado();
    }
    form.resetForm();
  }
  public resetRaton(): void {
    this.objetoSeleccionado = globals.inicializarRaton();
    this.misRutas.navigateByUrl("/dash/mouse");
  }
  public resetTeclado(): void {
    this.objetoSeleccionado = globals.inicializarTeclado();
    this.misRutas.navigateByUrl("/dash/keyboard");

  }
}
```

Tabla 23 Lógica para el formulario de los periféricos

¡¡¡Bonus!!! En el siguiente artículo se explica las variaciones y ejemplos además de las ventajas de usar el código genérico para no repetir código, ***"Generics en TypeScript"***

La Tabla 24. presenta el contenido de la plantilla de componente Formulario-Periferico, la cual está ubicada en el archivo "**src\app\componente\periferico\formulario-periferico\formulario-periferico.component.html**"

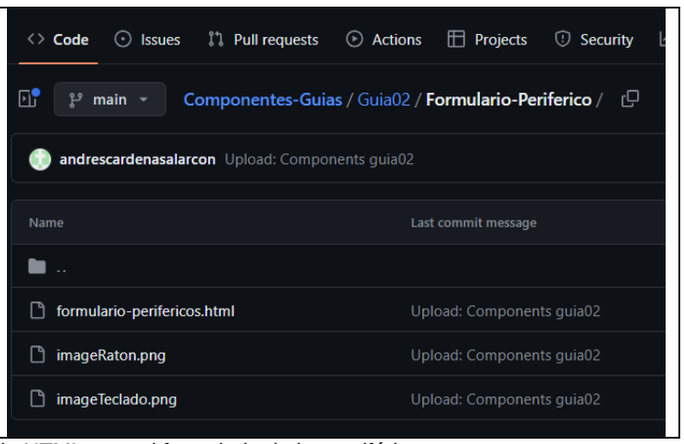
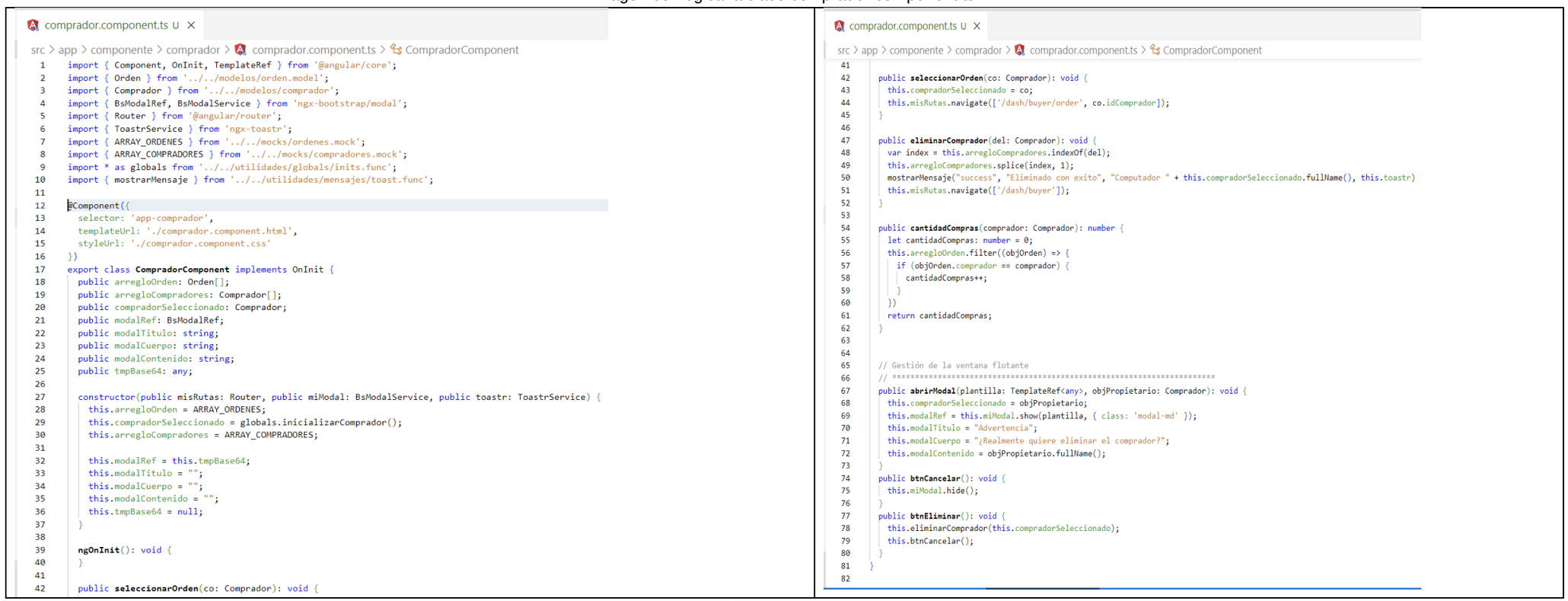
Componente Formulario-Periferico	

Tabla 24 Plantilla HTML para el formulario de los periféricos

5.26. Componente Comprador

El componente que presentará funcionalidades de tipo CRUD (Create, Read, Update y Delete) será el componente “Tabla-Comprador”. A continuación, se hará la gestión de la clase del componente. La Imagen 65 presenta el contenido del archivo “**src\app\componente\comprador\comprador.component.ts**”

Imagen 65 Lógica la clase comprador.component.ts



comprador.component.ts U

src > app > componente > comprador > comprador.component.ts > CompradorComponent

```ts
import { Component, OnInit, TemplateRef } from '@angular/core';
import { Orden } from '../../modelos/orden.model';
import { Comprador } from '../../modelos/comprador';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';
import { Router } from '@angular/router';
import { ToastrService } from 'ngx-toastr';
import { ARRAY_ORDENES } from '../../mocks/ordenes.mock';
import { ARRAY_COMPRADORES } from '../../mocks/compradores.mock';
import * as globals from '../../utilidades/globals/inits.func';
import { mostrarMensaje } from '../../utilidades/mensajes/toast.func';

@Component({
  selector: 'app-comprador',
  templateUrl: './comprador.component.html',
  styleUrl: './comprador.component.css'
})
export class CompradorComponent implements OnInit {
  public arregloOrden: Orden[];
  public arregloCompradores: Comprador[];
  public compradorSeleccionado: Comprador;
  public modalRef: BsModalRef;
  public modalTitulo: string;
  public modalCuerpo: string;
  public modalContenido: string;
  public tmpBase64: any;

  constructor(public misRutas: Router, public miModal: BsModalService, public toastr: ToastrService) {
    this.arregloOrden = ARRAY_ORDENES;
    this.compradorSeleccionado = globals.inicializarComprador();
    this.arregloCompradores = ARRAY_COMPRADORES;

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

  ngOnInit(): void {
  }

  public seleccionarOrden(co: Comprador): void {
```

comprador.component.ts U

src > app > componente > comprador > comprador.component.ts > CompradorComponent

```ts

  public seleccionarOrden(co: Comprador): void {
    this.compradorSeleccionado = co;
    this.misRutas.navigate(['/dash/buyer/order', co.idComprador]);
  }

  public eliminarComprador(del: Comprador): void {
    var index = this.arregloCompradores.indexOf(del);
    this.arregloCompradores.splice(index, 1);
    mostrarMensaje("success", "Eliminado con exito", "Computador " + this.compradorSeleccionado.fullName(), this.toastr)
    this.misRutas.navigate(['/dash/buyer']);
  }

  public cantidadCompras(comprador: Comprador): number {
    let cantidadCompras: number = 0;
    this.arregloOrden.filter((objOrden) => {
      if (objOrden.comprador == comprador) {
        cantidadCompras++;
      }
    })
    return cantidadCompras;
  }


  // Gestión de la ventana flotante
  // ********************************************************************
  public abrirModal(plantilla: TemplateRef<any>, objPropietario: Comprador): void {
    this.compradorSeleccionado = objPropietario;
    this.modalRef = this.miModal.show(plantilla, { class: 'modal-md' });
    this.modalTitulo = "Advertencia";
    this.modalCuerpo = "¿Realmente quiere eliminar el comprador?";
    this.modalContenido = objPropietario.fullName();
  }
  public btnCancelar(): void {
    this.miModal.hide();
  }
  public btnEliminar(): void {
    this.eliminarComprador(this.compradorSeleccionado);
    this.btnCancelar();
  }
}

```

Fuente: Autor

La Imagen 65 Presenta el código completo de la clase del componente “comprador” que corresponde al archivo “**src\app\componente\comprador\comprador.component.ts**”

```
import { Component, OnInit, TemplateRef } from '@angular/core';
import { Orden } from '../../modelos/orden.model';
import { Comprador } from '../../modelos/comprador.model';
import { BsModalRef, BsModalService } from 'ngx-bootstrap/modal';
import { Router } from '@angular/router';
import { ToastrService } from 'ngx-toastr';
import { ARRAY_ORDENES } from '../../mocks/ordenes.mock';
import { ARRAY_COMPRADORES } from '../../mocks/compradores.mock';
import * as globals from '../../utilidades/globals/inits.func';
import { mostrarMensaje } from '../../utilidades/mensajes/toast.func';

@Component({
  selector: 'app-comprador',
  templateUrl: './comprador.component.html',
  styleUrl: './comprador.component.css'
})

export class CompradorComponent{
  public arregloOrden: Orden[];
  public arregloCompradores: Comprador[];
  public compradorSeleccionado: Comprador;
  public modalRef: BsModalRef;
  public modalTitulo: string;
  public modalCuerpo: string;
  public modalContenido: string;
  public tmpBase64: any;

  constructor(public misRutas: Router, public miModal: BsModalService, public toastr:
ToastrService) {
    this.arregloOrden = ARRAY_ORDENES;
    this.compradorSeleccionado = globals.inicializarComprador();
    this.arregloCompradores = ARRAY_COMPRADORES;

    this.modalRef = this.tmpBase64;
    this.modalTitulo = "";
    this.modalCuerpo = "";
    this.modalContenido = "";
    this.tmpBase64 = null;
  }
  public seleccionarOrden(co: Comprador): void {
    this.compradorSeleccionado = co;
    this.misRutas.navigate(['/dash/buyer/order', co.idComprador]);
  }
  public eliminarComprador(del: Comprador): void {
    var index = this.arregloCompradores.indexOf(del);
    this.arregloCompradores.splice(index, 1);
    mostrarMensaje("success", "Eliminado con exito", "Comprador " +
this.compradorSeleccionado.fullName(), this.toastr)
    this.misRutas.navigate(['/dash/buyer']);
  }
  public cantidadCompras(comprador: Comprador): number {
    let cantidadCompras: number = 0;
    this.arregloOrden.filter((objOrden) => {
      if (objOrden.comprador == comprador) {
        cantidadCompras++;
      }
    })
    return cantidadCompras;
  }
  // Gestión de la ventana flotante
  // ************************************************************************
  public abrirModal(plantilla: TemplateRef<any>, objPropietario: Comprador): void {
    this.compradorSeleccionado = objPropietario;
    this.modalRef = this.miModal.show(plantilla, { class: 'modal-md' });
    this.modalTitulo = "Advertencia";
    this.modalCuerpo = "¿Realmente quiere eliminar el comprador?";
    this.modalContenido = objPropietario.fullName();
  }
```

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

  public btnEliminar(): void {
    this.eliminarComprador(this.compradorSeleccionado);
    this.btnCancelar();
  }
}
```

Tabla 25 Lógica para el comprador.component.ts

La Tabla 26 presenta el contenido de la plantilla de componente comprador, la cual está ubicada en el archivo "**src\app\componente\comprador\comprador.component.html**"

```
<nav class="pt-1" aria-label="breadcrumb">
    <ol class="breadcrumb">
      <li class="breadcrumb-item">
        <a [routerLink]="['/dash/root']" routerLinkActive="router-link-active">
          <i class="fa fa-home"></i>
        </a>
      </li>
      <li class="breadcrumb-item active"[routerLink]="['../buyer']" aria-current="page">
        <a href="javascript:void(0)">Ventas</a>
      </li>
      <li class="breadcrumb-item active" aria-current="page">Detalles</li>
    </ol>
  </nav>

    <div class="row m-2">
      <div class="col-md-6 mb-4 mb-md-0">
        <!-- TABLA COMPRADORES-VENTAS -->
        <h4>Tabla De Compradores</h4>
        <div class="table-responsive">
        <table class="table table-striped table-hover table-sm my-hover">
          <thead class="bg-dark text-white">
            <tr class="table-dark">
              <th scope="col">Código</th>
              <th scope="col">Nombres</th>
              <th scope="col">Apellidos</th>
              <th scope="col">Documento</th>
              <th scope="col">Compras</th>
              <th scope="col"></th>
            </tr>
          </thead>
          <tbody>
              <tr
                *ngFor="let miComprador of arregloCompradores let indice = index"
                (click)="seleccionarOrden(miComprador)"
                [class.table-info]="miComprador == compradorSeleccionado"
              >
                <td>{{ indice }}</td>
                <td>
                  {{ miComprador.primerNombre }}
                </td>
                <td>
                  {{ miComprador.primerApellido }}
                </td>
                <td>{{ miComprador.documento }}</td>
                <td>{{ cantidadCompras(miComprador) }}</td>
                <td>
                  <i
                  class="far fa-trash-alt papeleraRoja"
                  (click)="cantidadCompras(miComprador) ? 0 : abrirModal(ventanaEliminar,
miComprador)"
                  [ngClass]="
                  cantidadCompras(miComprador) ? 'papeleraGris' : 'papeleraRoja'
                  "
                  >
                </i>
```

```
                </td>
              </tr>
            </tbody>
          </table>
        </div>
      </div>
      <div class="col-md-6">
        <!-- TABLA DE ORDENES-->
        <router-outlet></router-outlet>
      </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 26 Plantilla HTML para el componente comprador.html

La Tabla 27 presenta el contenido de los estilos del componente comprador, la cual está ubicada en el archivo “**src\app\componente\comprador\comprador.component.css**”

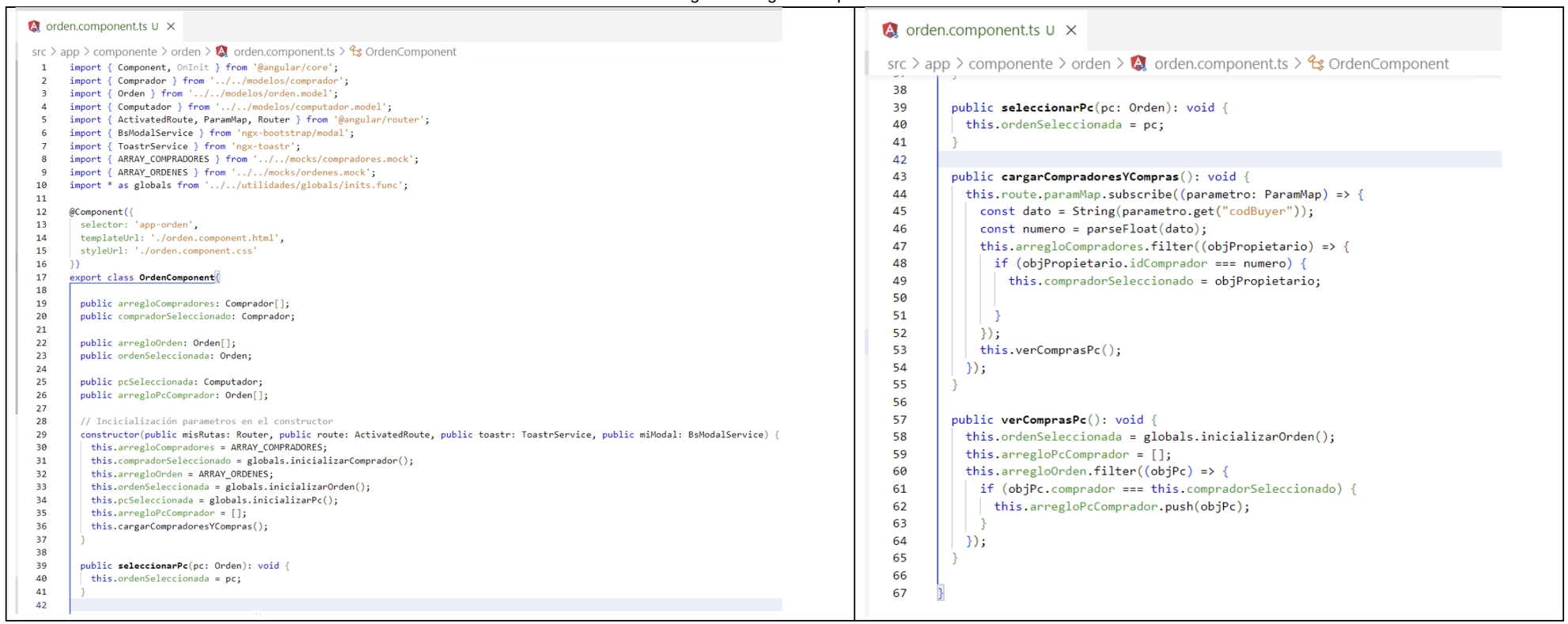
```
.papeleraRoja {
    color: #9e2702;
}

.papeleraGris {
    color: #a09898;
}
```

Tabla 27 Estilos componente comprador.css

5.27. Componente Orden de Venta

El componente que presentará funcionalidades de tipo CRUD (Create, Read, Update y Delete) será el componente “Orden”. A continuación, se hará la gestión de la clase del componente. La Imagen 66 presenta el contenido del archivo “**src\app\componente\orden\orden.component.ts**”

Imagen 66 Lógica componente orden.ts



orden.component.ts U

src > app > componente > orden > orden.component.ts > OrdenComponent

```ts
import { Component, OnInit } from '@angular/core';
import { Comprador } from '../../modelos/comprador';
import { Orden } from '../../modelos/orden.model';
import { Computador } from '../../modelos/computador.model';
import { ActivatedRoute, ParamMap, Router } from '@angular/router';
import { BsModalService } from 'ngx-bootstrap/modal';
import { ToastrService } from 'ngx-toastr';
import { ARRAY_COMPRADORES } from '../../mocks/compradores.mock';
import { ARRAY_ORDENES } from '../../mocks/ordenes.mock';
import * as globals from '../../utilidades/globals/inits.func';

@Component({
  selector: 'app-orden',
  templateUrl: './orden.component.html',
  styleUrl: './orden.component.css'
})
export class OrdenComponent{

  public arregloCompradores: Comprador[];
  public compradorSeleccionado: Comprador;

  public arregloOrden: Orden[];
  public ordenSeleccionada: Orden;

  public pcSeleccionada: Computador;
  public arregloPcComprador: Orden[];

  // Incicialización parametros en el constructor
  constructor(public misRutas: Router, public route: ActivatedRoute, public toastr: ToastrService, public miModal: BsModalService) {
    this.arregloCompradores = ARRAY_COMPRADORES;
    this.compradorSeleccionado = globals.inicializarComprador();
    this.arregloOrden = ARRAY_ORDENES;
    this.ordenSeleccionada = globals.inicializarOrden();
    this.pcSeleccionada = globals.inicializarPc();
    this.arregloPcComprador = [];
    this.cargarCompradoresYCompras();
  }

  public seleccionarPc(pc: Orden): void {
    this.ordenSeleccionada = pc;
  }

```

orden.component.ts U

src > app > componente > orden > orden.component.ts > OrdenComponent

```ts

  public seleccionarPc(pc: Orden): void {
    this.ordenSeleccionada = pc;
  }

  public cargarCompradoresYCompras(): void {
    this.route.paramMap.subscribe((parametro: ParamMap) => {
      const dato = String(parametro.get("codBuyer"));
      const numero = parseFloat(dato);
      this.arregloCompradores.filter((objPropietario) => {
        if (objPropietario.idComprador === numero) {
          this.compradorSeleccionado = objPropietario;

        }
      });
      this.verComprasPc();
    });
  }

  public verComprasPc(): void {
    this.ordenSeleccionada = globals.inicializarOrden();
    this.arregloPcComprador = [];
    this.arregloOrden.filter((objPc) => {
      if (objPc.comprador === this.compradorSeleccionado) {
        this.arregloPcComprador.push(objPc);
      }
    });
  }


}
```

Fuente: Autor

La Imagen 66 Presenta el código completo de la clase del componente “Orden” que corresponde al archivo “**src\app\componente\orden\orden.component.ts**”

```
import { Component, OnInit } from '@angular/core';
import { Comprador } from '../../modelos/comprador.model';
import { Orden } from '../../modelos/orden.model';
import { Computador } from '../../modelos/computador.model';
import { ActivatedRoute, ParamMap, Router } from '@angular/router';
import { BsModalService } from 'ngx-bootstrap/modal';
import { ToastrService } from 'ngx-toastr';
import { ARRAY_COMPRADORES } from '../../mocks/compradores.mock';
import { ARRAY_ORDENES } from '../../mocks/ordenes.mock';
import * as globals from '../../utilidades/globals/inits.func';
@Component({
  selector: 'app-orden',
  templateUrl: './orden.component.html',
  styleUrl: './orden.component.css'
})

export class OrdenComponent{

  public arregloCompradores: Comprador[];
  public compradorSeleccionado: Comprador;

  public arregloOrden: Orden[];
  public ordenSeleccionada: Orden;

  public pcSeleccionada: Computador;
  public arregloPcComprador: Orden[];

  // Incicialización parametros en el constructor
  constructor(public misRutas: Router, public route: ActivatedRoute, public toastr:
ToastrService, public miModal: BsModalService) {
    this.arregloCompradores = ARRAY_COMPRADORES;
    this.compradorSeleccionado = globals.inicializarComprador();
    this.arregloOrden = ARRAY_ORDENES;
    this.ordenSeleccionada = globals.inicializarOrden();
    this.pcSeleccionada = globals.inicializarPc();
    this.arregloPcComprador = [];
    this.cargarCompradoresYCompras();
  }

  public seleccionarPc(pc: Orden): void {
    this.ordenSeleccionada = pc;
  }

/**La función "cargarCompradoresYCompras" carga compradores y sus compras en función
de un parámetro de código de comprador * específico.*/
  public cargarCompradoresYCompras(): void {
//Trae los parámetros que vienen por la URL de nuestras rutas.ts
    this.route.paramMap.subscribe((parametro: ParamMap) => {
    const dato = String(parametro.get("codBuyer"));
```

```
      const numero = parseFloat(dato);
      this.arregloCompradores.filter((objPropietario) => {
        if (objPropietario.idComprador === numero) {
          this.compradorSeleccionado = objPropietario;
        }
      });
      this.verComprasPc();
    });
  }
/** La función `verComprasPc` inicializa un pedido y filtra un conjunto de PC en
función de un * comprador seleccionado.*/
  public verComprasPc(): void {
    this.ordenSeleccionada = globals.inicializarOrden();
    this.arregloPcComprador = [];
    this.arregloOrden.filter((objPc) => {
      if (objPc.comprador === this.compradorSeleccionado) {
        this.arregloPcComprador.push(objPc);
      }
    });
  }
}
```

Tabla 28 Lógica componente.orden.ts

La función método **`verComprasPc()`** se encarga de filtrar mediante el `this.arregloOrden` los computadores comprados o asignados a `this.compradorSeleccionado` seleccionado en la tabla de compradores.

La Tabla 29 presenta el contenido de la plantilla de componente orden, la cual está ubicada en el archivo "**src\app\componente\orden\orden.component.html**"

```
<div
  *ngIf="arregloPcComprador.length != 0; then mostrarTablaCompradores; else vacio"
></div>
<!-- TABLA ORDENES -->
<ng-template #mostrarTablaCompradores>
  <div class="row">
    <div class="col-md-8">
      <h4>Tabla De Ordenes</h4>
      <div class="table-responsive">
      <table class="table table-striped table-hover table-sm my-hover">
        <thead class="bg-dark text-white">
          <tr class="table-dark">
            <th scope="col" style="width: 10px">Código</th>
            <th scope="col" style="width: 30px">Nombre</th>
            <th scope="col" style="width: 15px">Monitor</th>
            <th scope="col" style="width: 15px">Ratón</th>
            <th scope="col" style="width: 20px">Teclado</th>
            <th scope="col" style="width: 20px">Precio</th>
          </tr>
        </thead>
        <tbody>
          <tr
            *ngFor="let miPc of arregloPcComprador; let index = index"
            (click)="seleccionarPc(miPc)"
            [class.table-info]="miPc == ordenSeleccionada"
          >
            <td>{{ miPc.computadoras.idComputador }}</td>
            <td>{{ miPc.computadoras.nombre }}</td>
            <td>{{ miPc.computadoras.monitor.marca }}[{{
miPc.computadoras.monitor.tamanno }}'']</td>
            <td>{{ miPc.computadoras.raton.marca }}</td>
            <td>{{ miPc.computadoras.teclado.marca }}</td>
            <td>{{ miPc.computadoras.getPrice!() }}</td>
          </tr>
        </tbody>
      </table>
    </div>
   </div>
   <div class="col-md-4">
      <div *ngIf="ordenSeleccionada.computadoras.idComputador != 0; then
contenido"></div>
   </div>
  </div>
</ng-template>
<ng-template #vacio>
  <div class="row mt-4">
    <div class="col-8">
      <div class="alert alert-danger text-danger">
        <h4>Advertencia!</h4>
        <p class="text-danger">
          No hay registros asociados al comprador seleccionado
        </p>
      </div>
    </div>
  </div>
</ng-template>
```

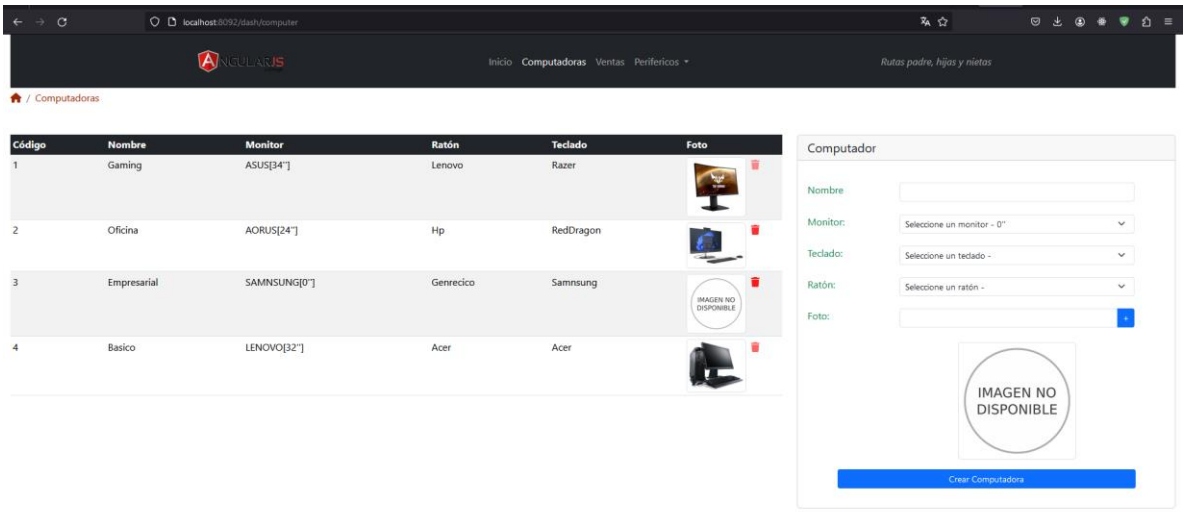
```
<ng-template #contenido>
  <div>
    <img
      class="img-thumbnail"
      onerror="this.src='../assets/images/noFoto.png'"
      [src]="ordenSeleccionada.comprador.base64Comprador"
    />
    <h6>
      <strong>Comprador:</strong>
      {{ordenSeleccionada.comprador.fullName()}}
    </h6>
    <h6>
      <strong>Documento:</strong>
        {{ordenSeleccionada.comprador.documento}}
    </h6>

    <h6>
      <strong>Cantidad:</strong>
      {{ordenSeleccionada.cantidad}}
    </h6>
    <h6>
      <strong>Total + <i>19% IVA</i>:</strong>
      USD ${{ordenSeleccionada.getTotal()}}
    </h6>
  </div>
  <div class="mt-2">
    <img
      class="img-thumbnail mt-2"
      style="max-height: 200px; width: auto;"
      onerror="this.src='../assets/images/noFoto.png'"
      [src]="ordenSeleccionada.computadoras.base64Computador"
    />
    <h6><strong>Computador :</strong> {{ ordenSeleccionada.computadoras.nombre}}</h6>
    <h6><strong>Perifericos :</strong><br>
      <strong>Monitor:</strong><p>{{
ordenSeleccionada.computadoras.monitor.toString()}}</p>
      <strong>Ratón:</strong><p>{{ ordenSeleccionada.computadoras.raton.toString()}}</p>
      <strong>Teclado:</strong><p>{{
ordenSeleccionada.computadoras.teclado.toString()}}</p>
    </h6>
    <h6><strong>Precio USD:</strong> {{ ordenSeleccionada.computadoras.getPrice!() }}</h6>
  </div>

</ng-template>
```

Tabla 29 Plantilla HTML orden.component.ts

Para verificar el resultado final de esta guía, se sugiere ejecutar en la termina de visual Studio Code del proyecto ejecute el siguiente comando.

```
ng serve
```

Imagen 67 Resultado final de la guia02



Fuente: Autor

5.27.1. Resultado

En el siguiente apartado podrá encontrar el resultado final, de cómo se debe ver y funcionar nuestra página web con los temas vistos para este material.

Click guia02

6. Reto para el lector

1. El próximo objetivo una vez terminado todo lo propuesto para esta guia03, como ingeniero es darle toda la funcionalidad CRUD al componente **Monitor**.
2. Como segundo objetivo se debe crear un nuevo componente con la funcionalidad para crear, editar y eliminar **Compradores**.
3. **Avanzado:** Se quiere seleccionar y almacenar una imagen base64 para los componentes Ratón y Teclado, su deber como ingeniero es de agregar esta nueva funcionalidad.

Enlace de los recursos propuestos

Titulo	Enlace
Guía 01	
Cómo Instalar Angular en Windows, macOS y Linux	https://kinsta.com/es/base-de-conocimiento/instalar-angular/#requisitos-previos-de-angular
Clases en Typescript	https://gustavodohara.com/blogangular/clases-en-typescript/
Herencia de clases en Typescript	https://gustavodohara.com/blogangular/herencia-de-clases-en-typescript/
Creando clases con constructores en Typescript	https://gustavodohara.com/blogangular/creando-clases-con-constructores-en-typescript/
Guía 02	
Interfaces en Typescript	https://gustavodohara.com/blogangular/interfaces-en-typescript/
Cómo manejar el router de Angular	https://codingpotions.com/angular-router/
Cómo usar el método de ngOnInit de Angular	https://codingpotions.com/angular-oninit
Generics en TypeScript	https://desarrolloweb.com/articulos/generics-typescript.html
Bootstrap Grid Generator	https://www.net-developer.nl/en/bootstrap-grid-generator.html
Documentación de Boostrap5.3	https://getbootstrap.com/docs/5.3/getting-star/introduction/