



# Tutorial de una aplicación básica – Rutas en Angular

PEV: Javier G. Palacios  
ARGENTINA PROGRAMA

## Prerrequisitos:

- Node.js

Revisa que node.js esté instalado en tu computador.

Si no está instalado descargar de <https://nodejs.org/en/download/>

Revisa la versión **node.js**:

```
node -v
```

**npm:**

```
npm -v
```

**Angular CLI** (es la herramienta con la que vamos a poder generar aplicaciones donde nos permite crear, depurar y publicar)

**Instalar Angular CLI:**

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

Deberías tener la última versión de Angular CLI.

## Visual Studio Code

Se puede descargar de la siguiente página: <https://code.visualstudio.com/>

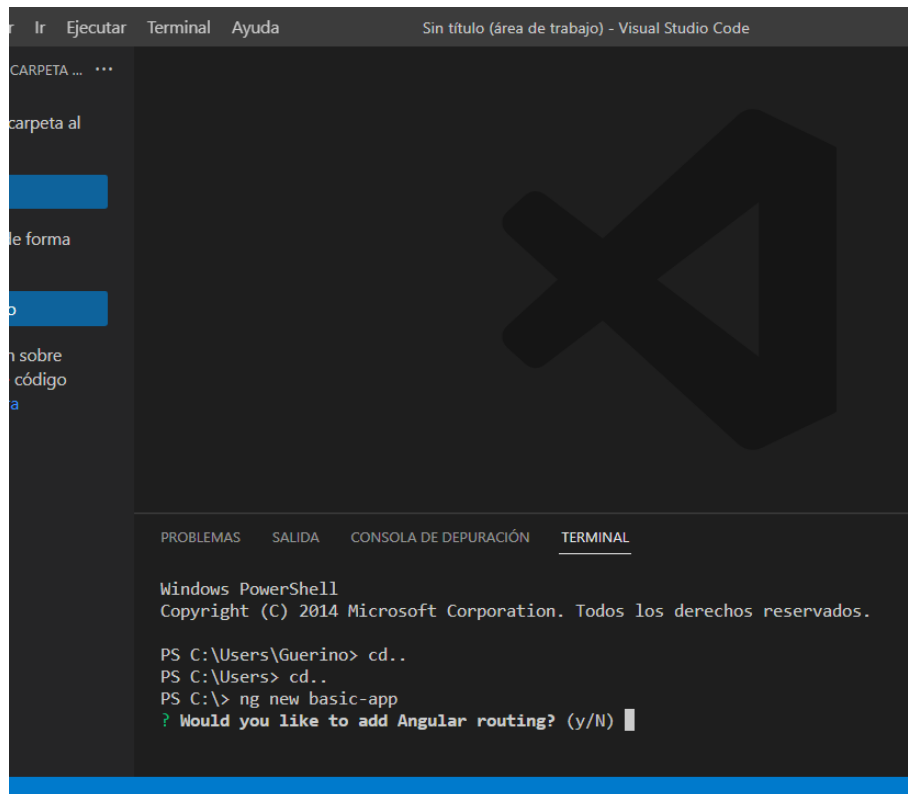
## Creando nuestra primera aplicación

Usaremos angular-cli para crear y generar nuestros componentes. Generará servicios, enrutadores, componentes y directivas.

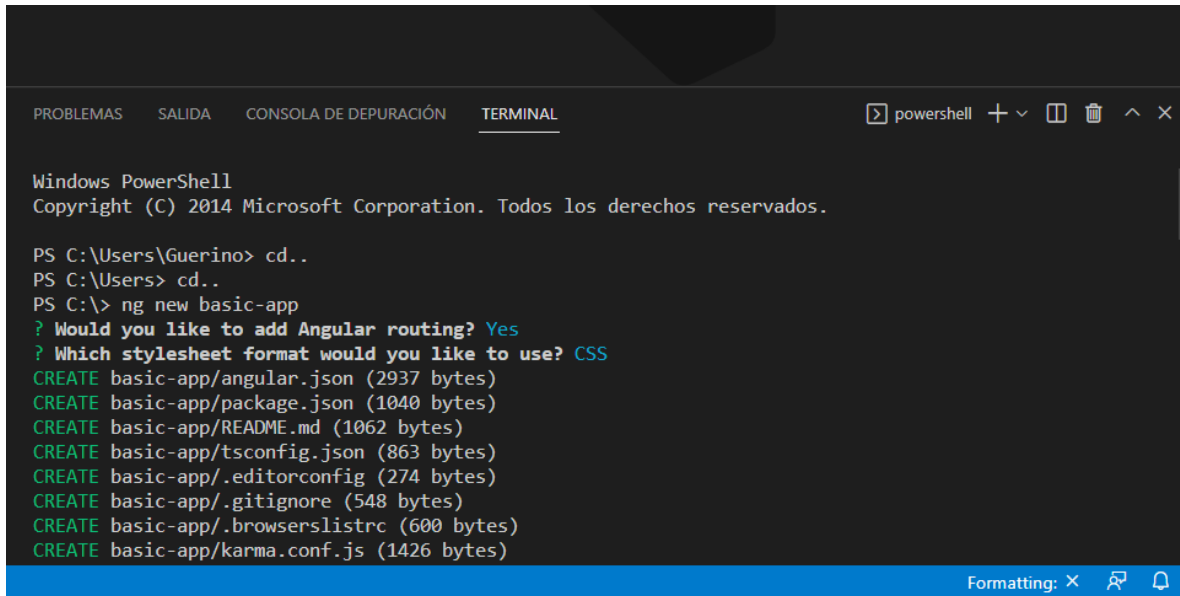
Para crear un nuevo proyecto Angular con Angular-cli, solo ejecuta:

```
ng new basic-app
```

El proyecto se generará automáticamente.



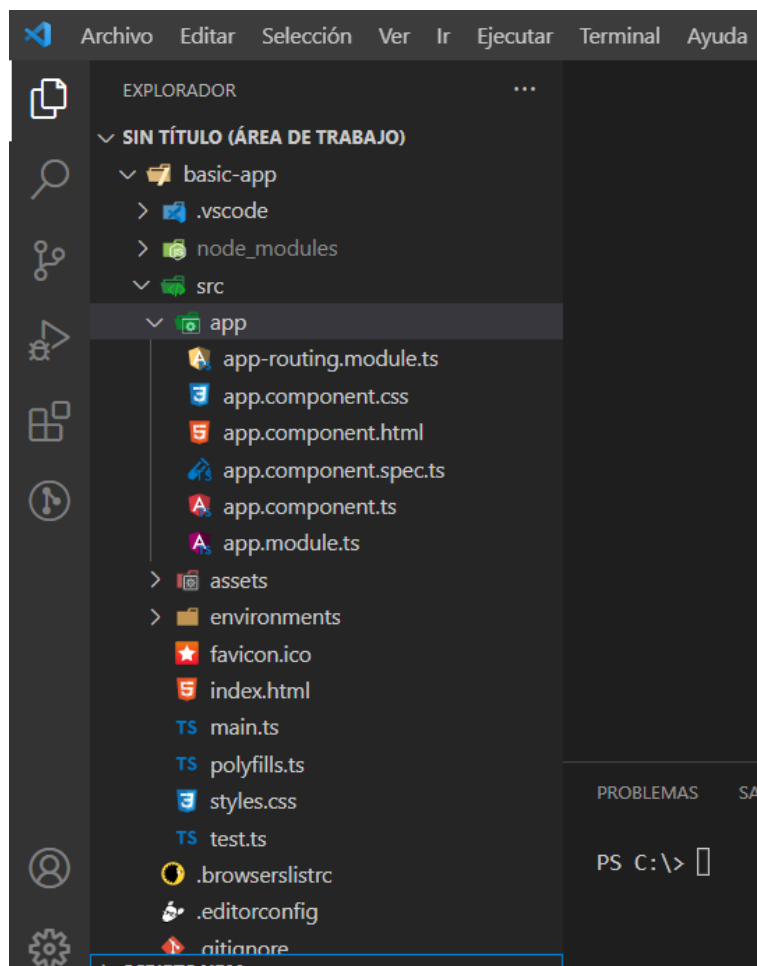
Ingresamos “y”, para agregar el componente que nos permite manejar las rutas en angular.



```
Windows PowerShell
Copyright (C) 2014 Microsoft Corporation. Todos los derechos reservados.

PS C:\Users\Guerino> cd..
PS C:\Users> cd..
PS C:\> ng new basic-app
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE basic-app/angular.json (2937 bytes)
CREATE basic-app/package.json (1040 bytes)
CREATE basic-app/README.md (1062 bytes)
CREATE basic-app/tsconfig.json (863 bytes)
CREATE basic-app/.editorconfig (274 bytes)
CREATE basic-app/.gitignore (548 bytes)
CREATE basic-app/.browserslistrc (600 bytes)
CREATE basic-app/karma.conf.js (1426 bytes)
```

Luego abrimos la carpeta creada con nuestra aplicación en visual studio code.

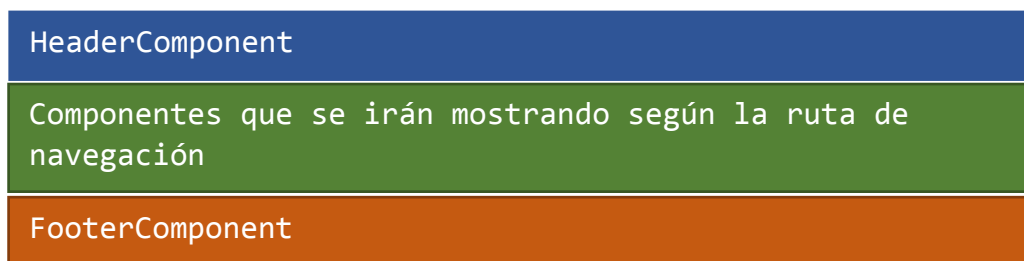


Para tener en cuenta, nuestra aplicación estará dividida en los siguientes componentes:

- Header
- Login
- Hello
- Dashboard
- Footer

De los cuales el Header y el Footer component, siempre se mostrarán y según como cambien las rutas en la barra de direcciones del navegador, se irán mostrando los otros componentes en pantalla.

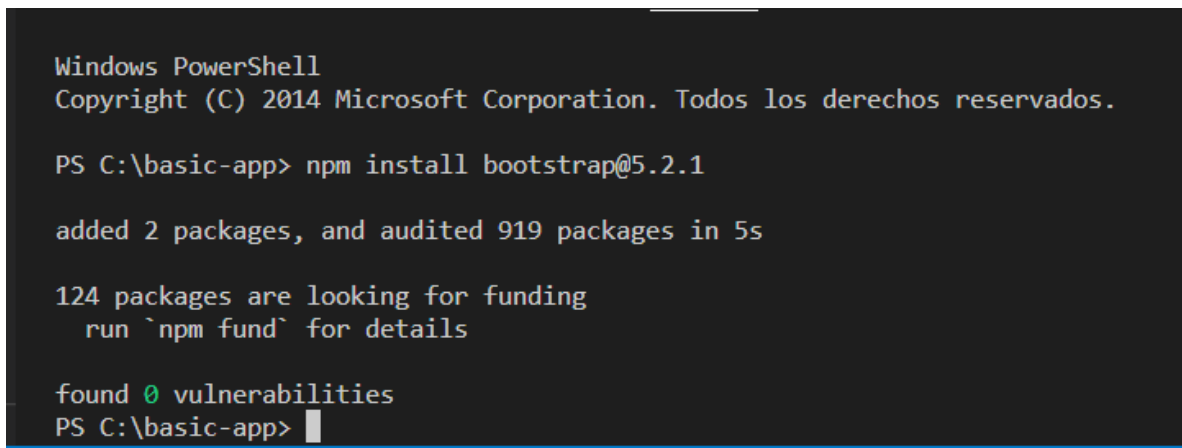
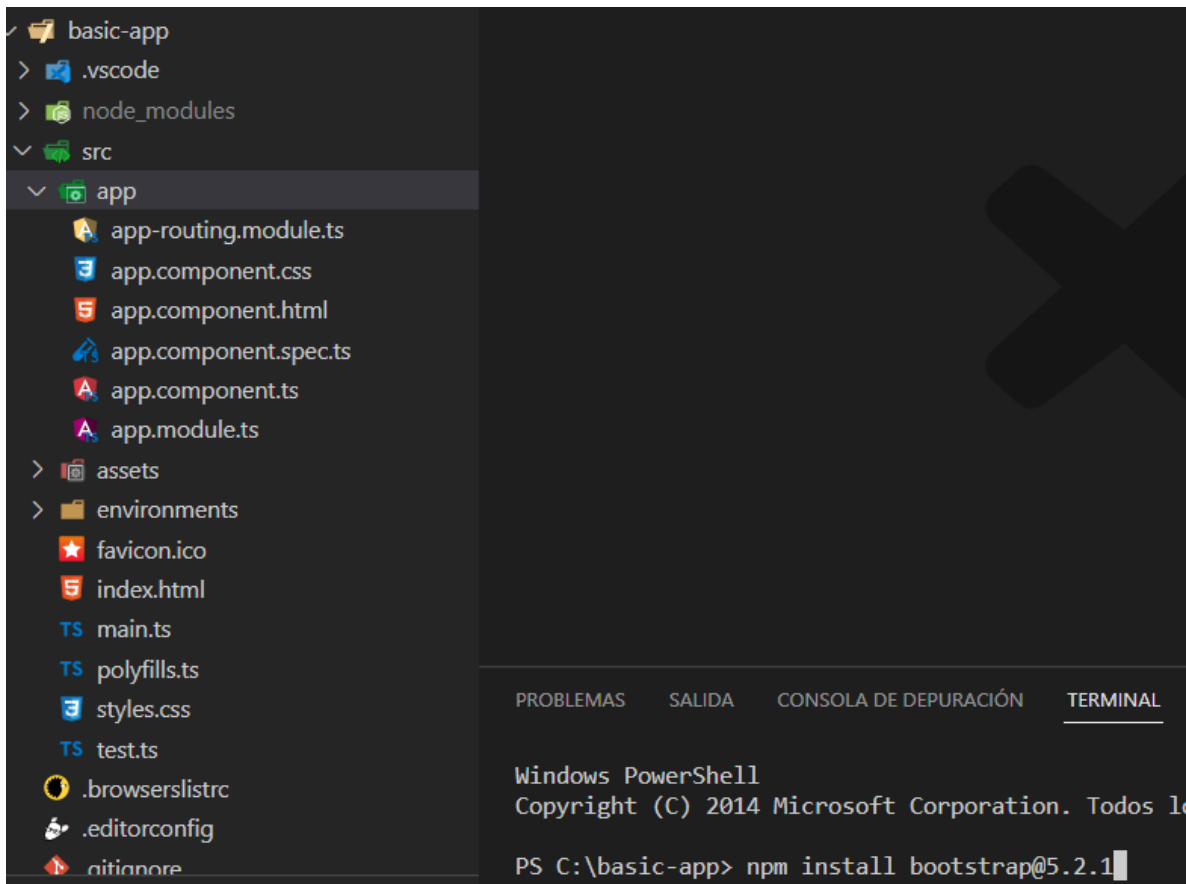
Ósea en pantalla se verá así:



Antes de continuar instalaremos Bootstrap para los estilos de nuestra aplicación:

Usaremos el siguiente comando (actualmente la última versión es la 5.2.1):

```
npm install bootstrap@5.2.1
```



A continuación, crearemos los cinco componentes nombrados anteriormente y lo haremos dentro de la carpeta components:

```
ng generate component components/header
```

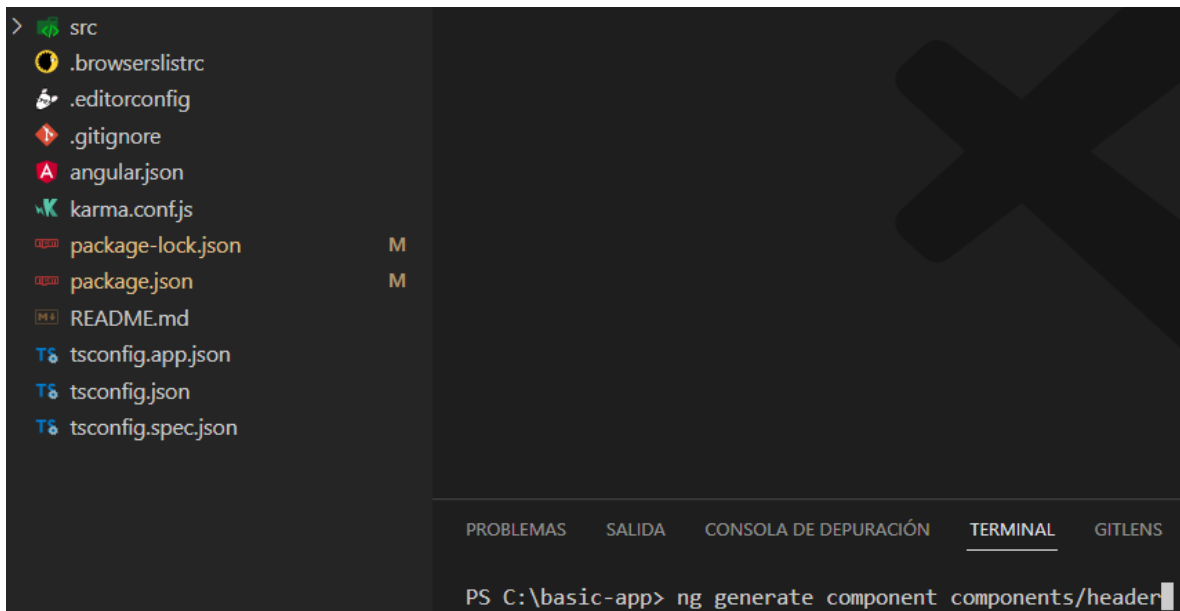
```
ng generate component components/login
```

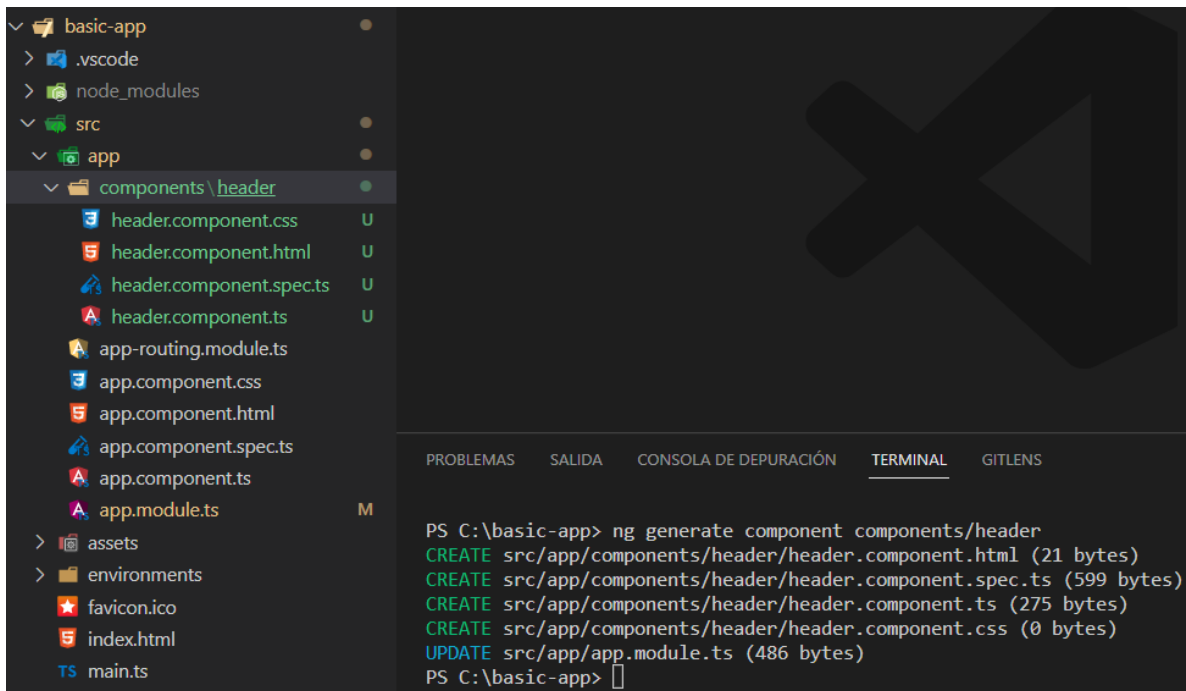
```
ng generate component components/hello
```

```
ng generate component components/dashboard
```

```
ng generate component components/footer
```

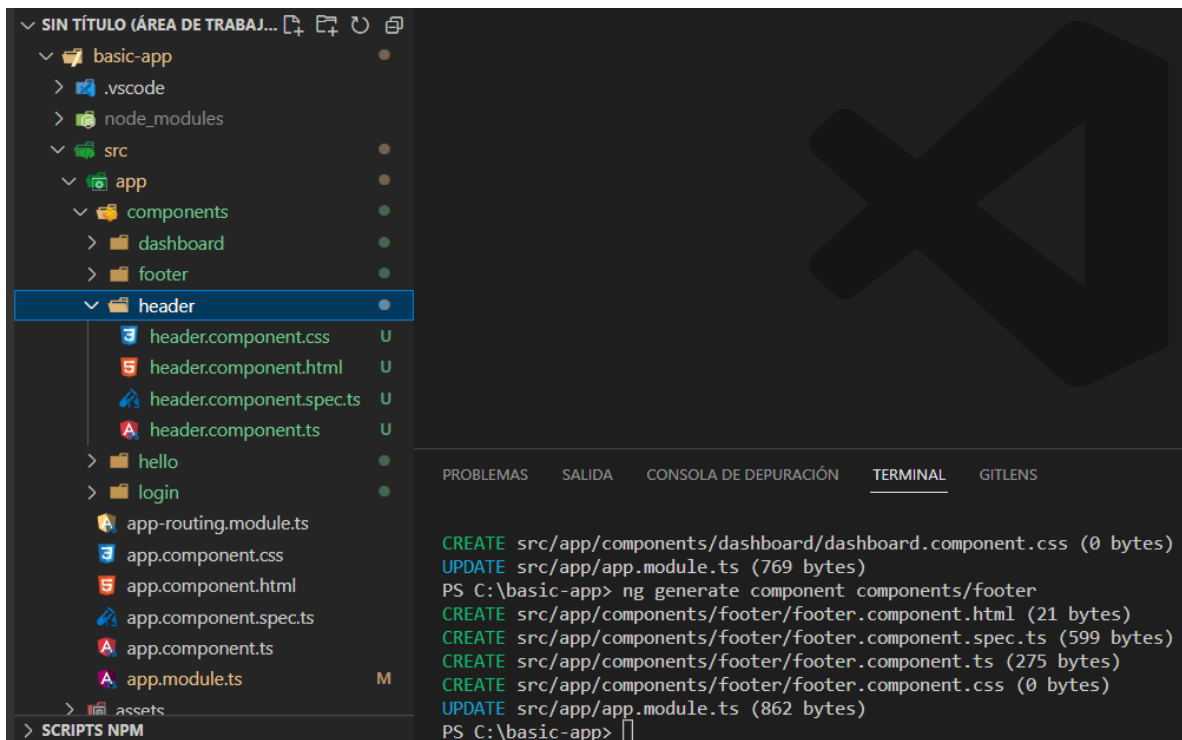
Para el primer componente:





```
PS C:\basic-app> ng generate component components/header
CREATE src/app/components/header/header.component.html (21 bytes)
CREATE src/app/components/header/header.component.spec.ts (599 bytes)
CREATE src/app/components/header/header.component.ts (275 bytes)
CREATE src/app/components/header/header.component.css (0 bytes)
UPDATE src/app/app.module.ts (486 bytes)
PS C:\basic-app>
```

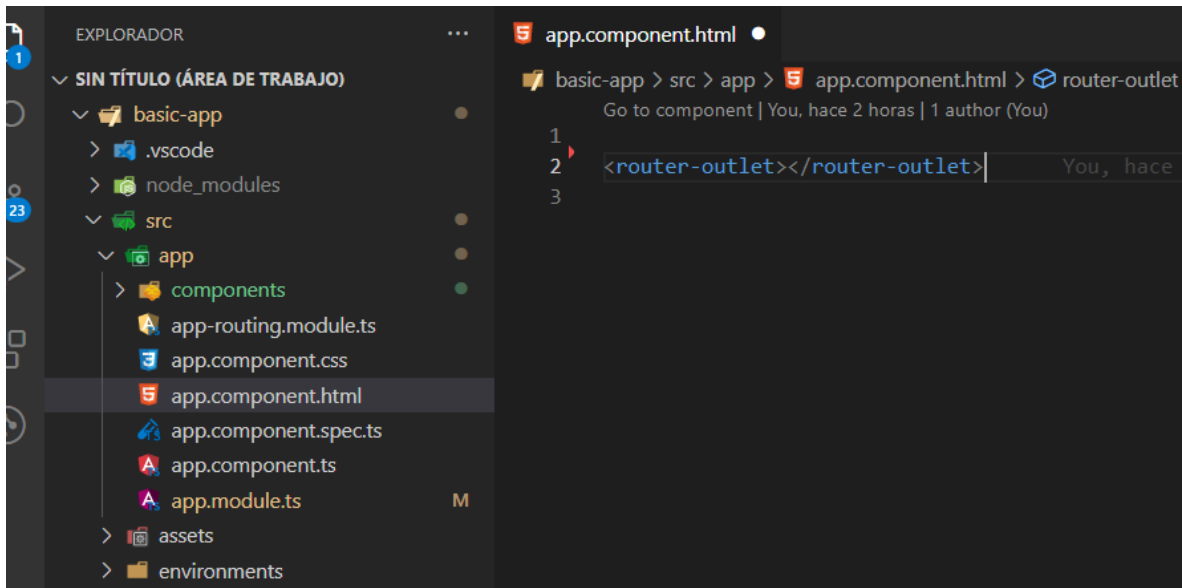
Quedando así nuestra aplicación con los cinco componentes creados:



```
CREATE src/app/components/dashboard/dashboard.component.css (0 bytes)
UPDATE src/app/app.module.ts (769 bytes)
PS C:\basic-app> ng generate component components/footer
CREATE src/app/components/footer/footer.component.html (21 bytes)
CREATE src/app/components/footer/footer.component.spec.ts (599 bytes)
CREATE src/app/components/footer/footer.component.ts (275 bytes)
CREATE src/app/components/footer/footer.component.css (0 bytes)
UPDATE src/app/app.module.ts (862 bytes)
PS C:\basic-app>
```



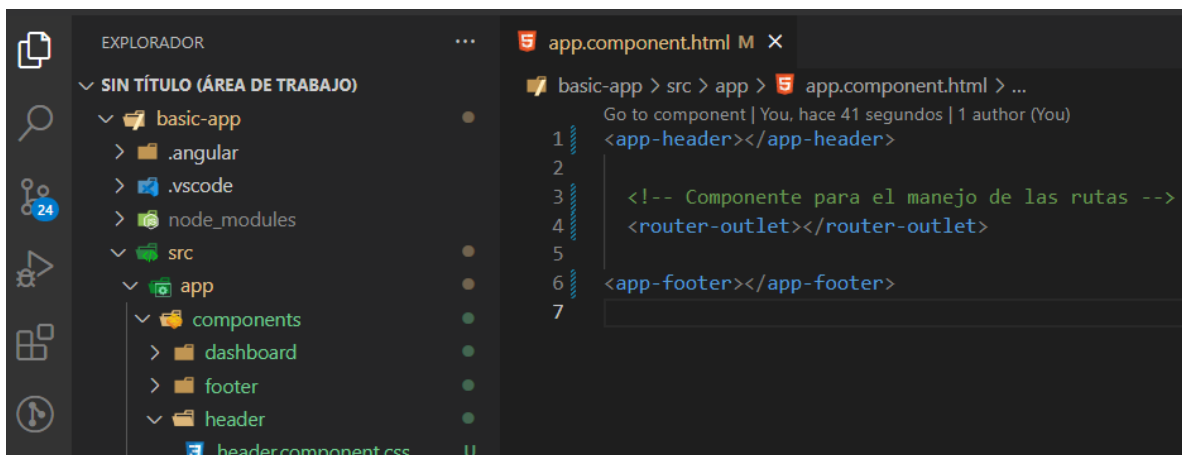
Lo siguiente va a ser limpiar el archivo **app.component.html**, eliminando todo el código en ese archivo, menos las etiquetas del router componet de angular, entonces nos queda asi:



```
1  
2 <router-outlet></router-outlet>  
3
```

El componente `<router-outlet></router-outlet>` es el que va realizar el cambio de componentes en pantalla según la ruta que tengamos en la barra de direcciones del navegador web del usuario.

Ahora, vamos a armar la estructura o diseño (en ingles se lo suele llamar layout) de nuestra aplicación según como vimos anteriormente en la página 4.

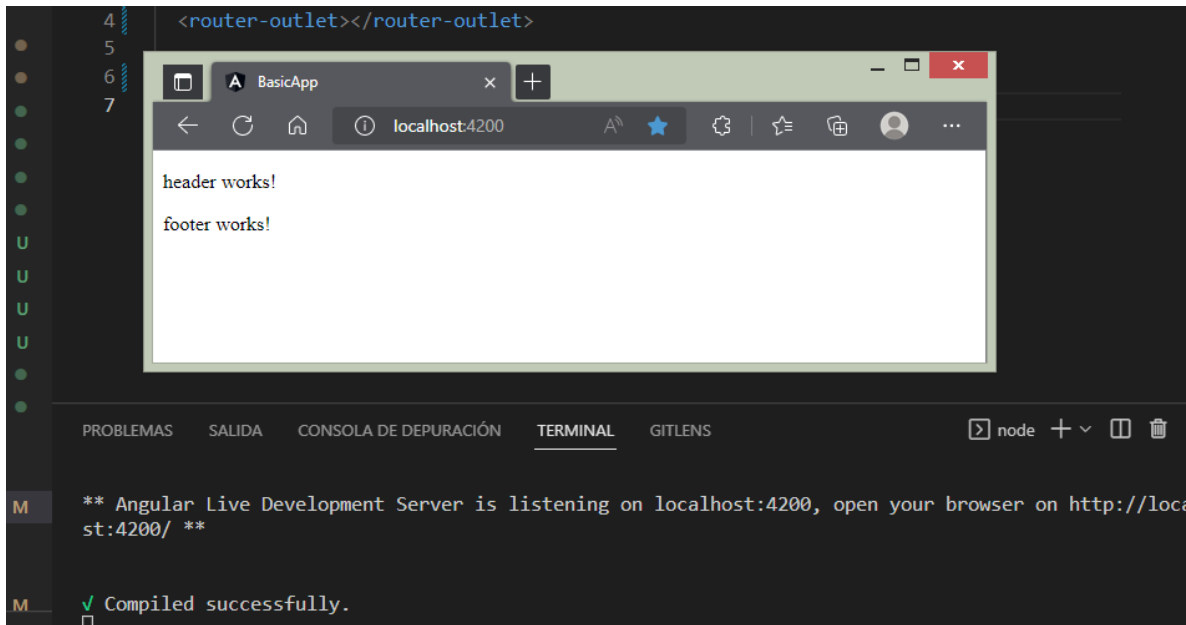


```
1 <app-header></app-header>  
2  
3 <!-- Componente para el manejo de las rutas -->  
4 <router-outlet></router-outlet>  
5  
6 <app-footer></app-footer>  
7
```

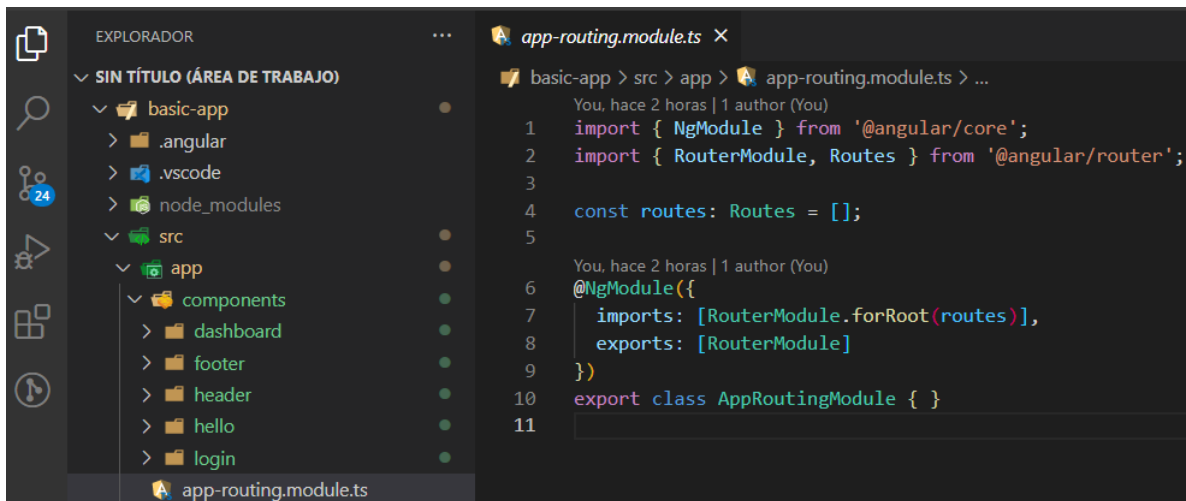
Si en este punto ejecutamos el siguiente comando:

```
ng serve
```

Veremos en la dirección: <http://localhost:4200/> en nuestro navegador web la siguiente pantalla.



El siguiente paso es ir al archivo **app-routing.module.ts**



Allí adentro en donde dice:

```
const routes: Routes = [];
```

agregaremos las rutas a los componentes que se mostraran en pantalla.

Primero debemos importarlos:

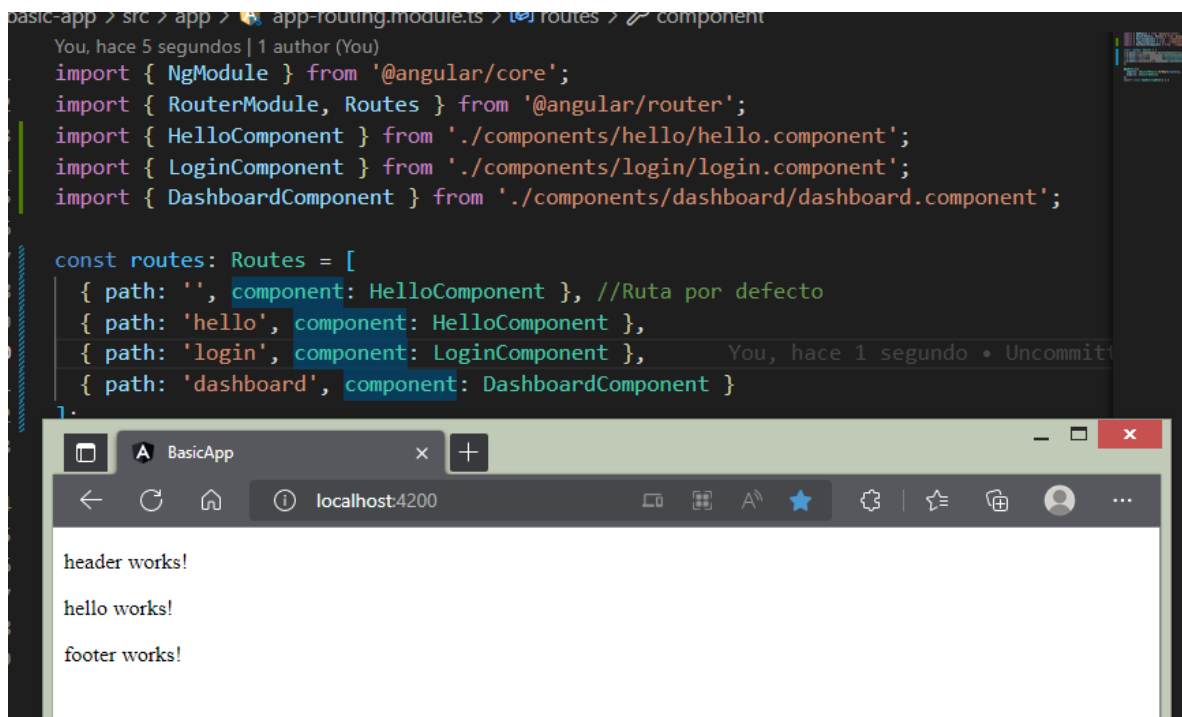
```
... app-routing.module.ts M X
basic-app > src > app > app-routing.module.ts > ...
You, hace 1 segundo | 1 author (You)
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { HelloComponent } from './components/hello/hello.component';
4 import { LoginComponent } from './components/login/login.component';
5 import { DashboardComponent } from './components/dashboard/dashboard.component';
6
7 const routes: Routes = [
8
9 ];
10
11 @NgModule({
12   imports: [RouterModule.forRoot(routes)],
13   exports: [RouterModule]
14 })
15 export class AppRoutingModule { }
```

Luego agregamos las rutas, una ruta por cada componente que queremos mostrar, teniendo en cuenta que:

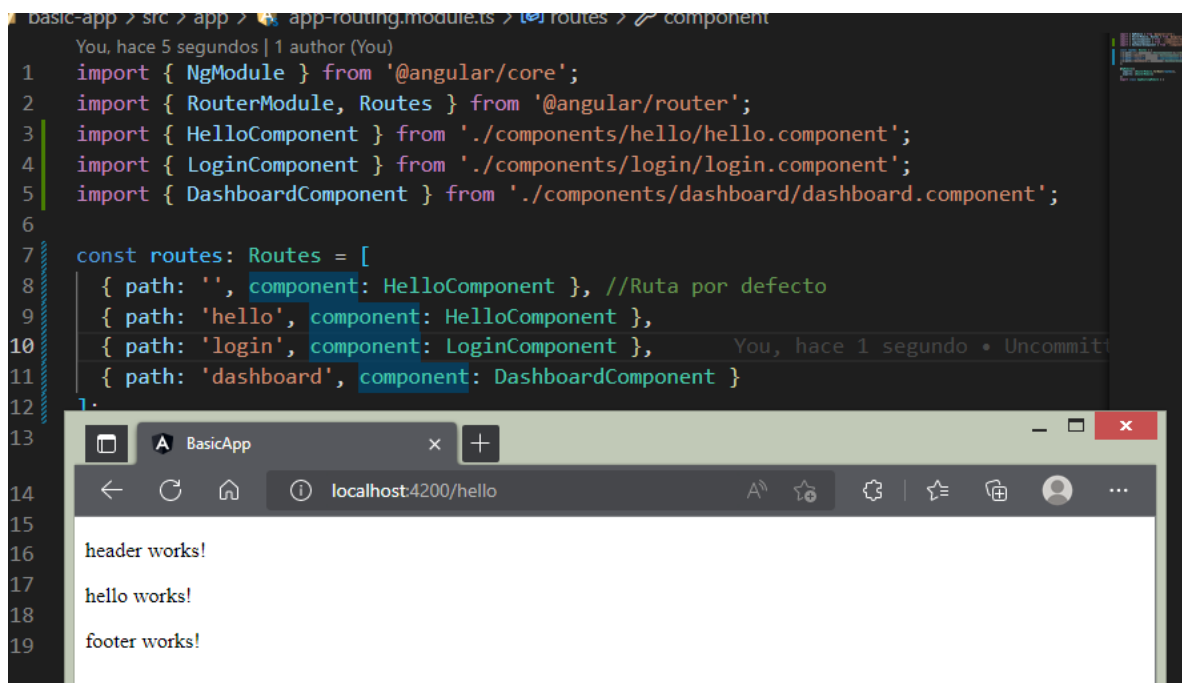
- **path**: define la ruta virtual de nuestra aplicación.
- **component**: define el componente que le dice al enrutador que componente corresponde al seleccionar dicha ruta.

```
app-routing.module.ts M X
basic-app > src > app > app-routing.module.ts > ...
You, ahora | 1 author (You)
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { HelloComponent } from './components/hello/hello.component';
4 import { LoginComponent } from './components/login/login.component';
5 import { DashboardComponent } from './components/dashboard/dashboard.component';
6
7 const routes: Routes = [
8   { path: 'hello', component: HelloComponent },
9   { path: 'login', component: LoginComponent },
10  { path: 'dashboard', component: DashboardComponent }
11 ];
12
13 @NgModule({
14   imports: [RouterModule.forRoot(routes)],
15   exports: [RouterModule]
16 })
17 export class AppRoutingModule { }
```

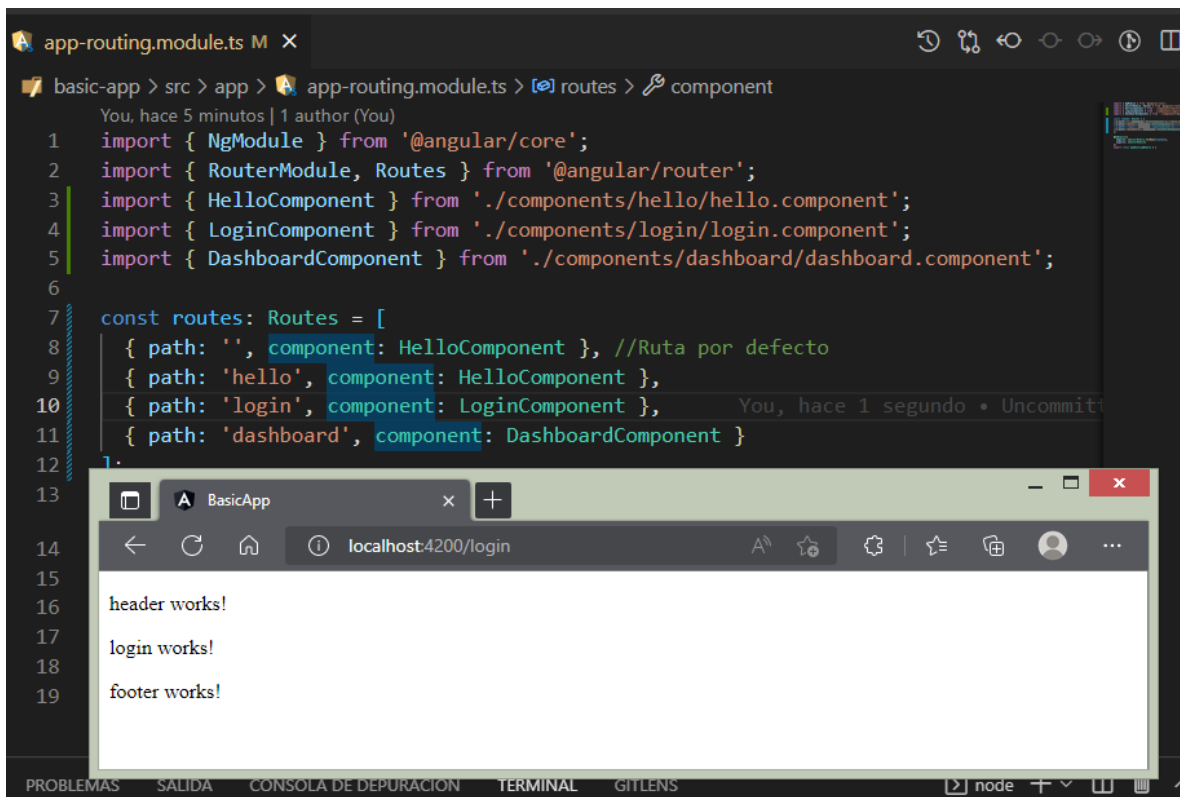
Bien, ahora que ya tenemos las rutas y sus componentes asociados registrados en el módulo **app-routing.module.ts**, podemos probarlas en el navegador web:



Como el `HelloComponent` es el componente por defecto, ósea el que se muestra sin haber ingresado ninguna ruta en la barra de direcciones, angular procede a cargarlo y mostrarlo en pantalla, pero como también está registrada la ruta 'hello', si la ingresamos se muestra la misma pantalla:



Para los otros dos componentes se muestra así:



```
app-routing.module.ts M X
basic-app > src > app > app-routing.module.ts > routes > component
You, hace 5 minutos | 1 author (You)
1 import { NgModule } from '@angular/core';
2 import { RouterModule, Routes } from '@angular/router';
3 import { HelloComponent } from './components/hello/hello.component';
4 import { LoginComponent } from './components/login/login.component';
5 import { DashboardComponent } from './components/dashboard/dashboard.component';
6
7 const routes: Routes = [
8   { path: '', component: HelloComponent }, //Ruta por defecto
9   { path: 'hello', component: HelloComponent },
10  { path: 'login', component: LoginComponent },
11  { path: 'dashboard', component: DashboardComponent }
12 ]
13
14
15
16
17
18
19
```

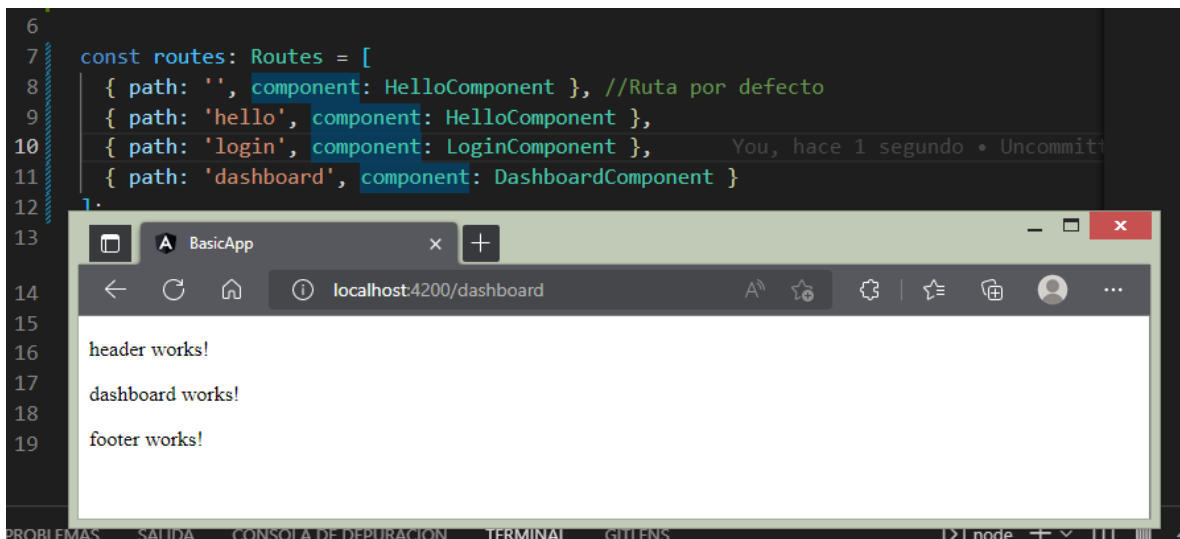
BasicApp

localhost:4200/login

header works!

login works!

footer works!



```
6
7 const routes: Routes = [
8   { path: '', component: HelloComponent }, //Ruta por defecto
9   { path: 'hello', component: HelloComponent },
10  { path: 'login', component: LoginComponent },
11  { path: 'dashboard', component: DashboardComponent }
12 ]
13
14
15
16
17
18
19
```

BasicApp

localhost:4200/dashboard

header works!

dashboard works!

footer works!

Otra forma de crear la **ruta por defectos** es así:

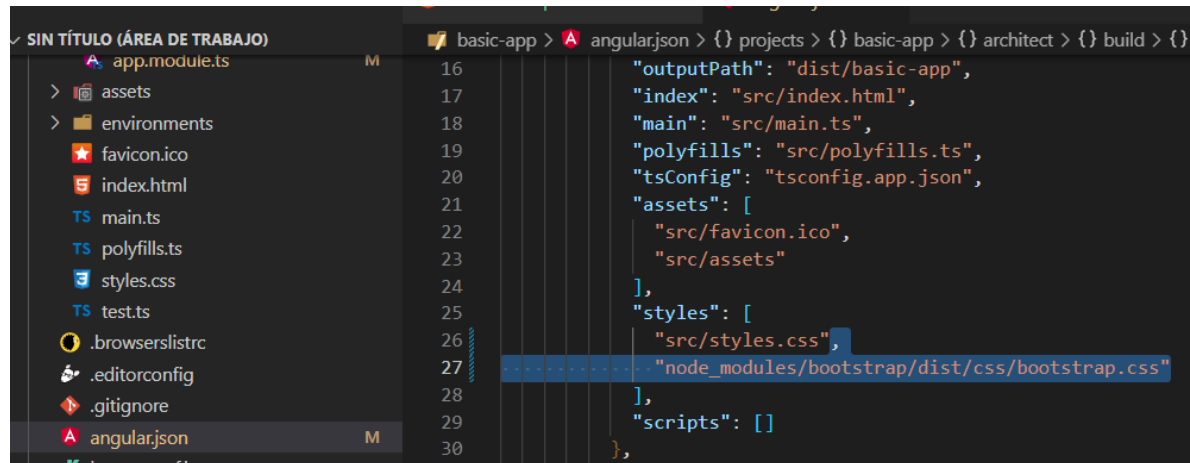
```
{ path: '', redirectTo: 'hello', pathMatch: 'full' }
```

Con **redirectTo** redireccionamos a la ruta que nosotros definimos como componente de inicio.

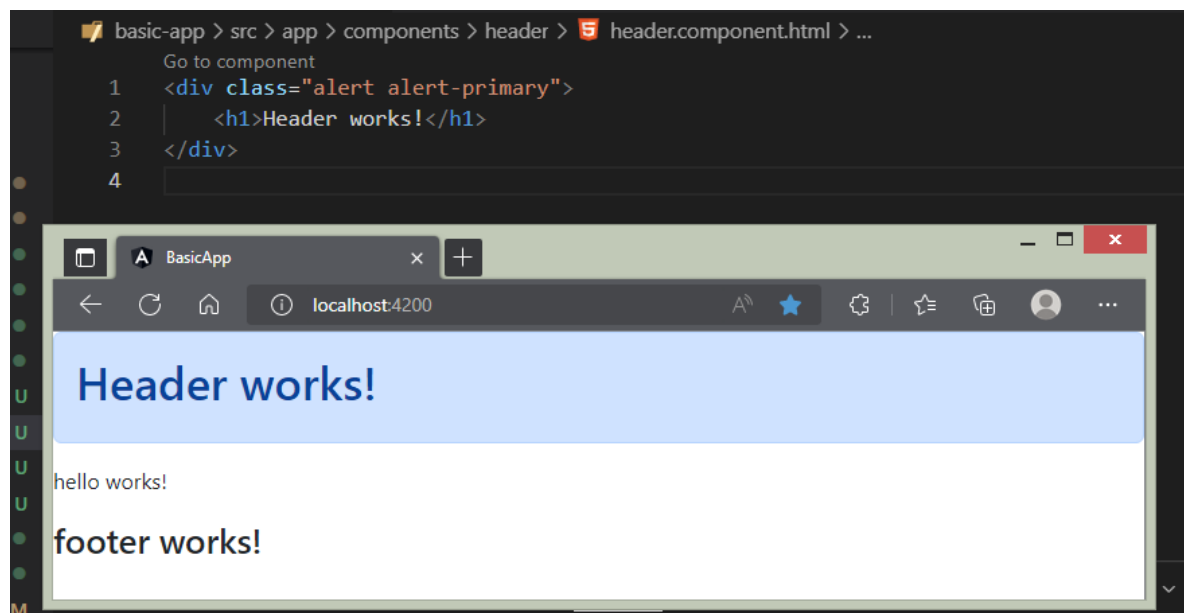
Antes de poder aplicar estilos, debemos agregar en el archivo **angular.json** en la sección “styles”, el archivo css de bootstrap:

```
"styles": [  
  "src/styles.css",  
  "node_modules/bootstrap/dist/css/bootstrap.css"  
],
```

Debe quedar así:



Luego detenemos el servidor con **ctrl+c**, y lo volvemos a iniciar con: **ng serve**



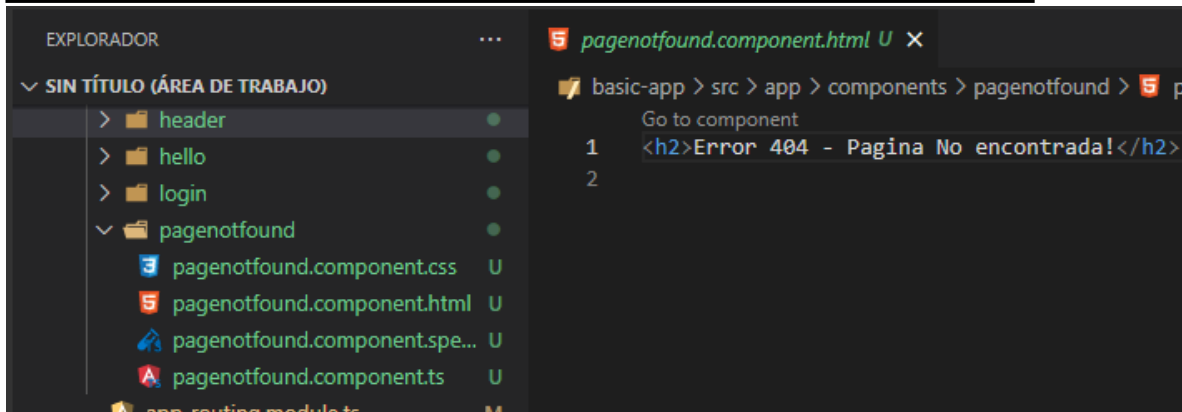
Hay que reiniciar el servidor para que cargue bootstrap.

Fin de este mini tutorial de rutas sin parámetros en angular.

## Rutas con parámetros

Ahora vamos a mejorar nuestra aplicación, primero vamos a crear un nuevo componente para el error 404.

```
ng generate component components/pagenotfound
```



Este mensaje aparecerá en pantalla cuando se ingrese en la barra de direcciones del navegador una ruta que no esta registrada en el `app-routing.module.ts`

Luego, agregamos la ruta en el **`app-routing.module.ts`**



```
//Componente para las rutas no encontradas - Error 404
{ path: '**', pathMatch: 'full', component: PagenotfoundComponent }
```

## Rutas dinámicas

Las rutas con parámetros son aquellas que son aquellas llamadas **rutas dinámicas** porque una parte es fija, pero en la otra parte van cambiando el valor de las variables que se le pasan a un determinado componente.

Ejemplo:

Definimos que vamos a utilizar **tres variables** (id, nombre y edad) que se le van a pasar a un componente llamado **UserComponent**, y su ruta es /user:

<http://localhost:4200/user/:id/:nombre/:edad>

luego en un enlace quedaría así:

```
<a routerLink="/user/1/Juan/35" class="link">Ver usuario</a>
```

**Nota:** el valor de estas variables generalmente viene de la base datos, atreves de una API Rest.

Creamos un componente User para practicar:

```
ng generate component components/user
```



Agregamos un nuevo enlace a nuestra nav:

```
<div class="alert alert-primary">
  <h1>Encabezado de página</h1>

  <nav class="navbar navbar-expand-lg bg-light">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link" routerLink="/">Inicio</a>
      </li>

      <li class="nav-item">
        <a class="nav-link" routerLink="/hello">Hello</a>
      </li>

      <li class="nav-item">
        <a class="nav-link" routerLink="/login">Login</a>
      </li>

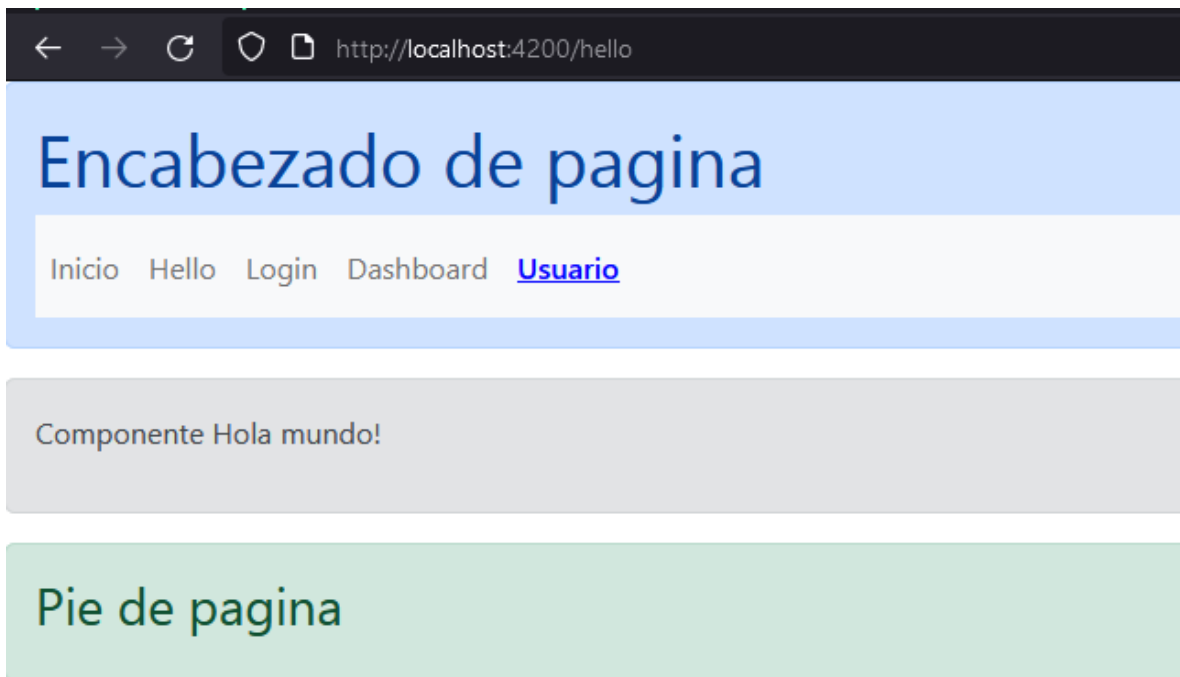
      <li class="nav-item">
        <a class="nav-link" routerLink="/dashboard">Dashboard</a>
      </li>

      <li class="nav-item">
        <a class="nav-link" routerLink="/user/1/Juan/35">Usuario</a>
      </li>
    </ul>
  </nav>
</div>
```

Y luego agregamos la ruta en el **app-routing.module.ts**:

```
//Rutas de nuestra aplicacion
const routes: Routes = [
  { path: '', redirectTo: 'hello', pathMatch: 'full'}, //Ruta por defecto
  { path: 'hello', component: HelloComponent },
  { path: 'login', component: LoginComponent },
  { path: 'dashboard', component: DashboardComponent },
  { path: 'user/:id/:nombre/:edad', component: UserComponent },
  //Componente para las rutas no encontradas - Error 404
  { path: '**', pathMatch: 'full', component: PagenotfoundComponent }
];
```

En pantalla nos queda así la página:



Ahora recibiremos los parámetros del Usuario en el componente UserComponent:

En user.component.html agregamos:

```
<h4>Hola usuario:</h4>
<p>id: {{ getId() }}</p>
<p>Nombre: {{ getName() }}</p>
<p>Edad: {{ getAge() }}</p>
```

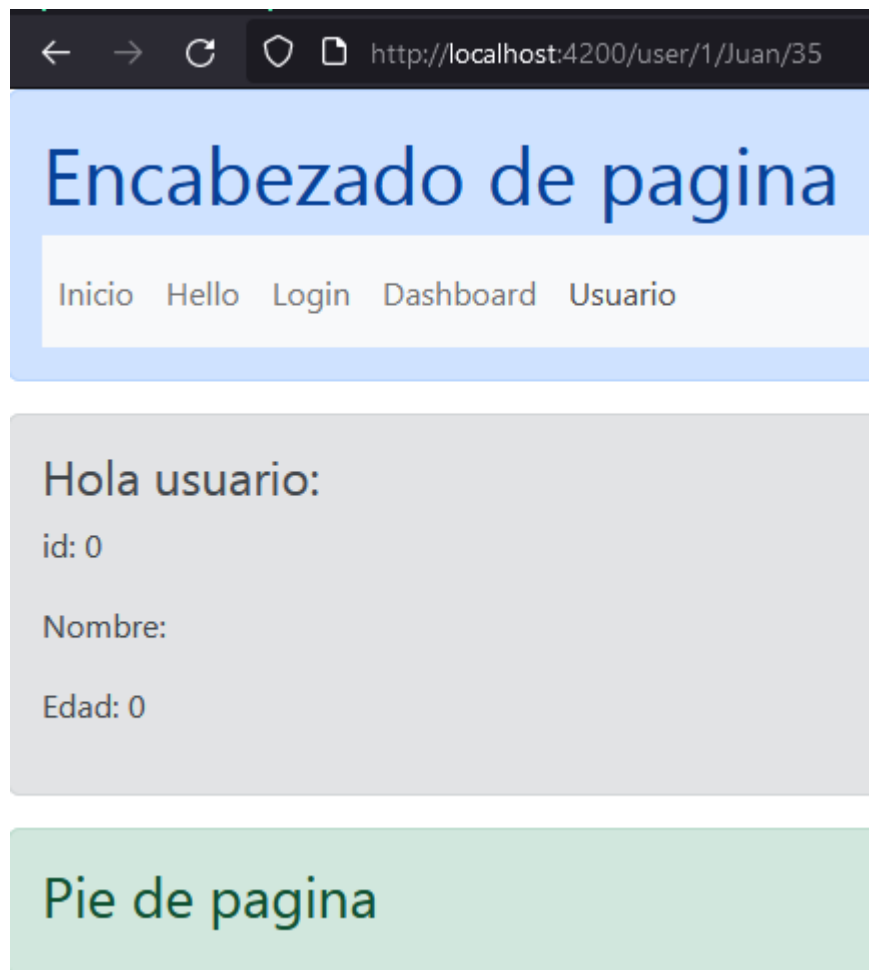
Y en user.components.ts agregamos 3 atributos, los inicializamos y creamos los métodos **get** correspondientes a cada uno:

```
export class UserComponent implements OnInit {
  private id:number;
  private name:string;
  private age:number;

  constructor() {
    //Inicializamos los atributos
    this.id = 0;
    this.name = "";
    this.age = 0;
  }
}
```

```
ngOnInit(): void {  
  
}  
  
public getId(): number {  
    return this.id;  
}  
  
public getName(): string {  
    return this.name;  
}  
  
public getAge(): number {  
    return this.age;  
}  
}
```

Ahora en pantalla se ve así:



Ahora, vamos a proceder a capturar los parámetros pasados al UserComponent en la ruta:

Primero importamos:

```
import { ActivatedRoute, Params } from '@angular/router';
```

luego inyectamos la dependencia en el constructor:

```
constructor(private ruta: ActivatedRoute) {  
  //Inicializamos los atributos  
  this.id = 0;  
  this.name = "";  
  this.age = 0;  
}
```

Y procedemos a capturar los datos y mostrarlos en pantalla:

```
ngOnInit(): void {  
  this.ruta.params.subscribe((params: Params) => {  
    this.id = params['id'];  
    this.name = params['nombre'];  
    this.age = params['edad'];  
  });  
}
```

