



TECNOLÓGICO NACIONAL DE MÉXICO

INSTITUTO TECNOLÓGICO DE MILPA ALTA II

ASIGNATURA:

MEAN STACK FOR FRONT – END

TEMA:

Examen Unidad 4

ELABORA:

POZOS RIVERA ALEJANDRO

NÚMERO DE CONTROL:

171190025

PROFESOR:

ROLDAN AQUINO SEGURA

09 DE JUNIO DE 2021

Practica 1 Módulos: creación y consumo

Como primer paso nos vamos a dirigir a nuestra consola de comandos y poner la ruta donde se albergará el proyecto de angular.

```
Selecc... Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.985]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos
```

Una vez que ya tengamos definida la ruta de la carpeta introduciremos la siguiente línea de comando para poder generar el proyecto de angular “ng new modulos”. Y nos pregunta “¿Le gustaría agregar enrutamiento angular?” y solo pondremos la letra “y”.

```
npm
Microsoft Windows [Versión 10.0.19041.985]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos

C:\Users\Admin\Documents\AngularProyectos>ng new modulos
? Would you like to add Angular routing? (y/N)
```

Después nos va a preguntar con que tipos de estilos trabajaremos y escogeremos “css”. Y empezara a crear el proyecto.

```
C:\Users\Admin\Documents\AngularProyectos>ng new modulos
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
SCSS   [ https://sass-lang.com/documentation/syntax#scss ]
Sass   [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
Less   [ http://lesscss.org ]
```

```
C:\Users\Admin\Documents\AngularProyectos>ng new modulos
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE modulos/angular.json (3045 bytes)
CREATE modulos/package.json (1069 bytes)
CREATE modulos/README.md (1053 bytes)
CREATE modulos/tsconfig.json (783 bytes)
CREATE modulos/.editorconfig (274 bytes)
CREATE modulos/.gitignore (604 bytes)
CREATE modulos/.browserslistrc (703 bytes)
CREATE modulos/karma.conf.js (1424 bytes)
CREATE modulos/tsconfig.app.json (287 bytes)
CREATE modulos/tsconfig.spec.json (333 bytes)
CREATE modulos/src/favicon.ico (948 bytes)
CREATE modulos/src/index.html (293 bytes)
CREATE modulos/src/main.ts (372 bytes)
CREATE modulos/src/polyfills.ts (2820 bytes)
CREATE modulos/src/styles.css (80 bytes)
CREATE modulos/src/test.ts (743 bytes)
CREATE modulos/src/assets/.gitkeep (0 bytes)
CREATE modulos/src/environments/environment.prod.ts (51 bytes)
CREATE modulos/src/environments/environment.ts (658 bytes)
CREATE modulos/src/app/app-routing.module.ts (245 bytes)
CREATE modulos/src/app/app.module.ts (393 bytes)
CREATE modulos/src/app/app.component.html (23809 bytes)
CREATE modulos/src/app/app.component.spec.ts (1060 bytes)
CREATE modulos/src/app/app.component.ts (211 bytes)
CREATE modulos/src/app/app.component.css (0 bytes)
/ Installing packages (npm)...
```

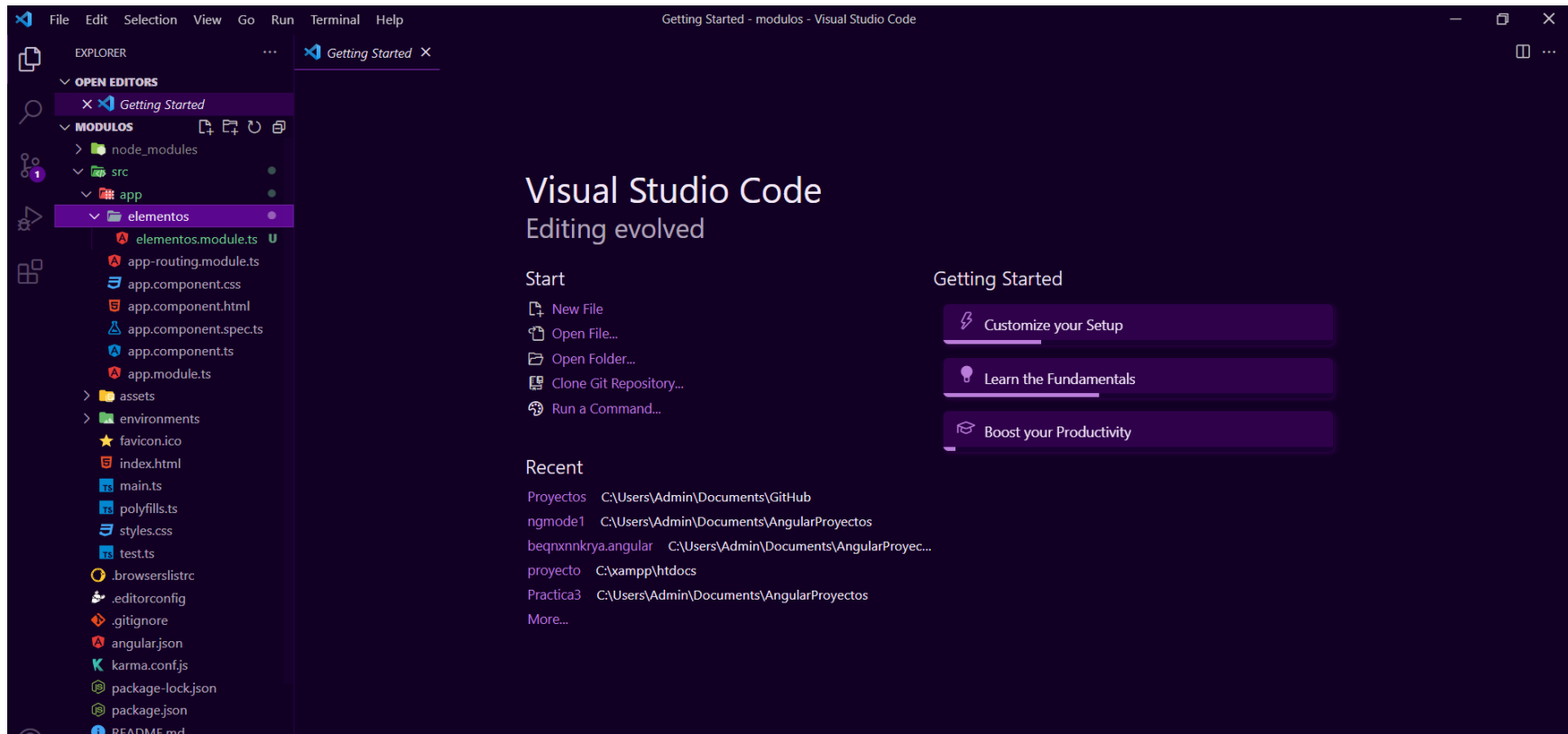
Primero descendemos a la carpeta modulos y nuevamente desde la línea de comandos procedemos a crear el módulo 'elementos' escribiendo: "ng generate module elementos".

```
C:\Users\Admin\Documents\AngularProyectos>cd modulos

C:\Users\Admin\Documents\AngularProyectos\modulos>ng generate module elementos
CREATE src/app/elementos/elementos.module.ts (195 bytes)

C:\Users\Admin\Documents\AngularProyectos\modulos>_
```

Abriremos nuestro editor de código en mi casi Visual Studio Code, y procederemos a abrir el proyecto “modulos” que creamos, el cual dentro del mismo existe la carpeta del módulo llamada “elementos”.



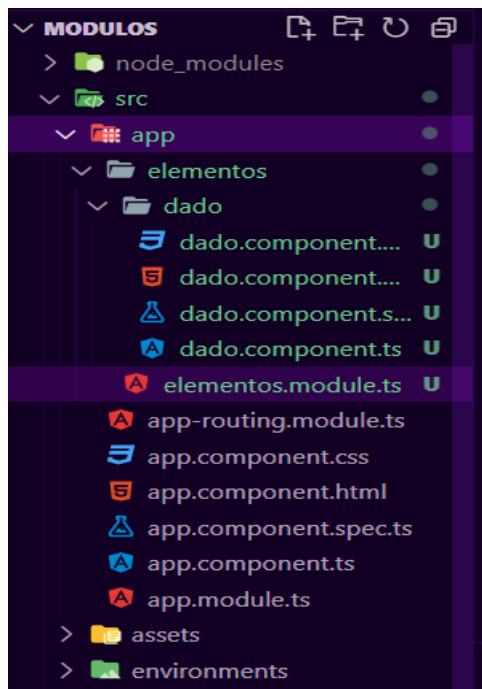
Contenido del archivo “elementos.module.ts”.

```
src > app > elementos > elementos.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3
4
5
6  @NgModule({
7    declarations: [],
8    imports: [
9      CommonModule
10   ]
11 })
12 export class ElementosModule { }
```

Ahora vamos a crear el componente dado almacenando la misma en el módulo elementos (no en el módulo principal): con la siguiente línea de comando “ng generate component elementos/dado”.

```
PS C:\Users\Admin\Documents\AngularProyectos\modulos> ng generate component elementos/dado
CREATE src/app/elementos/dado/dado.component.html (19 bytes)
CREATE src/app/elementos/dado/dado.component.spec.ts (612 bytes)
CREATE src/app/elementos/dado/dado.component.ts (267 bytes)
CREATE src/app/elementos/dado/dado.component.css (0 bytes)
UPDATE src/app/elementos/elementos.module.ts (271 bytes)
PS C:\Users\Admin\Documents\AngularProyectos\modulos>
```

Como ya sabemos con este comando se crean los cuatro archivos de la componente y se modifica el archivo 'elementos.module.ts', es importante anteceder al nombre de la componente el módulo donde se debe registrar:



Si abrimos nuevamente el archivo 'elementos.module.ts' podemos ver que la componente 'DadoComponent' se encuentra registrada:

```
src > app > elementos > elementos.module.ts > ElementosModule
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { DadoComponent } from '../dado/dado.component';
4
5
6
7  @NgModule({
8    declarations: [
9      DadoComponent
10   ],
11   imports: [
12     CommonModule
13   ]
14 })
15 export class ElementosModule { }
```

Codificamos los archivos de la componente 'dado' dado.component.ts.

```
src > app > elementos > dado > dado.component.ts > ...
1  import { Component, OnInit, Input } from '@angular/core';
2
3  @Component({
4    selector: 'app-dado',
5    templateUrl: './dado.component.html',
6    styleUrls: ['./dado.component.css']
7  })
8  export class DadoComponent implements OnInit {
9
10   @Input() valor: number = 0;
11
12   constructor() { }
13
14   ngOnInit(): void {
15   }
16
17 }
```

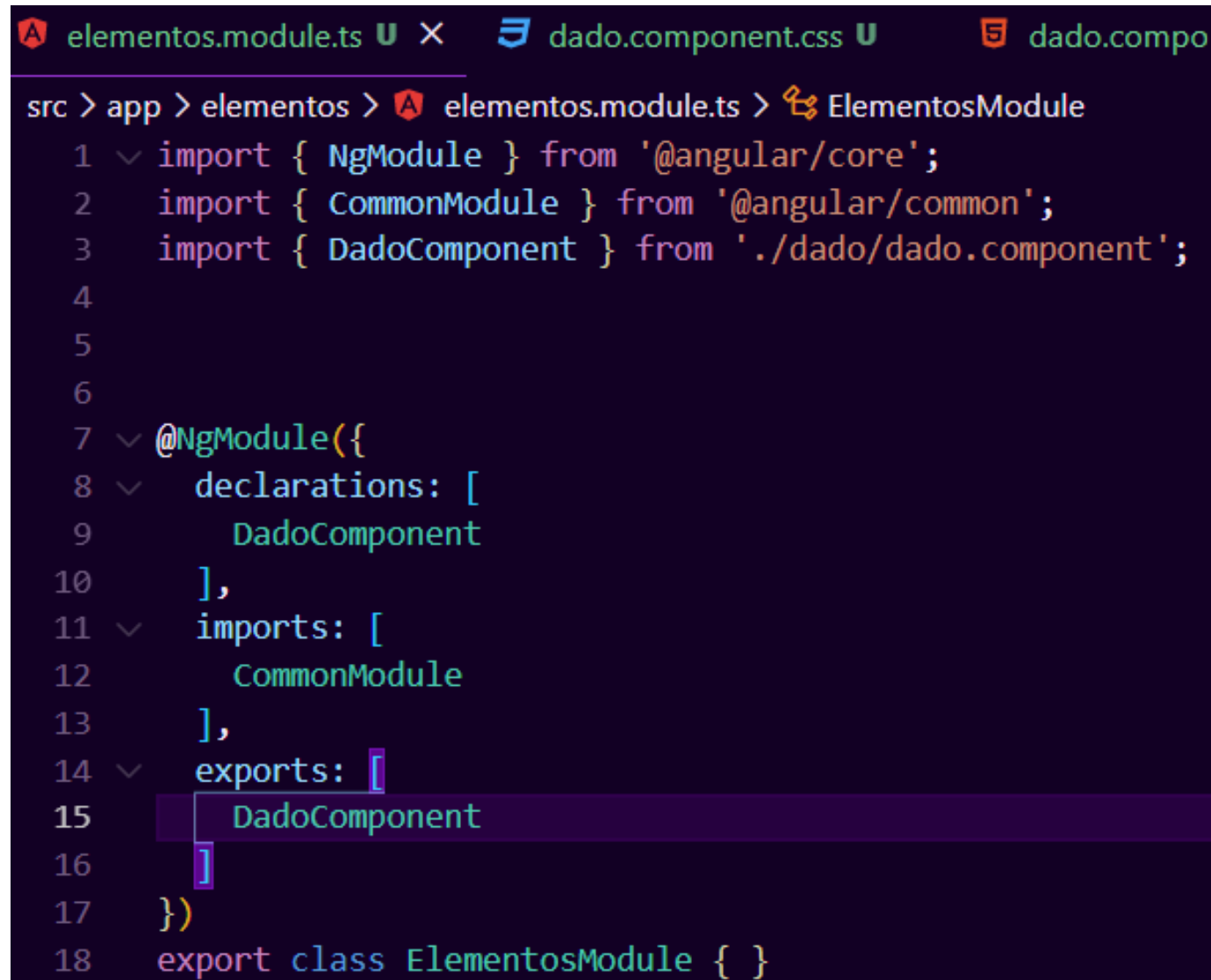
En el apartado de “dato.component.html” vamos a crear un div donde tendrá la clase ‘forma’ y dentro de este solo mandaremos a traer el valor que tenemos en “dato.component.ts”.

```
dato.component.html x elementos.module.ts U dato.c
src > app > elementos > dato > dato.component.html > div.forma
1 <div class="forma">
2   {{valor}}
3 </div>
```

Ahora crearemos los estilos en “dato.component.css” los cuales son ‘forma’.

```
dato.component.css x dato.component.html U e
src > app > elementos > dato > dato.component.css > .forma
1 .forma {
2   width: 5rem;
3   height: 5rem;
4   font-size: 3rem;
5   color: white;
6   background-color: black;
7   border-radius: 1rem;
8   display: inline-flex;
9   justify-content: center;
10  align-items: center;
11  margin: 10px;
12 }
```

El siguiente paso fundamental para que otros módulos pueden consumir la componente 'dado' es modificar el archivo 'elementos.module.ts' exportando la componente que acabamos de crear:



```
src > app > elementos > elementos.module.ts > ElementosModule
1  import { NgModule } from '@angular/core';
2  import { CommonModule } from '@angular/common';
3  import { DadoComponent } from './dado/dado.component';
4
5
6
7  @NgModule({
8    declarations: [
9      DadoComponent
10   ],
11   imports: [
12     CommonModule
13   ],
14   exports: [
15     DadoComponent
16   ]
17 })
18 export class ElementosModule { }
```


Ahora abrimos y modificamos el módulo principal que la va a importar para poder consumirla:

```
src > app > app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6
7  import { ElementosModule } from './elementos/elementos.module';
8
9  @NgModule({
10   declarations: [
11     AppComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule,
16     ElementosModule
17   ],
18   providers: [],
19   bootstrap: [AppComponent]
20 })
21 export class AppModule { }
22
```

Finamente en el archivo 'app.component.html' podemos utilizar la clase 'dado' que se encuentra en otro módulo:

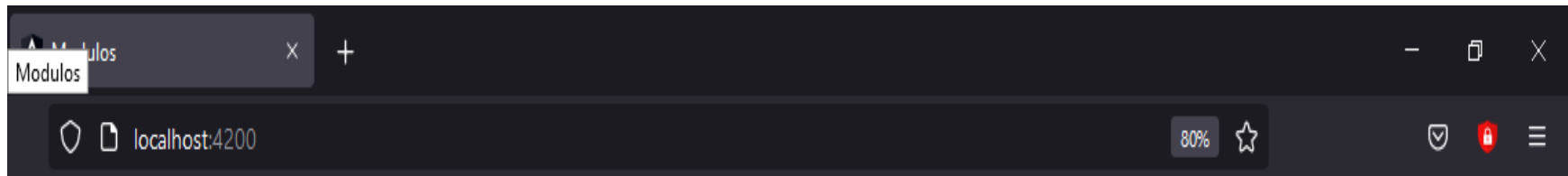
```
app.component.html M X
src > app > app.component.html > div > app-dado
1 | <div style="text-align:center">
2 |   <app-dado valor = "6" >/app-dado>
3 | </div>
```

Toda esta complejidad de agrupar componentes en distintos módulos tiene sentido si la aplicación es compleja.

Si ejecutamos ahora el proyecto:

```
PS C:\Users\Admin\Documents\AngularProyectos\modulos> ng serve -o
Generating browser application bundles (phase: setup)...
```

Podemos ver que tenemos:



6

2.- Practica 2 Petición de un archivo JSON a un servidor

Como primer paso nos vamos a dirigir a nuestra consola de comandos y poner la ruta donde se albergará el proyecto de angular.

```
C:\> Seleccionar Administrador: Símbolo del sistema  
Microsoft Windows [Versión 10.0.19041.985]  
(c) Microsoft Corporation. Todos los derechos reservados.  
C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos
```

Una vez que ya tengamos definida la ruta de la carpeta introduciremos la siguiente línea de comando para poder generar el proyecto de angular “ng new archivojson”. Y nos pregunta “¿Le gustaría agregar enrutamiento angular?” y solo pondremos la letra “y”.

```
C:\> npm  
Microsoft Windows [Versión 10.0.19041.985]  
(c) Microsoft Corporation. Todos los derechos reservados.  
C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos  
C:\Users\Admin\Documents\AngularProyectos>ng new archivojson  
? Would you like to add Angular routing? (y/N)
```

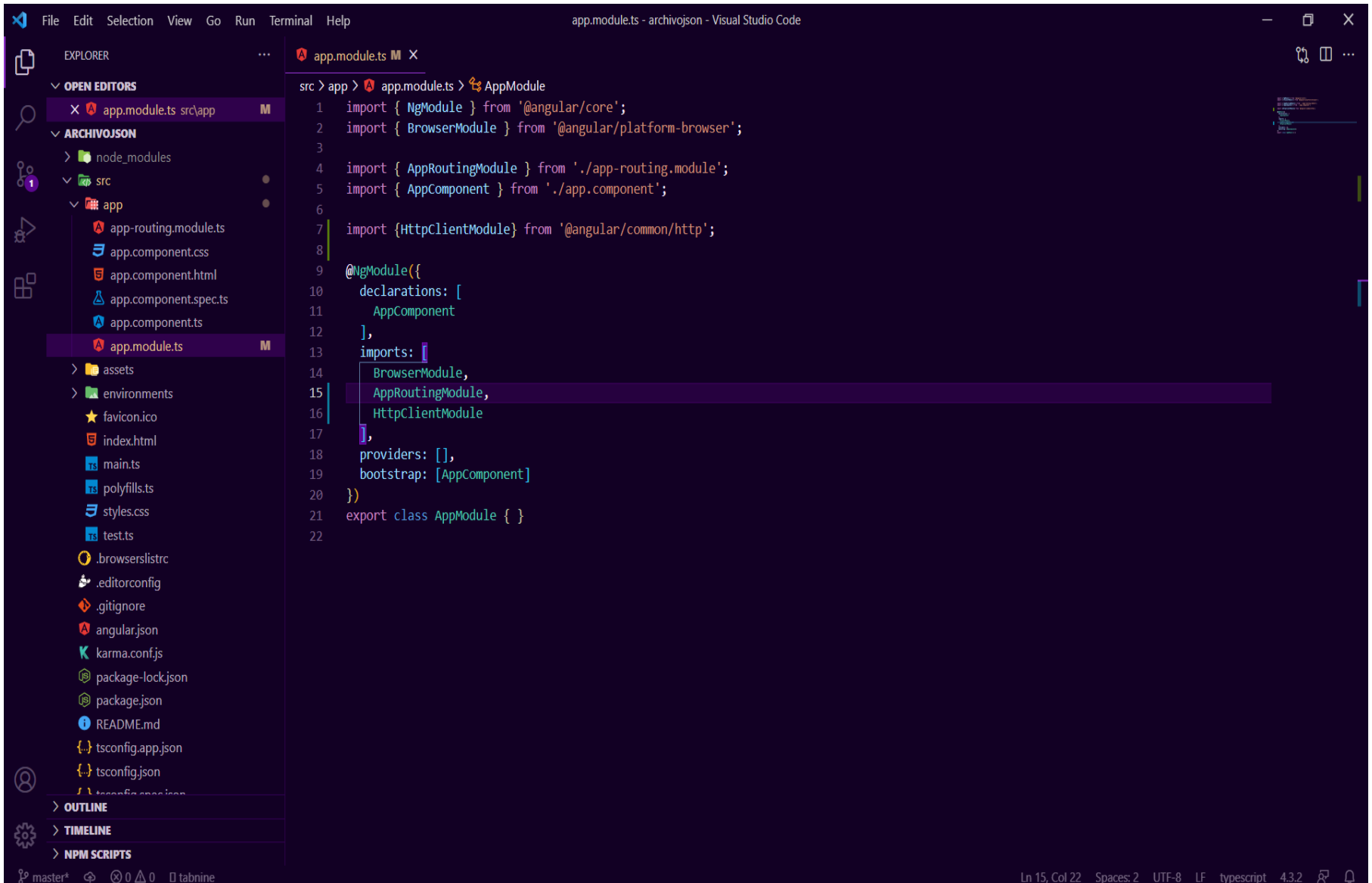
Después nos va a preguntar con qué tipos de estilos trabajaremos y escogeremos “css”. Y empezara a crear el proyecto.

```
C:\Users\Admin\Documents\AngularProyectos>ng new archivojson  
? Would you like to add Angular routing? Yes  
? Which stylesheet format would you like to use? (Use arrow keys)  
> CSS  
SCSS [ https://sass-lang.com/documentation/syntax#scss ]  
Sass [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]  
Less [ http://lesscss.org ]
```

```
C:\Users\Admin\Documents\AngularProyectos>ng new archivovjson
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? CSS
CREATE archivovjson/angular.json (3069 bytes)
CREATE archivovjson/package.json (1073 bytes)
CREATE archivovjson/README.md (1057 bytes)
CREATE archivovjson/tsconfig.json (783 bytes)
CREATE archivovjson/.editorconfig (274 bytes)
CREATE archivovjson/.gitignore (604 bytes)
CREATE archivovjson/.browserslistrc (703 bytes)
CREATE archivovjson/karma.conf.js (1428 bytes)
CREATE archivovjson/tsconfig.app.json (287 bytes)
CREATE archivovjson/tsconfig.spec.json (333 bytes)
CREATE archivovjson/src/favicon.ico (948 bytes)
CREATE archivovjson/src/index.html (297 bytes)
CREATE archivovjson/src/main.ts (372 bytes)
CREATE archivovjson/src/polyfills.ts (2820 bytes)
CREATE archivovjson/src/styles.css (80 bytes)
CREATE archivovjson/src/test.ts (743 bytes)
CREATE archivovjson/src/assets/.gitkeep (0 bytes)
CREATE archivovjson/src/environments/environment.prod.ts (51 bytes)
CREATE archivovjson/src/environments/environment.ts (658 bytes)
CREATE archivovjson/src/app/app-routing.module.ts (245 bytes)
CREATE archivovjson/src/app/app.module.ts (393 bytes)
CREATE archivovjson/src/app/app.component.html (23809 bytes)
CREATE archivovjson/src/app/app.component.spec.ts (1072 bytes)
CREATE archivovjson/src/app/app.component.ts (215 bytes)
CREATE archivovjson/src/app/app.component.css (0 bytes)
```

Abriremos nuestro editor de código en mi casi Visual Studio Code, y procederemos a abrir el proyecto “modulos” que creamos, el cual dentro del mismo existe la carpeta del módulo llamada “archivojson”.

Como primer paso importaremos el módulo HttpClientModule en el archivo app.module.ts:



```
File Edit Selection View Go Run Terminal Help
app.module.ts - archivojson - Visual Studio Code

EXPLORER
OPEN EDITORS
  app.module.ts src/app
ARCHIVOJSON
  node_modules
  src
  app
    app-routing.module.ts
    app.component.css
    app.component.html
    app.component.spec.ts
    app.component.ts
    app.module.ts
  assets
  environments
  favicon.ico
  index.html
  main.ts
  polyfills.ts
  styles.css
  test.ts
  .browserslistrc
  .editorconfig
  .gitignore
  angular.json
  karma.conf.js
  package-lock.json
  package.json
  README.md
  tsconfig.app.json
  tsconfig.json
  tsconfig.spec.json
OUTLINE
TIMELINE
NPM SCRIPTS

src > app > app.module.ts > AppModule
1 import { NgModule } from '@angular/core';
2 import { BrowserModule } from '@angular/platform-browser';
3
4 import { AppRoutingModule } from './app-routing.module';
5 import { AppComponent } from './app.component';
6
7 import { HttpClientModule } from '@angular/common/http';
8
9 @NgModule({
10   declarations: [
11     AppComponent
12   ],
13   imports: [
14     BrowserModule,
15     AppRoutingModule,
16     HttpClientModule
17   ],
18   providers: [],
19   bootstrap: [AppComponent]
20 })
21 export class AppModule { }
22
```

Implementaremos toda la lógica de lectura de datos en la componente por defecto que ha creado Angular CLI, luego en conceptos futuros veremos cómo distribuir las responsabilidades entre distintas clases.

```
app.module.ts M X app.component.ts M X
src > app > app.component.ts > AppComponent > ngOnInit > subscribe() callback
1  import { Component } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3
4  @Component({
5    selector: 'app-root',
6    templateUrl: './app.component.html',
7    styleUrls: ['./app.component.css']
8  })
9  export class AppComponent {
10
11    title = 'archivojson';
12    articulos:any = null;
13
14    constructor (private http: HttpClient) { }
15
16    ngOnInit() {
17      this.http.get("http://scratchya.com.ar/vue/datos.php")
18        .subscribe(
19        result => {
20          this.articulos = result;
21        },
22        error => {
23          console.log("Problemas");
24        }
25      );
26    }
27
28  }
```

Importamos la clase HttpClient:

```
> app > app.component.ts > AppComponent > ngOnInit > subscribe() callback
1  import { Component } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3
```

Definimos un atributo llamado artículos de tipo any con valor inicial null:

```
title = 'archivojson';  
articulos:any = null;
```

En Angular podemos definir una propiedad en los parámetros del constructor que se inyecta cuando se crea la componente:

```
constructor (private http: HttpClient) { }
```

Luego la propiedad http que es de la clase HttpClient nos servirá para hacer la petición al servidor.

En el método “ngOnInit” que se ejecuta una vez que el template de la componente está creado procedemos a recuperar del servidor los datos llamando al método get de la propiedad http:

```
ngOnInit() {  
    this.http.get("http://scratchya.com.ar/vue/datos.php")  
        .subscribe(  
            result => {  
                this.articulos = result;  
            },  
            error => {  
                console.log("Problemas");  
            }  
        );  
}
```

A partir del objeto que retorna el método get llamamos al método subscribe y le pasamos dos funciones. La primera de esas funciones recibe como parámetro los datos recuperados del servidor.

Falta que veamos como en la vista procedemos a mostrar los datos recuperados 'app.component.html':

```
app.module.ts M  app.component.ts M  app.component.html M
src > app > app.component.html > div.container.mt-3 > div.row.mt-3 >
1  <div class="container mt-3">
2    <div class="row mt-3">
3      <div *ngIf="articulos != null; else espera">
4        <table class="table table-striped" border="1">
5          <tr>
6            <th>Codigo</th>
7            <th>Descripcion</th>
8            <th>Precio</th>
9          </tr>
10         <tbody>
11           <tr *ngFor="let art of articulos">
12             <td>{{art.codigo}}</td>
13             <td>{{art.descripcion}}</td>
14             <td>{{art.precio}}</td>
15           </tr>
16         </tbody>
17       </table>
18     </div>
19   </div>
20 </div>
21 <ng-template #espera>Esperando datos...</ng-template>
```

Como las peticiones JSON a un servidor pueden demorarse un tiempo mediante la directiva *ngIf verificamos si la variable articulos tiene un null procedemos a mostrar el contenido de la etiqueta 'ng-template':

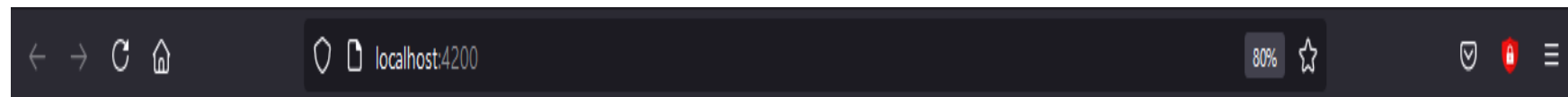
```
<div *ngIf="articulos != null; else espera">
  <table class="table table-striped" border="1">
```


Para mostrar los datos de la propiedad 'articulos' lo hacemos como lo hemos visto anteriormente:

```
<table class= "table table-striped" border="1">
  <tr>
    <th>Codigo</th>
    <th>Descripcion</th>
    <th>Precio</th>
  </tr>
  <tbody>
    <tr *ngFor="let art of articulos">
      <td>{{art.codigo}}</td>
      <td>{{art.descripcion}}</td>
      <td>{{art.precio}}</td>
    </tr>
  </tbody>
</table>
```

Si ejecutamos ahora el proyecto:

```
PS C:\Users\Admin\Documents\AngularProyectos\archivojson> ng serve -o
```



Codigo	Descripcion	Precio
1	papas	34
2	manzanas	23.5
3	sandia	31

3.- Practica 3 Router: definición de rutas

Como primer paso nos vamos a dirigir a nuestra consola de comandos y poner la ruta donde se albergará el proyecto de angular.

```
Selecciónar Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.985]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos
```

Una vez que ya tengamos definida la ruta de la carpeta introduciremos la siguiente línea de comando para poder generar el proyecto de angular “ng new defRutas --routing”. Y nos pregunta “¿Le gustaría agregar enrutamiento angular?” y solo pondremos la letra “y”.

```
C:\Users\Admin\Documents\AngularProyectos>ng new defRutas --routing
? Which stylesheet format would you like to use? (Use arrow keys)
```

Después nos va a preguntar con qué tipos de estilos trabajaremos y escogeremos “css”. Y empezara a crear el proyecto.

```
Microsoft Windows [Versión 10.0.19041.985]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos

C:\Users\Admin\Documents\AngularProyectos>ng new defRutas --routing
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS  [ https://sass-lang.com/documentation/syntax#scss ]
  Sass  [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less  [ http://lesscss.org ]
```

```
C:\Users\Admin\Documents\AngularProyectos>ng new defRutas --routing
? Which stylesheet format would you like to use? CSS
CREATE defRutas/angular.json (3051 bytes)
CREATE defRutas/package.json (1071 bytes)
CREATE defRutas/README.md (1054 bytes)
CREATE defRutas/tsconfig.json (783 bytes)
CREATE defRutas/.editorconfig (274 bytes)
CREATE defRutas/.gitignore (604 bytes)
CREATE defRutas/.browserslistrc (703 bytes)
CREATE defRutas/karma.conf.js (1425 bytes)
CREATE defRutas/tsconfig.app.json (287 bytes)
CREATE defRutas/tsconfig.spec.json (333 bytes)
CREATE defRutas/src/favicon.ico (948 bytes)
CREATE defRutas/src/index.html (294 bytes)
CREATE defRutas/src/main.ts (372 bytes)
CREATE defRutas/src/polyfills.ts (2820 bytes)
CREATE defRutas/src/styles.css (80 bytes)
CREATE defRutas/src/test.ts (743 bytes)
CREATE defRutas/src/assets/.gitkeep (0 bytes)
CREATE defRutas/src/environments/environment.prod.ts (51 bytes)
CREATE defRutas/src/environments/environment.ts (658 bytes)
CREATE defRutas/src/app/app-routing.module.ts (245 bytes)
CREATE defRutas/src/app/app.module.ts (393 bytes)
CREATE defRutas/src/app/app.component.html (23809 bytes)
CREATE defRutas/src/app/app.component.spec.ts (1063 bytes)
CREATE defRutas/src/app/app.component.ts (212 bytes)
CREATE defRutas/src/app/app.component.css (0 bytes)
/ Installing packages (npm)...
```

En la carpeta app se crea el archivo 'app-routing.module.ts'. Debemos modificar y cargar las dos rutas de nuestra aplicación e importar las componentes que visualizan dichas rutas:

```
src > app > app-routing.module.ts > AppRoutingModuleModule
1  import { NgModule } from '@angular/core';
2  import { RouterModule, Routes } from '@angular/router';
3
4  import { JuegodadosComponent } from './juegodados/juegodados.component';
5  import { AcercadeComponent } from './acercade/acercade.component';
6
7  const routes: Routes = [
8    {
9      path: 'juegodados',
10     component: JuegodadosComponent
11   },
12   {
13     path: 'acercade',
14     component: AcercadeComponent
15   }
16 ];
17
18 @NgModule({
19   imports: [RouterModule.forRoot(routes)],
20   exports: [RouterModule]
21 })
22 export class AppRoutingModule { }
23
```

Por el momento debe mostrar un error cuando iniciamos los nombres de las componentes 'JuegodadosComponent' y 'AcercadeComponent' debido que no las hemos codificado aún.

En la propiedad path estamos indicando que si en el navegador disponemos:

<http://localhost:4200/juegodados>

Desde la línea de comandos de Node.js procedemos a crear cada una de las tres componentes que faltan en la aplicación:

```
PS C:\Users\Admin\Documents\AngularProyectos\defRutas> ng generate component juegodados
CREATE src/app/juegodados/juegodados.component.html (25 bytes)
CREATE src/app/juegodados/juegodados.component.spec.ts (654 bytes)
CREATE src/app/juegodados/juegodados.component.ts (291 bytes)
CREATE src/app/juegodados/juegodados.component.css (0 bytes)
UPDATE src/app/app.module.ts (491 bytes)
```

Codificamos dos archivos, juegodados.component.ts

```
app-routing.module.ts 1, M X  juegodados.component.ts U X
src > app > juegodados > juegodados.component.ts > JuegodadosComponent > retornarAleatorio
1  import { Component, OnInit } from '@angular/core';
2
3  @Component({
4    selector: 'app-juegodados',
5    templateUrl: './juegodados.component.html',
6    styleUrls: ['./juegodados.component.css']
7  })
8  export class JuegodadosComponent implements OnInit {
9    valor1: number = 0;
10   valor2: number = 0;
11   valor3: number = 0;
12   resultado: String = '';
13   constructor() {
14     this.valor1 = this.retornarAleatorio();
15     this.valor2 = this.retornarAleatorio();
16     this.valor3 = this.retornarAleatorio();
17   }
18
19   retornarAleatorio() {
20     return Math.trunc(Math.random() * 6) + 1;
21   }
22
23   tirar() {
24     this.valor1 = this.retornarAleatorio();
25     this.valor2 = this.retornarAleatorio();
26     this.valor3 = this.retornarAleatorio();
27     if (this.valor1 == this.valor2 && this.valor3 == this.valor1) {
28       this.resultado = 'Gano!!!';
29     } else {
30       this.resultado = 'Perdio!!!';
31     }
32   }
33
34   ngOnInit(): void {
35   }
36
37 }
38
```

juegodados.component.html

```
src > app > juegodados > juegodados.component.html > p
1 <app-dado valor="{{valor1}}"></app-dado>
2 <app-dado valor="{{valor2}}"></app-dado>
3 <app-dado valor="{{valor3}}"></app-dado>
4 <hr>
5 <button (click)="tirar()">Tirar</button>
6 <hr>
7 <p>Resultado:{{resultado}}</p>
```

Generamos ahora la componente 'dado':

```
PS C:\Users\Admin\Documents\AngularProyectos\defRutas> ng generate component dado
CREATE src/app/dado/dado.component.html (19 bytes)
CREATE src/app/dado/dado.component.spec.ts (612 bytes)
CREATE src/app/dado/dado.component.ts (267 bytes)
CREATE src/app/dado/dado.component.css (0 bytes)
UPDATE src/app/app.module.ts (565 bytes)
```

Codificamos sus tres archivos, dado.component.ts

```
src > app > dado > dado.component.ts > ...
1 import { Component, OnInit, Input } from '@angular/core';
2
3 @Component({
4   selector: 'app-dado',
5   templateUrl: './dado.component.html',
6   styleUrls: ['./dado.component.css']
7 })
8 export class DadoComponent implements OnInit {
9
10   @Input() valor:number = 0
11
12   constructor() { }
13
14   ngOnInit(): void {
15   }
16
17 }
```

dado.component.html

```
src > app > dado > dado.component.html > div.forma
1 <div class="forma">
2   {{valor}}
3 </div>
```

dado.component.css

```
src > app > dado > dado.component.css > .forma
1 .forma {
2   width: 5rem;
3   height: 5rem;
4   font-size: 3rem;
5   color: white;
6   background-color: black;
7   border-radius: 1rem;
8   display: inline-flex;
9   justify-content: center;
10  align-items: center;
11  margin: 10px;
12 }
```

Finalmente creamos la última componente: ng generate component acercade

```
PS C:\Users\Admin\Documents\AngularProyectos\defRutas> ng generate component acercade
CREATE src/app/acercade/acercade.component.html (23 bytes)
CREATE src/app/acercade/acercade.component.spec.ts (640 bytes)
CREATE src/app/acercade/acercade.component.ts (283 bytes)
CREATE src/app/acercade/acercade.component.css (0 bytes)
UPDATE src/app/app.module.ts (655 bytes)
```

Modificamos el archivo 'acercade.component.html':

```
src > app > acercade > acercade.component.html
```

```
<h1>Programa: xxxxxxxxxxx</h1>  
<p>Desarrollado: xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx</p>  
<p>Fecha: xxxxxx</p>
```

Si abrimos ahora el archivo `app.module.ts` tenemos declarados las 4 componentes e importado el módulo `AppRoutingModule`:

```
src > app > app.module.ts > ...
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppRoutingModuleModule } from './app-routing.module';
5
6  import { AppComponent } from './app.component';
7  import { JuegodadosComponent } from './juegodados/juegodados.component';
8  import { DadoComponent } from './dado/dado.component';
9  import { AcercadeComponent } from './acercade/acercade.component';
10
11  @NgModule({
12    declarations: [
13      AppComponent,
14      JuegodadosComponent,
15      DadoComponent,
16      AcercadeComponent
17    ],
18    imports: [
19      BrowserModule,
20      AppRoutingModuleModule
21    ],
22    providers: [],
23    bootstrap: [AppComponent]
24  })
25  export class AppModule { }
```


Si abrimos ahora el archivo `app.module.ts` tenemos declarados las 4 componentes e importado el módulo `AppRoutingModule`:

```
src > app > app.component.html > div.container.mt-3 > div.row.mt-3 > div > div > router-outlet
1  <div class="container mt-3">
2    <div class="row mt-3">
3      <div style="text-align:center">
4        <a routerLink="/juegodados">Juego de dados</a> -
5        <a routerLink="/acercade">Acerca de ...</a>
6      <div>
7        <router-outlet></router-outlet>
8      </div>
9    </div>
10  </div>
11 </div>
```

Mediante la etiqueta '`router-outlet`' indicamos el lugar que debe mostrar la componente especificada por la ruta configurada en el archivo '`app-routing.module.ts`'

Para cambiar de ruta mediante hipervínculos debemos iniciar la propiedad '`routerLink`' asignando la ruta respectiva.

4.- Practica 4 Servicios: concepto y pasos para crearlos

Como primer paso nos vamos a dirigir a nuestra consola de comandos y poner la ruta donde se albergará el proyecto de angular.

```
Seleccionar Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.985]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos
```

Una vez que ya tengamos definida la ruta de la carpeta introduciremos la siguiente línea de comando para poder generar el proyecto de angular “ng new servicios”. Y nos pregunta “¿Le gustaría agregar enrutamiento angular?” y solo pondremos la letra “y”.

```
C:\Users\Admin\Documents\AngularProyectos>ng new servicios
? Would you like to add Angular routing? (y/N) y
```

Después nos va a preguntar con qué tipos de estilos trabajaremos y escogeremos “css”. Y empezara a crear el proyecto.

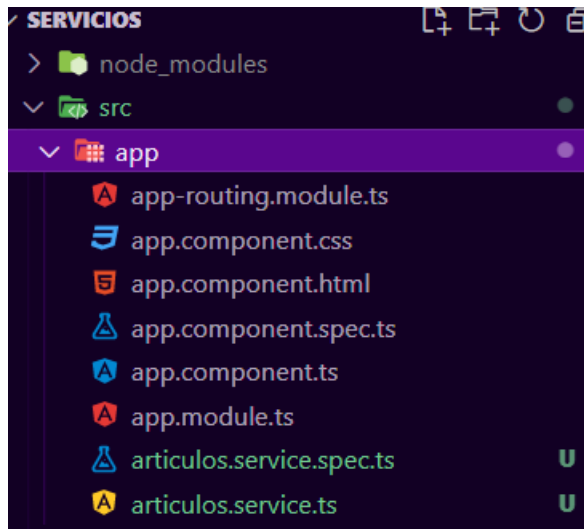
```
C:\Users\Admin\Documents\AngularProyectos>ng new servicios
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
SCSS  [ https://sass-lang.com/documentation/syntax#scss ]
Sass  [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
Less  [ http://lesscss.org ]
```

Crearemos el servicio que contiene en memoria la lista de artículos (en muchos casos como veremos más adelante el servicio tiene la responsabilidad de recuperar los datos de un servidor web): ng generate service artículos

```
PS C:\Users\Admin\Documents\AngularProyectos\servicios> ng generate service artículos
CREATE src/app/articulos.service.spec.ts (372 bytes)
CREATE src/app/articulos.service.ts (138 bytes)
```

Con el comando anterior estamos creando la clase 'ArticulosService'.

Se crean dos archivos:



El código generado de la clase 'ArticulosService' es:

```
articulos.service.ts U X
src > app > articulos.service.ts > ...
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class ArticulosService {
7
8    constructor() { }
9  }
```

Lo modificamos por el siguiente código que permita recuperar desde la componente el vector de artículos:

```
articulos.service.ts x
src > app > A articulos.service.ts > ArticulosService > retornar
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class ArticulosService {
7
8    constructor() { }
9
10   retornar() {
11     return [
12       {
13         codigo: 1,
14         descripcion: 'papas',
15         precio: 12.33
16       },
17       {
18         codigo: 2,
19         descripcion: 'manzanas',
20         precio: 54
21       },
22       {
23         codigo: 3,
24         descripcion: 'sandia',
25         precio: 14
26       }
27     ];
28   }
29 }
```

El decorador @Injectable() será de suma importancia para poder acceder a esta clase desde la componente.

El archivo 'app.module.ts' no se modifica.

Ahora veremos como consumimos el servicio desde nuestra componente. Procedemos a modificar la componente que se crea por defecto 'AppComponent' que tiene por responsabilidad mostrar en la página el listado de artículos:

```
src > app > app.component.ts > AppComponent > ngOnInit
1 | import { Component, OnInit } from '@angular/core';
2 | import { ArticulosService } from './articulos.service';
3 |
4 | @Component({
5 |   selector: 'app-root',
6 |   templateUrl: './app.component.html',
7 |   styleUrls: ['./app.component.css']
8 | })
9 | export class AppComponent {
10 |   articulos: any = null;
11 |
12 |   constructor(private articulosServicio: ArticulosService) {
13 |   }
14 |
15 |   ngOnInit() {
16 |     this.articulos = this.articulosServicio.retornar();
17 |   }
18 | }
19 |
```

Primero importamos el servicio llamado ArticulosService que se almacena en el archivo 'articulos.service.ts':

```
import { ArticulosService } from './articulos.service';
```

Para inyectar el objeto de la clase 'ArticulosService' que crea Angular en forma automática lo hacemos en el parámetro del constructor:

```
constructor(private articulosServicio: ArticulosService) {
}
```

Se almacena en el atributo 'articulosServicio' la referencia del objeto de la clase 'ArticulosService' que crea Angular.

En el método ngOnInit actualizamos la variable 'articulos' con el vector que devuelve el método 'retornar':

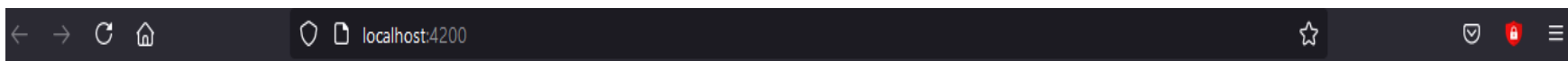
```
ngOnInit() {  
  this.articulos = this.articulosServicio.retornar();  
}
```

Esta asignación dispara la actualización de la página HTML.

Falta que codifiquemos la vista con los datos recuperados: app.component.html

```
src > app > app.component.html > div.container.mt-4  
1  <div class="container mt-4">  
2    <div class="row mt-4">  
3      <table class="table table-striped" border="1">  
4        <tr *ngFor="let articulo of articulos">  
5          <td>{{articulo.codigo}}</td>  
6          <td>{{articulo.descripcion}}</td>  
7          <td>{{articulo.precio}}</td>  
8        </tr>  
9      </table>  
10    </div>  
11  </div>
```

Si ejecutamos ahora el proyecto012 veremos en el navegador el listado de artículos: ng server -o



1	papas	12.33
2	manzanas	54
3	sandia	14

5.- Practica 5 Servicios: recuperación de datos de un servidor web

Como primer paso nos vamos a dirigir a nuestra consola de comandos y poner la ruta donde se albergará el proyecto de angular.

```
Selecc... Administrador: Símbolo del sistema
Microsoft Windows [Versión 10.0.19041.985]
(c) Microsoft Corporation. Todos los derechos reservados.

C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos
```

Una vez que ya tengamos definida la ruta de la carpeta introduciremos la siguiente línea de comando para poder generar el proyecto de angular “ng new recuperacion”. Y nos pregunta “¿Le gustaría agregar enrutamiento angular?” y solo pondremos la letra “y”.

```
C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos

C:\Users\Admin\Documents\AngularProyectos>ng new recuperacion
? Would you like to add Angular routing? (y/N) _
```

Después nos va a preguntar con qué tipos de estilos trabajaremos y escogeremos “css”. Y empezara a crear el proyecto.

```
C:\WINDOWS\system32>cd C:\Users\Admin\Documents\AngularProyectos

C:\Users\Admin\Documents\AngularProyectos>ng new recuperacion
? Would you like to add Angular routing? Yes
? Which stylesheet format would you like to use? (Use arrow keys)
> CSS
  SCSS  [ https://sass-lang.com/documentation/syntax#scss ]
  Sass  [ https://sass-lang.com/documentation/syntax#the-indented-syntax ]
  Less  [ http://lesscss.org ]
```


Crearemos el servicio que recuperará desde un servidor la lista de artículos

```
C:\Users\Admin\Documents\AngularProyectos\recuperacion>ng generate service articulos
CREATE src/app/articulos.service.spec.ts (372 bytes)
CREATE src/app/articulos.service.ts (138 bytes)
```

Con el comando anterior estamos creando la clase 'ArticulosService'

Se crean dos archivos.

El código generado de la clase 'ArticulosService' es:

```
articulos.service.ts U X
src > app > A articulos.service.ts > ...
1  import { Injectable } from '@angular/core';
2
3  @Injectable({
4    providedIn: 'root'
5  })
6  export class ArticulosService {
7
8    constructor() { }
9  }
```

Lo modificamos por el siguiente código que permita recuperar desde un servidor web el archivo JSON:

```
articulos.service.ts U X
src > app > A articulos.service.ts > ArticulosService > retornar
1  import { Injectable } from '@angular/core';
2  import { HttpClient } from '@angular/common/http';
3
4  @Injectable({
5    providedIn: 'root'
6  })
7  export class ArticulosService {
8
9    constructor(private http: HttpClient) { }
10
11    retornar(){
12      return this.http.get("http://scratchya.com.ar/vue/datos.php");
13    }
14  }
```

El archivo 'app.module.ts' se modifica con el siguiente código (se importa la clase HttpClientModule):

```
app.module.ts M X
src > app > app.module.ts > AppModule
1  import { NgModule } from '@angular/core';
2  import { BrowserModule } from '@angular/platform-browser';
3
4  import { AppRoutingModule } from './app-routing.module';
5  import { AppComponent } from './app.component';
6
7  import { HttpClient } from '@angular/common/http';
8
9  @NgModule({
10     declarations: [
11         AppComponent
12     ],
13     imports: [
14         BrowserModule,
15         AppRoutingModule,
16         HttpClientModule
17     ],
18     providers: [],
19     bootstrap: [AppComponent]
20 })
21 export class AppModule { }
```

Ahora veremos cómo consumimos el servicio desde nuestra componente. Procedemos a modificar la componente que se crea por defecto 'AppComponent' que tiene por responsabilidad mostrar en la página el listado de artículos:

```
src > app > app.component.ts > AppComponent > articulos
1 | import { Component, OnInit } from '@angular/core';
2 | import { ArticulosService } from './articulos.service';
3
4 | @Component({
5 |   selector: 'app-root',
6 |   templateUrl: './app.component.html',
7 |   styleUrls: ['./app.component.css']
8 | })
9 | export class AppComponent {
10 |   articulos:any = null;
11
12 |   constructor(private articulosService:ArticulosService){}
13
14 |   ngOnInit(){
15 |     this.articulosService.retornar()
16 |     .subscribe( result => this.articulos = result)
17 |   }
18 | }
19
```

Primero importamos el servicio llamado ArticulosService que se almacena en el archivo 'articulos.service.ts':

```
2 | import { ArticulosService } from './articulos.service';
3
```

Para inyectar el objeto de la clase 'ArticulosService' que crea Angular en forma automática lo hacemos en el parámetro del constructor:

```
constructor(private articulosService:ArticulosService){}
```

Se almacena en el atributo 'articulosServicio' la referencia del objeto de la clase 'ArticulosService' que crea Angular.

En el método ngOnInit actualizamos la propiedad 'articulos' con el resultado devuelto:

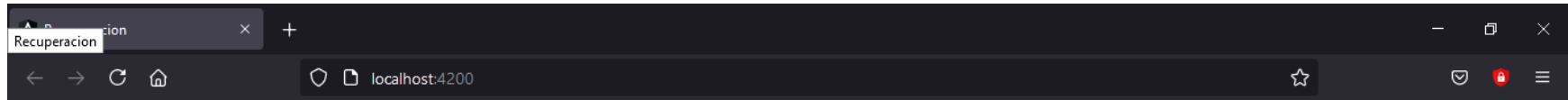
```
ngOnInit(){
  this.articulosService.retornar()
  .subscribe( result => this.articulos = result)
}
```

Esta asignación dispara la actualización de la página HTML.

Falta que codifiquemos la vista con los datos recuperados: app.component.html

```
src > app > app.component.html > div.container.mt-4 > div.row.mt-4 > div.col.mt-4 > div > table
1  <div class="container mt-4">
2    <div class="row mt-4">
3      <div class="col mt-4">
4        <div *ngIf="articulos != null; else espera">
5          <table class="table table-striped" border="1">
6            <tr>
7              <th>Codigo</th>
8              <th>Descripción</th>
9              <th>Precio</th>
10             </tr>
11            <tbody>
12              <tr *ngFor="let art of articulos">
13                <td>{{art.codigo}}</td>
14                <td>{{art.descripcion}}</td>
15                <td>{{art.precio}}</td>
16              </tr>
17            </tbody>
18          </table>
19        </div>
20      </div>
21    </div>
22  </div>
23  <div class="container mt-4">
24    <div class="row mt-4">
25      <div class="col mt-4 lead">
26        <ng-template #espera>Esperando datos...</ng-template>
27      </div>
28    </div>
29  </div>
30 </div>
31 </div>
```

Si ejecutamos ahora el proyecto “recuperacion” veremos en el navegador el listado de artículos, pero ahora recuperados de un servidor y no extraídos de un vector como en el concepto anterior: `ng server -o`



Codigo	Descripcion	Precio
1	papas	34
2	manzanas	23.5
3	sandia	31