esarrollado toda la lógica en la componente que se crea por defecto al crear un proyecto con Angular

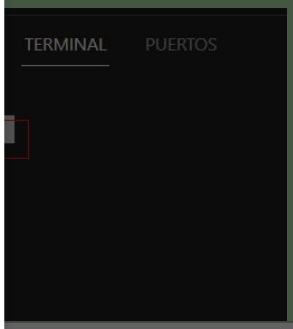
proyectos de mediano y gran tamaño no podemos disponer toda la lógica en una única componente.

una de las características fundamentales de Angular. Ayudan a extender las características básicas y encapsular código.

ee la posibilidad de crearlas desde la línea de comandos de Node.js

nponente 'dado' además de la componente que crea por defecto Angular CLI

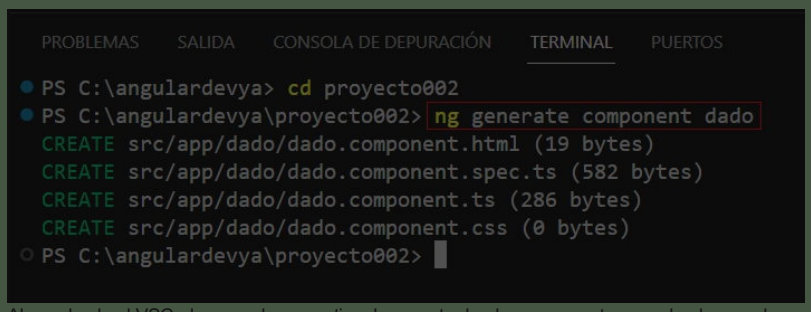
e procedemos a crear el segundo proyecto (cuidado de crearlo desde la carpeta c:\angujardevya\):



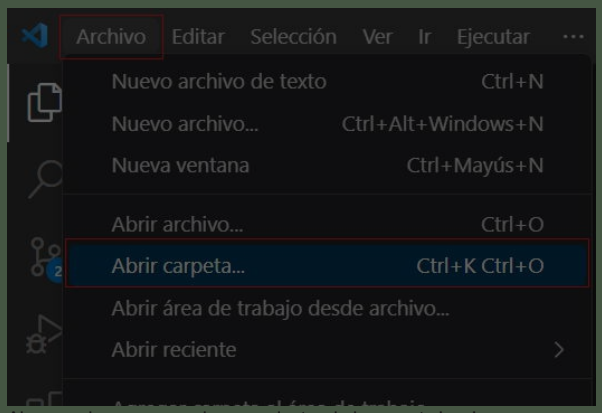

2. Primero descendemos a la carpeta proyecto002 y nuevamente desde la línea de comandos procedemos a crear la componente 'dado' escribiendo:

```
c:\angularya> cd proyecto002
c:\angularya\proyecto002> ng generate component dado
```

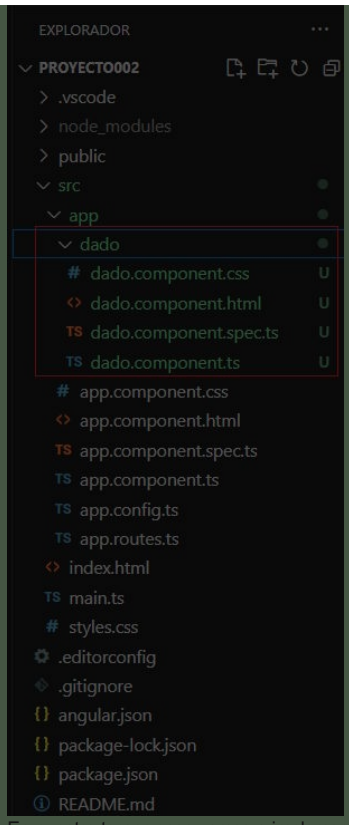
Al ejecutar este comando se crean 4 archivos:




Ahora desde el VSCode ya podemos activar la carpeta donde se encuentra creado el segundo proyecto:




Ahora podemos comprobar que dentro de la carpeta 'app' se crea una carpeta llamada 'dado' y dentro de ella se localizan los cuatro archivos creados:




3. En nuestro tercer paso vamos a implementar la vista de la componente 'dado' y su modelo. Abrimos el archivo 'dado.component.ts' y codificamos:

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

@Component({
  selector: 'app-dado',
  imports: [],
  templateUrl: './dado.component.html',
  styleUrl: './dado.component.css'
})
export class DadoComponent {
  valor: number = Math.floor(Math.random() * 6) + 1;
}
```

La propiedad valor se inicializa con un valor aleatorio comprendido entre 1 y 6.

4. Codificamos ahora el archivo 'dado.component.html':

```
<div class="forma">
  {{valor}}
</div>
```

Como podemos ver solo mostramos el valor almacenado en la propiedad 'valor' definido en el modelo.

5. Para definir la hoja de estilo del 'dado' abrimos el archivo 'dado.component.css' y codificamos:

```
.forma {
  width: 5rem;
  height: 5rem;
  font-size: 3rem;
  color:white;
  background-color: black;
  border-radius: 1rem;
  display: inline-flex;
  justify-content: center;
  align-items: center;
  margin:10px;
}
```

6. Finamente nos falta definir tres objetos de nuestra clase 'DadoComponent', si volvemos a ver el archivo 'dado.component.ts' podemos identificar en la llamada a @Component que tiene una propiedad llamada 'selector' con el valor 'app-dado':

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

Este es el selector que debemos utilizar para definir objetos de la clase DadoComponent en las vistas.

Abrimos ahora el archivo 'app.component.html' y remplazamos su contenido con la definición de tres dados:

```
<div style="text-align:center">
  <app-dado></app-dado>
  <app-dado></app-dado>
  <app-dado></app-dado>
</div>

<router-outlet />
```

Todavía podemos ver que nos muestra un error como que no existe la etiqueta 'app-dado', esto debido que la debemos importar a la clase en el archivo 'app.component.ts'.

Debemos importar la clase e indicar que la utilizaremos:

```
import { Component } from '@angular/core';
import { RouterOutlet } from '@angular/router';
import { DadoComponent } from './dado/dado.component';

@Component({
  selector: 'app-root',
  imports: [RouterOutlet, DadoComponent],
  templateUrl: './app.component.html',
  styleUrl: './app.component.css'
})
export class AppComponent {
  title = 'proyecto002';
}
```

Es decir que importamos la componente del dado:

```
import { DadoComponent } from './dado/dado.component';
```

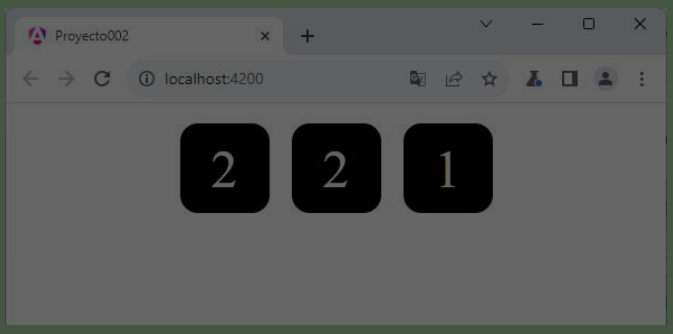
Y también indicamos el nombre de la clase en la propiedad 'imports':

```
imports: [RouterOutlet, DadoComponent]
```

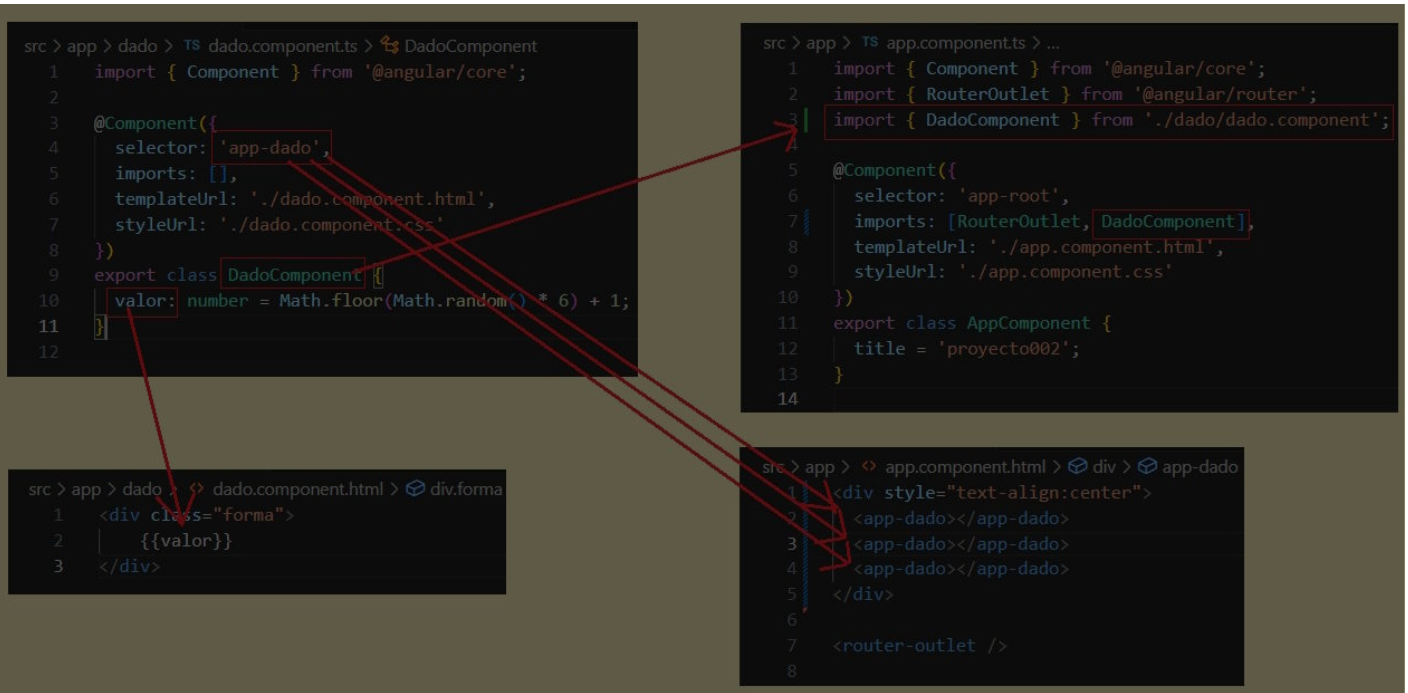
Si ejecutamos ahora el proyecto:

```
ng serve -o
```

Podemos ver que tenemos los tres dados en pantalla:




Es importante notar la relación de colaboración entre la clase 'DadoComponent' y la clase 'AppComponent':




La clase AppComponent se crea en forma automática cuando creamos un proyecto en Angular utilizando la herramienta 'Angular CLI'.

## Acotaciones

La división de un proyecto en componentes en Angular nos permite crear piezas independientes y reutilizables.

Siempre debe haber una primer componente donde arranca la aplicación, si utilizamos la herramienta Angular CLI se llama 'AppComponent'. Luego podemos crear otras componentes como en nuestro caso de 'DadoComponent'.

Tenemos que toda componente tiene un nombre de clase, por ejemplo 'DadoComponent' y luego un nombre de selector definida para dicha componente 'app-dado'. En las vistas para definir objetos de una determinada componente debemos hacer referencia al nombre del selector:

```
<app-dado></app-dado>
```

La componente 'AppComponent' si vemos su nombre de selector es 'app-root', luego si queremos ver donde se crea un objeto de este tipo de selector debemos abrir el archivo 'index.html' que lo generó automáticamente Angular CLI:

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Proyecto002</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Como podemos ver en el 'body' se define una etiqueta de tipo 'app-root':

```
<body>
  <app-root></app-root>
</body>
```

Gracias a esto siempre se crea un objeto de la clase AppComponent. Si borramos ésta etiqueta y corremos la aplicación luego tendremos una página web vacía.

# Retornar