

33 - Herramienta Angular CLI - comando: ng generate

Otro comando que hemos utilizado a lo largo de los primeros conceptos de este curso de Angular es 'generate'.

Mediante el comando 'generate' de la herramienta Angular CLI podemos crear:

- component
- service
- module
- pipe
- class
- interface
- enum
- directive
- application
- library
- guard
- service-worker
- web-worker
- resolver
- interceptor
- app-shell

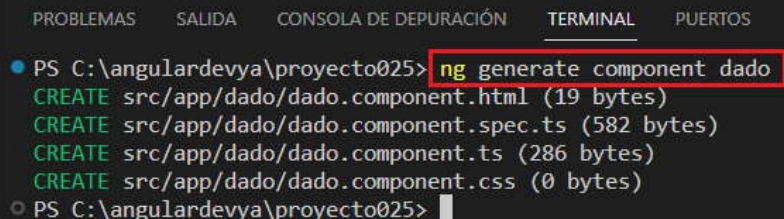
ng generate component

Mediante el comando:

```
ng generate component dado
```

Se crea una componente 'dado' con los archivos respectivos.

Se informa en la consola los archivos creados:



```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS
● PS C:\angulardevya\proyecto025> ng generate component dado
CREATE src/app/dado/dado.component.html (19 bytes)
CREATE src/app/dado/dado.component.spec.ts (582 bytes)
CREATE src/app/dado/dado.component.ts (286 bytes)
CREATE src/app/dado/dado.component.css (0 bytes)
○ PS C:\angulardevya\proyecto025>
```

Podemos pasar varias opciones cuando creamos la componente:

```
ng generate component dado --inline-style
```

Evita que se cree el archivo dado.component.css

Lo mismo podemos evitar que se cree el archivo HTML:

```
ng generate component dado --inline-template
```

Tenemos como resultado la no creación del archivo HTML:

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

● PS C:\angulardevya\proyecto025> ng generate component dado --inline-template
CREATE src/app/dado/dado.component.spec.ts (582 bytes)
CREATE src/app/dado/dado.component.ts (300 bytes)
CREATE src/app/dado/dado.component.css (0 bytes)
○ PS C:\angulardevya\proyecto025>
```

Podemos definir el prefijo para la componente mediante la opción --prefix (alias: -p):

```
ng generate component dado --prefix juego
```

Luego la componente que se crea tiene dicho prefijo:

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

@Component({
  selector: 'juego-dado',
  imports: [],
  templateUrl: './dado.component.html',
  styleUrls: ['./dado.component.css']
})
export class DadoComponent {

}
```

Si queremos definir el nombre completo para el selector tenemos la opción --selector:

```
ng generate component dado --selector ju-dado
```

Luego la componente se crea con dicho nombre de selector:

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

@Component({
  selector: 'ju-dado',
  imports: [],
  templateUrl: './dado.component.html',
  styleUrls: ['./dado.component.css']
})
export class DadoComponent {

}
```

Para evitar que se cree el archivo 'dado.spec.js' debemos insertar el comando --skip-tests:

```
ng generate component dado --skip-tests
```

Luego no se genera el archivo 'dado.spec.js'.

Tengamos en cuenta que todas estas opciones se pueden combinar y ejecutar en forma simultanea, por ejemplo si queremos generar solo el archivo *.ts de la componente y que no genere el archivo spec, *.css y *.html:

```
ng generate component dado --skip-tests --inline-template --inline-style
```

```
PROBLEMAS  SALIDA  CONSOLA DE DEPURACIÓN  TERMINAL  PUERTOS

PS C:\angulardevya\proyecto025> ng generate component dado --skip-tests --inline-template --inline-style
CREATE src/app/dado/dado.component.ts (278 bytes)
○ PS C:\angulardevya\proyecto025>
```

Otras opciones posibles cuando creamos una componente son:

- --force (alias: -f) Forzar la sobrescritura de los archivos existentes (se borra la componente anterior que tiene el mismo nombre)
- --dry-run (alias: -d) Informa los archivos que se crearán sin hacer dicha actividad.

Angular CLI permite ingresar comandos en formato resumido utilizando el primer caracter:

```
ng g c dado
```

En lugar de escribir:

```
ng generate component dado
```

ng generate service

Mediante el comando:

```
ng generate service articulos
```

Se crea una clase `ArticulosService` y se inyecta a nivel de 'root':

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

@Injectable({
  providedIn: 'root'
})
export class ArticulosService {

  constructor() { }
}
```

Disponemos de los siguientes opciones en este comando:

- --dry-run
- --force
- --skip-tests

ng generate pipe

Mediante el comando:

```
ng generate pipe letras
```

Se genera el archivo 'letras.pipe.ts':

```
import { Pipe, PipeTransform } from '@angular/core';

@Pipe({
  name: 'letras'
})
export class LetrasPipe implements PipeTransform {

  transform(value: unknown, ...args: unknown[]): unknown {
    return null;
  }
}
```

Disponemos de los siguientes opciones en este comando:

- --dry-run

- --force
- --skip-tests

ng generate class

Mediante el comando:

```
ng generate class articulo
```

Se crea el archivo 'articulo.ts':

```
export class Articulo {  
}
```

ng generate interface

Mediante el comando:

```
ng generate interface venta
```

Se crea el archivo 'venta.ts':

```
export interface Venta {  
}
```

ng generate enum

Mediante el comando:

```
ng generate enum operaciones
```

Se crea el archivo 'operaciones.enum.ts':

```
export enum Operaciones {  
}
```

Retornar