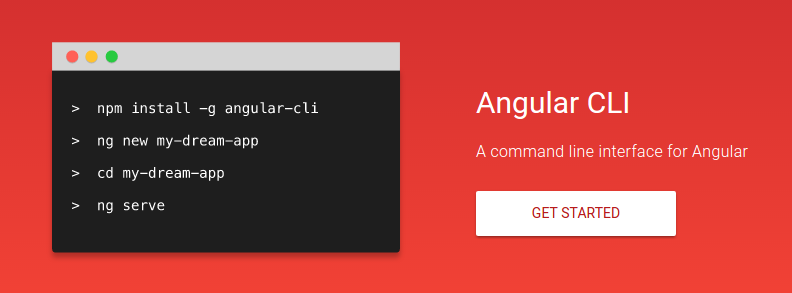
New CREAMOS UN PROYECTO ANGULAR CON ANGULAR CLI

[Angular CLI](https://cli.angular.io/) nos va a hacer la vida más fácil con **Angular**, ya que nos permite crear de forma sencilla un aplicación lista para funcionar después de la instalación por línea de comandos con **npm**, además incorpora las **buenas prácticas** de programación con **Angular**, como en el apartado anterior ya hemos explicado los diferentes ficheros que necesitamos, aquí, nos centraremos en la estructura que creamos con **Angular CLI**, y veremos lo sencillo que será crear un nuevo componente para nuestra aplicación. [Angular CLI](https://cli.angular.io/) esta desarrollado por el mismo equipo que desarrolla **Angular**.

[](https://cli.angular.io/)

Así que , si te has asustado en el apartado anterior, tranquilo, aquí te recuperarás.

En la web de desarrollo de **Angular CLI** tienes toda la documentación: [Angular-CLI en GitHub](https://github.com/angular/angular-cli), aquí iremos comentando lo que nos vaya haciendo falta. **Angular CLI** está en una versión beta, por lo que tenemos que estar atentos a las actualizaciones que se vayan realizando.

Instalamos Angular CLI

Para su instalación tenemos que tener en cuenta los requisitos que os comentaba al principio **Node.js** con versión al menos **5.x.x** y npm con versión al menos **3.x.x** , es importante ya que sino **Angular CLI** no funcionará. Una vez revisado procedemos a instalar **Angular CLI** desde la línea de comandos ejecutando:

|  |  |
| --- | --- |
| 1 | npm install -g angular-cli |

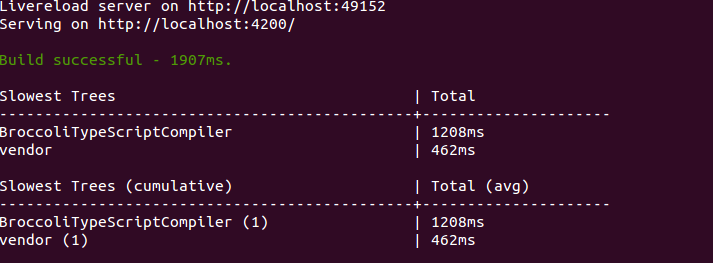
Listo, ya no necesitamos hacer nada más, vamos a crear el proyecto.

Creamos nuestro proyecto con Angular CLI

Una vez instalado **Angular CLI** usaremos **ng** para usar la aplicación, teclea **ng –help** para ver las opciones en líneas disponibles, para crear un nuevo proyecto utilizaremos simplemente **ng new** más el nombre del proyecto a continuación, tal que así:

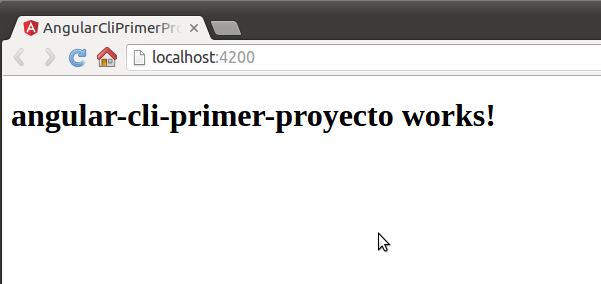
|  |  |
| --- | --- |
| 1 | ng new angular01 |

Es un proceso que durará unos 2 minutos, a cambio, al finalizar seremos capaces de ejecutar directamente la aplicación por defecto, comprobémoslo, al igual que antes accedemos a la carpeta del proyecto y ejecutamos **npm start**, si te fijas en la ejecución se llama al comando **ng serve** que será el que utilicemos con **Angular CLI**:



*Angular CLI – Consola de ejecución*

En la consola puedes ir comprobando como se detectan automáticamente los cambios para su puesta en marcha, este es el resultado una vez a lanzado **ng serve** como ves se indica la url de nuestra aplicación para probar <http://localhost:4200/>:



*Angular CLI – Resultado después de crear el proyecto*

Analizamos la estructura creada y ejecutamos el proyecto

En un primer vistazo nos podemos fijar en la carpeta src ya que será donde vamos a crear nuestro código, su contenido será similar al que hemos creado manualmente, con la diferencia de que ya lo han hecho por nosotros y solo tenemos que preocuparnos de programar, a continuación, el contenido:

*Angular CLI – Estructura del proyecto*

¿Qué más tenemos en la raíz de nuestro proyecto

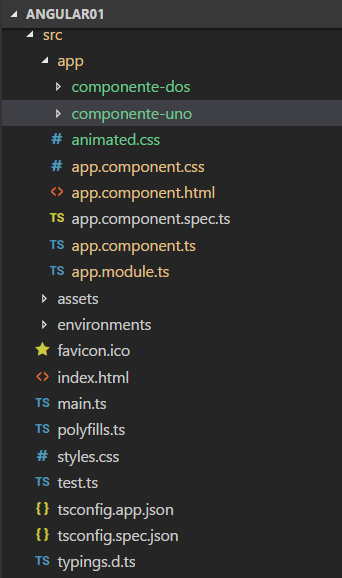
Pues ese **montonazo de código** es el que nos ha generado **Angular CLI**, esto nos puede hacer pensar en que en producción tendremos también todo, pues no es así, se generará en nuestro proyecto una carpeta **dist** con el contenido para poner nuestra aplicación en producción, y ahí, si que solo se incluyen las librerías necesarias.

Los **ficheros que aparecen en la raíz** son propios de **Angular CLI**, los cambiaremos para cambiar las configuraciones por defecto, veremos un ejemplo sencillo cuando creemos un componente.

Las demás carpetas como **node\_modules, typings** y config pertenecen a la confguración del entorno con **Angular CLI**.

Para que veas la utilidad de **Angular CLI** estas son algunas de las utilidades que trae para crear **componentes, servicios, directivas, clases**, …, en el siguiente apartado utilizaremos la de **creación de componentes**:

* **Component**: **ng g component my-new-component**
* **Directive**: **ng g directive my-new-directive**
* **Pipe**: **ng g pipe my-new-pipe**
* **Service**: **ng g service my-new-service**
* **Class**: **ng g class my-new-class**
* **Interface**: **ng g interface my-new-interface**
* **Enum**: **ng g enum my-new-enum**

Analizamos los ficheros raíz de nuestra aplicación

Para empezar nos dirigimos al directorio **app**, ahí encontramos 4 ficheros, como puedes ver todos tienen extensión **ts**, empezamos por **app.component.ts**

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11 | import { Component } from '@angular/core';    @Component({    moduleId: module.id,    selector: 'app-root',    templateUrl: 'app.component.html',    styleUrls: ['app.component.css']  })  export class AppComponent {    title = 'angular-cli-primer-proyecto works!';  autor = 'Julián Villegas';  } |
| 12 |  |
|  |  |

Esta estructura ya es conocida por que la hemos visto y [explicado en el anteriormente](http://codigoxules.org/en/creando-el-primer-proyecto-en-angular-2-manual-vs-angular-cli/#definicion-del-componente), como puedes ver ya por defecto nos ha creado el fichero para la plantilla **html** y un fichero **css** , como puedes ver se define un **array** por lo que podríamos incorporar múltiples **css**.

Viendo la plantilla **html** que tenemos en **app.component.html**, entenderemos fácilmente el código **title = ‘angular-cli-primer-proyecto works!’;:**

|  |  |
| --- | --- |
| 1  2  3 | <h1>    {{title}}  </h1> |

Aquí aparecen ya las primeras dobles llaves típicas de **AngularJS**, como puedes ver dentro de nuestro componente le damos valor a la variables.

La plantilla **css** que tenemos en **app.component.css** está vacía, podemos hacer algún cambio sencillo para comprobar la actualización automática del código, por ejemplo:

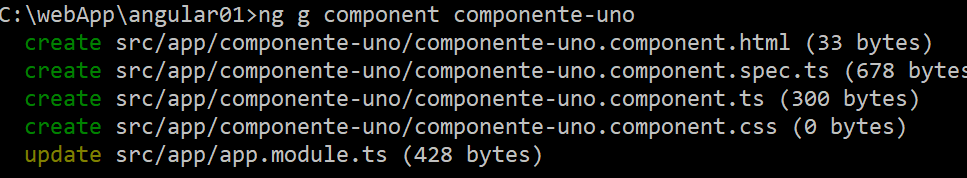
|  |  |
| --- | --- |
| 1  2  3 | h1{      color: blue  } |

Creamos un nuevo componente con Angular CLI

Como ya vimos antes la estructura para crear un componente vamos a utilizarla para crear **componente-uno**:

|  |  |
| --- | --- |
| 1 | ng g component componente-uno |

Este estos son los ficheros creados, que como puedes ver llevan el nombre del componente y son los mismos que creamos para el componente raíz:



*Angular CLI – Component – Creando el primer componente*

Este componente crea por defecto un párrafo con el mensaje: **componente-uno works!**

|  |  |
| --- | --- |
| 1  2  3 | <p>    componente-uno works!  </p> |

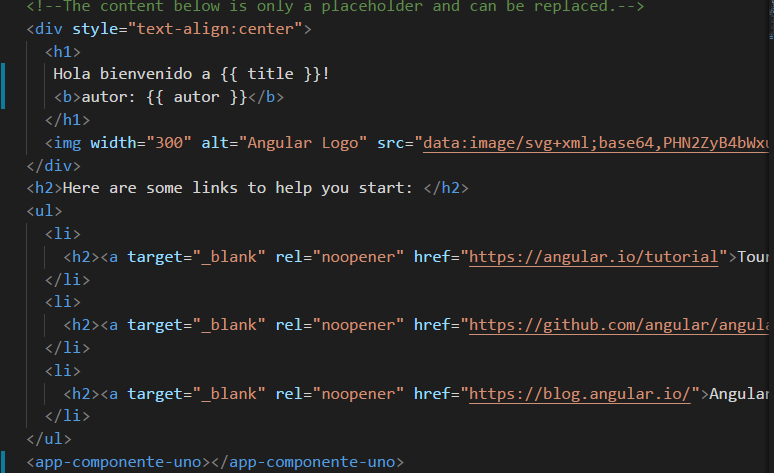
Si ejecutas ahora el proyecto cosa que recomiendo que hagas antes de hacer los siguientes pasos para que compruebes como se actualiza tu proyecto automáticamente con lo que podrás ir viendo el efecto de los cambios que hacemos, así que ejecuta **ng serve** si no tienes el proyecto en ejecución.

Antes te mostré los ficheros que nos crea para el componente, cómo pudiste ver, son los mismo que los que creamos para el componente principal de la aplicación, hasta aquí nada nuevo, este es el fichero creado para nuestro componente, únicamente te lo enseño para que veas el selector utilizado: **app-componente-uno** porque nos referiremos a él para incorporarlo al **html** de la aplicación (**index.html**):



Paso 1: Añadimos el componente al html del componente de la aplicación

Buscamos el fichero **app.component.html** y añadimos el componente así: , como ves el selector que utilizamos antes, ahora este es nuestro código:



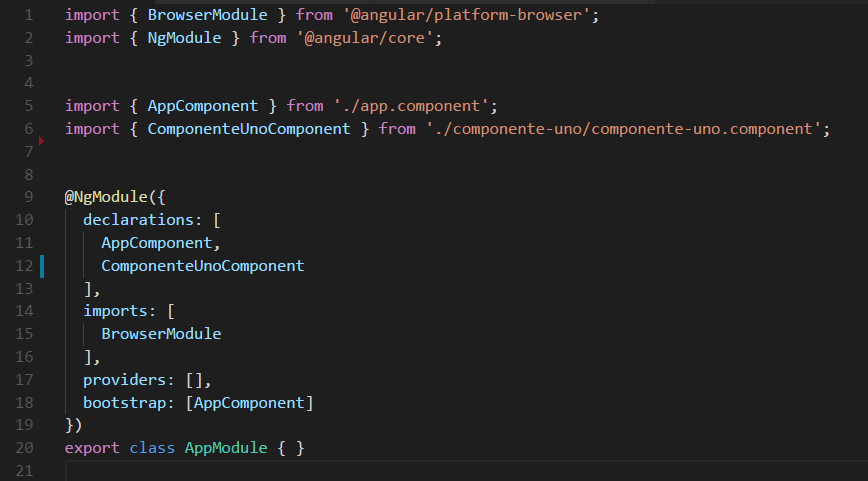
En automático se reflejará la actualización en nuestra ventana principal.

Paso 2: Necesitamos registrar el componente a utilizar para poder ser reconocido en la vista html

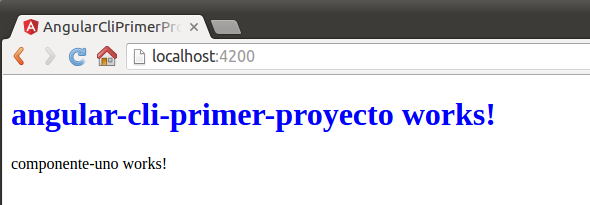
Para registrar el componente nos dirigimos a **app.component.ts**, aquí haremos dos cosas:

* Importar el componente: **import { ComponenteUnoComponent} from ‘./componente-uno’;**, la sintaxis es sencilla indicamos de donde tenemos que hacer la importación y después, declaramos el componente que vamos a utilizar, haremos lo mismo para los componentes que usemos de **Angular**.
* Registrar el componente que vamos a utilizar: **directives: [ComponenteUnoComponent], sin profundizar mucho, diremos que además de importarlo tenemos que declarar que lo vamos a usar.**

Así, este sería nuestro código final:



Si todo ha ido bien puedes comprobar como se actualizado tu página y ahora si se muestra el contenido de la página tal que así:



*Angular CLI – Resultado del componente UNO*

Creamos un segundo componente con Angular CLI personalizando su creación

Por último, para perder el miedo a **Angular CLI** vamos a hacer una modificación sencilla para actualizar el prefijo **app** por defecto que nos añado **Angular CLI** a nuestro componente anterior, para ello nos vamos al fichero de configuración **angular-cli.json** de **Angular CLI**:

|  |  |
| --- | --- |
| 1 | { |
| 2 | "$schema": "./node\_modules/@angular/cli/lib/config/schema.json", |
| 3 | "project": { |
| 4 | "name": "angular01" |
| 5 | }, |
| 6 | "apps": [ |
| 7 | { |
| 8 | "root": "src", |
| 9 | "outDir": "dist", |
| 10 | "assets": [ |
| 11 | "assets", |
| 12 | "favicon.ico" |
| 13 | ], |
| 14 | "index": "index.html", |
| 15 | "main": "main.ts", |
| 16 | "polyfills": "polyfills.ts", |
| 17 | "test": "test.ts", |
| 18 | "tsconfig": "tsconfig.app.json", |
| 19 | "testTsconfig": "tsconfig.spec.json", |
| 20 | "prefix": "app", |
| 21 | "styles": [ |
| 22 | "styles.css" |
| 23 | ], |
| 24 | "scripts": [], |
| 25 | "environmentSource": "environments/environment.ts", |
| 26 | "environments": { |
| 27 | "dev": "environments/environment.ts", |
| 28 | "prod": "environments/environment.prod.ts" |
| 29 | } |
| 30 | } |
| 31 | ], |
| 32 | "e2e": { |
| 33 | "protractor": { |
| 34 | "config": "./protractor.conf.js" |
| 35 | } |
| 36 | }, |
| 37 | "lint": [ |
| 38 | { |
| 39 | "project": "src/tsconfig.app.json", |
| 40 | "exclude": "\*\*/node\_modules/\*\*" |
| 41 | }, |
| 42 | { |
| 43 | "project": "src/tsconfig.spec.json", |
| 44 | "exclude": "\*\*/node\_modules/\*\*" |
| 45 | }, |
| 46 | { |
| 47 | "project": "e2e/tsconfig.e2e.json", |
| 48 | "exclude": "\*\*/node\_modules/\*\*" |
| 49 | } |
| 50 | ], |
| 51 | "test": { |
| 52 | "karma": { |
| 53 | "config": "./karma.conf.js" |
| 54 | } |
| 55 | }, |
| 56 | "defaults": { |
| 57 | "styleExt": "css", |
| 58 | "component": {} |
| 59 | } |
| 60 | } |

Aquí, puedes ver la versión que estás utilizando de **Angular CLI**, en el momento de realizar este proyecto estaba en la versión **beta 5** , ahora en el día de está publicación ya estamos en la beta 6, y es que lo cambios y actualizaciones son continuos mientras no sale una versión final de Angular, en este punto también se indica el nombre del proyecto.

Se registran que ficheros estamos utilizando para nuestra aplicación así como las librerías de test como **Karma**, …

Puedes ver en el apartado **defaults** la definición del prefijo **app** este será el que **Angular CLI** utilizará por defecto, en mi caso lo cambio por **cx**, con lo que el nuevo componente que voy a crear y que se llamará **componente-dos** (perdón por la originalidad) su selector será **cx-componente-dos** como veremos. Ver línea 20 del archivo de configuración.

Creamos el componente dos con el nuevo prefijo

Creamos el componente dos:

|  |  |
| --- | --- |
| 1 | ng g component componente-dos |

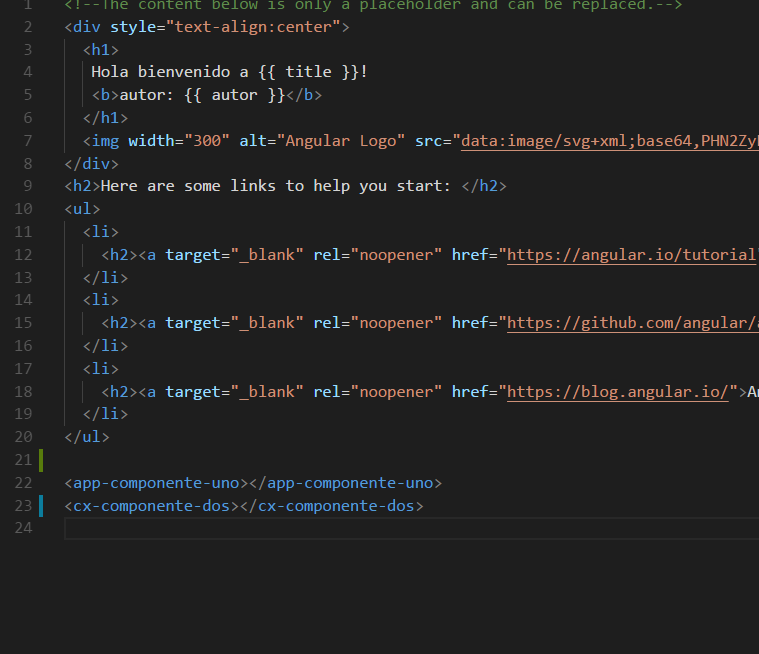
Comprobamos que los ficheros generados son los mismo que antes y que ahora el prefijo del componente dos es el que hemos definido en mi caso:



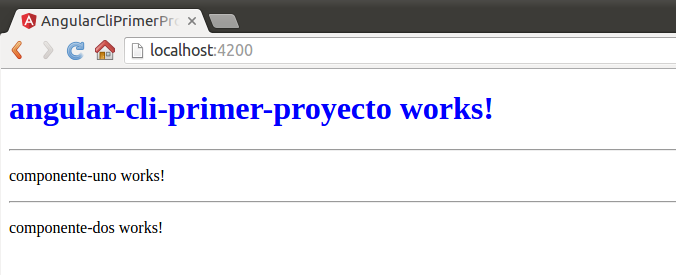
Hacemos el registro del componente y lo añadimos al html

Seguimos los pasos de antes, primero añadimos el componente al código <h1> {{title}} </h1> <hr /> <app-componente-uno></app-componente-uno> <hr /> <cx-componente-dos></cx-componente-dos>

Ahora registramos el componente dos al igual que antes primero importándolo y después, registrándolo como se puede ver en el código:



Ahora, en tu pantalla verás los dos componentes añadidos:



*Angular CLI – Resultado del componente DOS*

Con estos dos sencillos componentes pretendía simplemente introducirte en la forma de trabajo de **Angular 2**, en próximos tutoriales nos adentraremos ya en la creación de una aplicación sencilla con Angular.