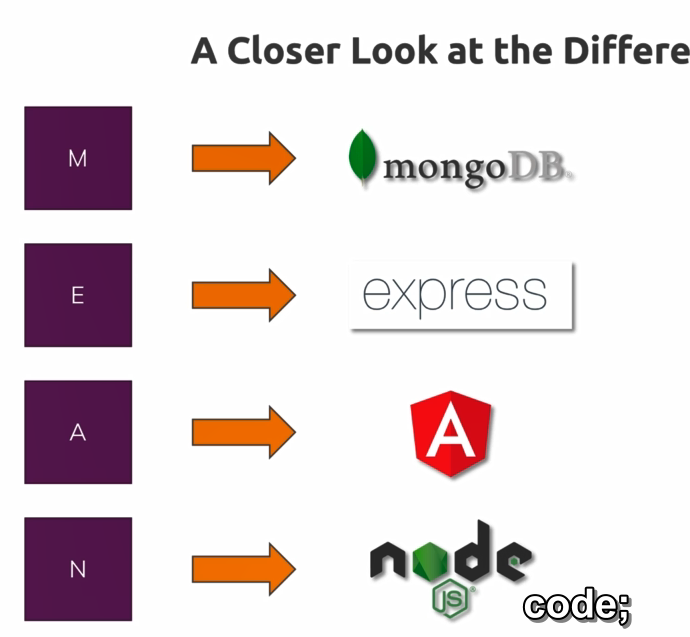
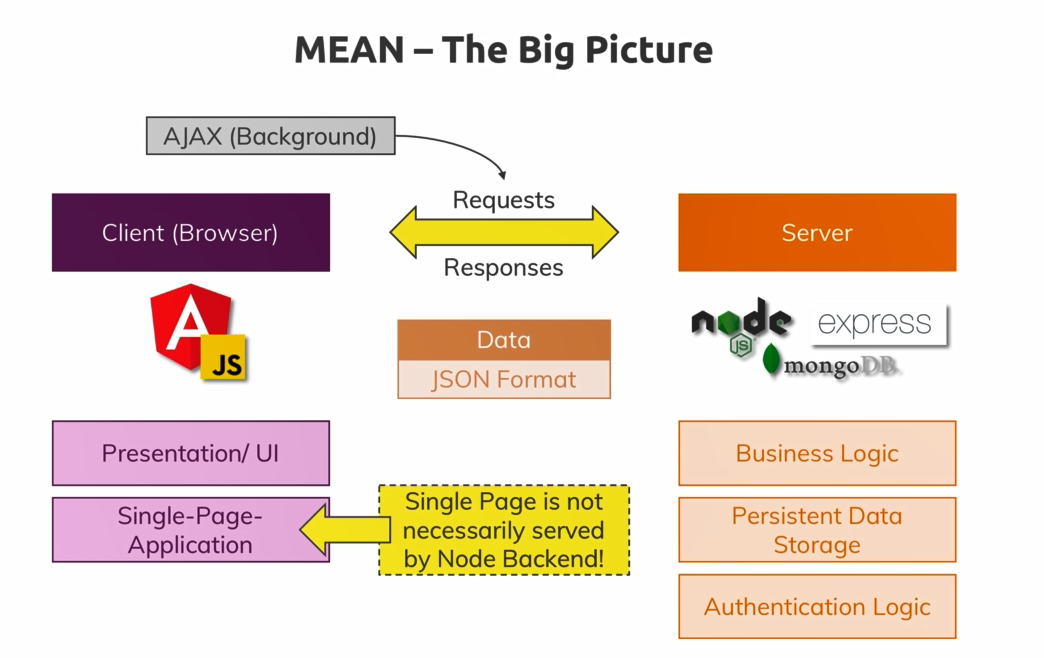
Mean Stack

Mongo Db: base de datos  
Express: framewok de nodeJs  
Angular: Client Side  
N: Node Logica en el servidor, validación de usuarios etc.  


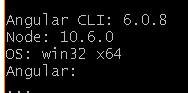




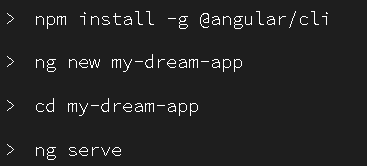
Es necesario bajar e instalarnode.

<https://nodejs.org/es/>

Verificar versión de Angular Cli  
ng -v



Comandos basicos



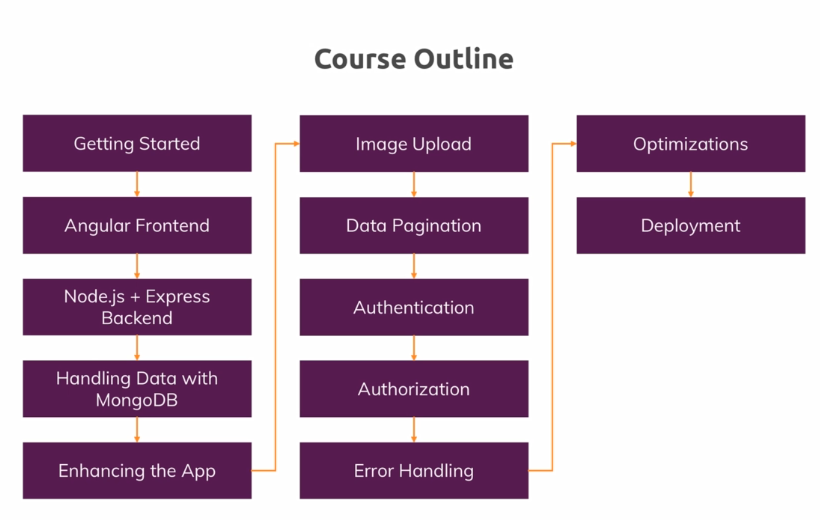
npm install @angular/cli@latest

Crear el Proyecto, y cambiarse a la carpeta.

Como editor de texto se usara visual studio Code

Instalando algunas extensiones>

Angular essentials



La estructura inicial de la pagina en angular es



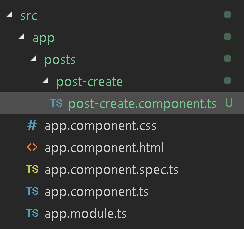
Estando el componente app-root dentro del archivo index.html

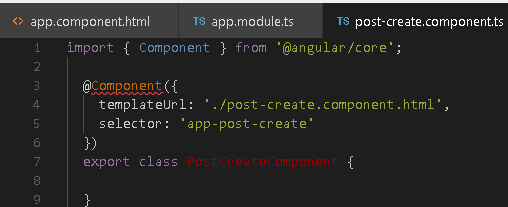


El app module, es el conector principal de los modulos que se prensentan en el single page application.

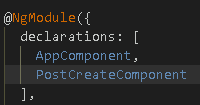
Creando el primer componente

Se crea la estructura de un folder con otro subfolder dentro que tiene un archivo post-create.component.ts

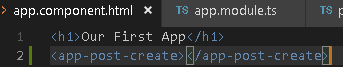


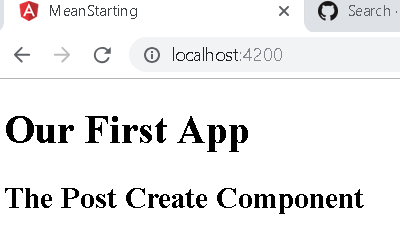


Se crea el el html un contenido, se agrega este componente al app.modules



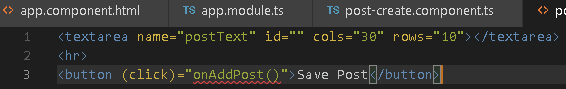
Y luego ya podemos utilizarlo en el app.html

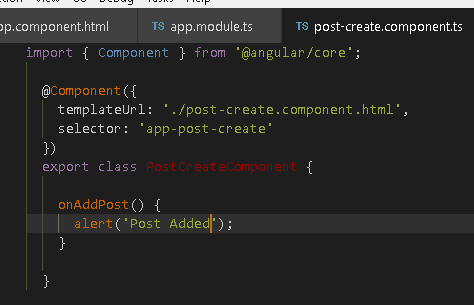
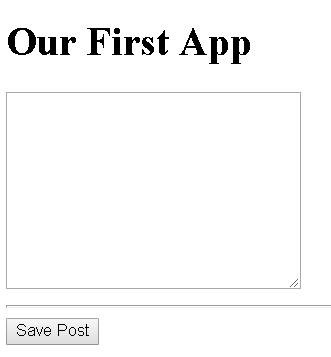




Listening to events

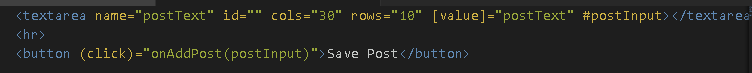
Creando un html básico para hacer un post

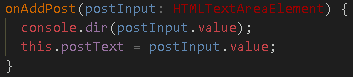


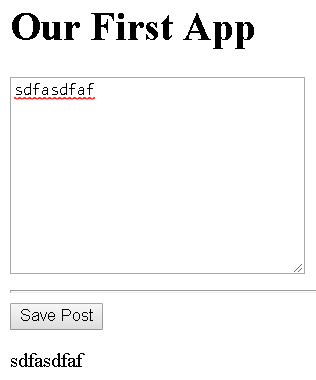


Mostrando lo que introdujo el usuario.

Para hacer referencia a un elemento del html, en este caso un input, se utiliza # para crear una referencia y luego poder pasarlo en una función como parámetro





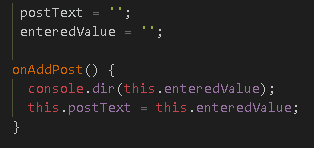


Existe otra forma de unir el elemento como un two way binding, utilizando la directiva [( ngModel)]

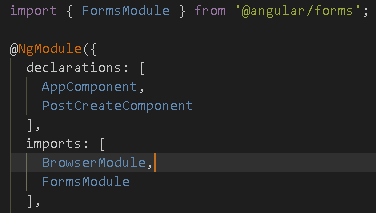
En el html utilizamos la directiva y se asigna a una variable que luego se crea en el ts

Sin ser necesario pasar el valor de la variable como parámetro.





Y finalmente para poder utilizar el ngModule es necesario importarlo en el app.modules



Para borrar node modules

npm install rimraf -g

rimraf node\_modules

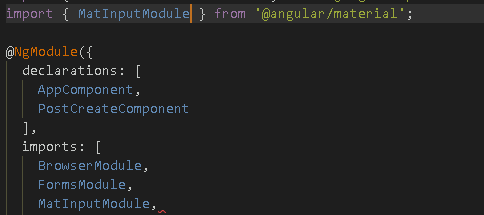
Instalando Angular Material:

Npm install –save @angular/material

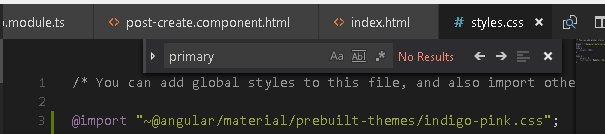
O con la ultima versión del cli

Ng add @angular/material

Luego de instalar el angular material podemos ver en la documentación de la pagina oficial los componentes que podemos utilizar, por ejemplo un input.Es necesario importarlo en el app.modules.



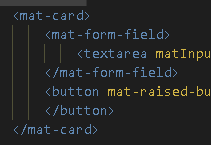
Importar en el styles.css



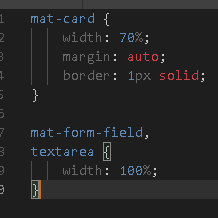
@import "~@angular/material/prebuilt-themes/indigo-pink.css";

Manejando la nueva estructura con el mat-…

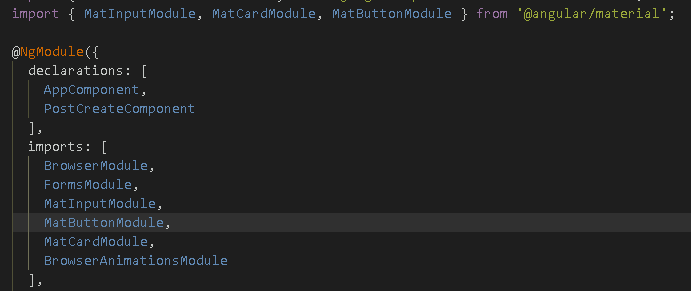
En el html…



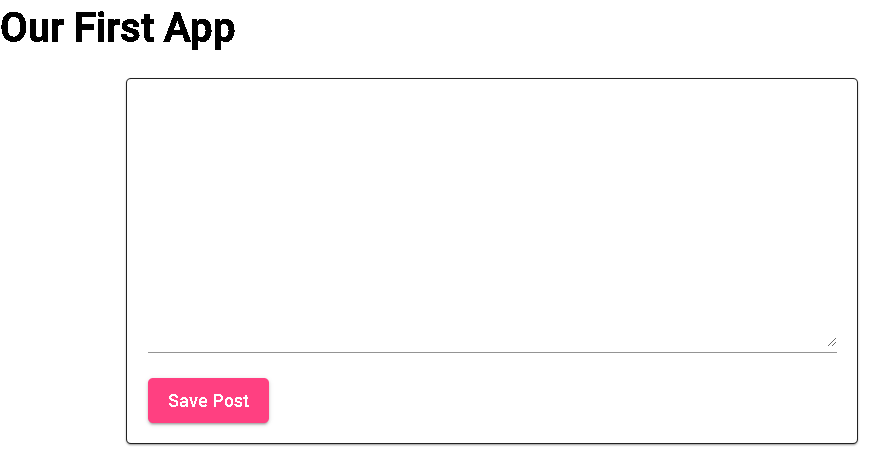
Css…



App.module importando lo necesario que estamos utilizando, botones, card, etc



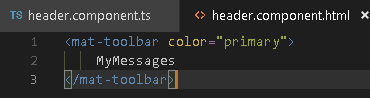
Resulta:



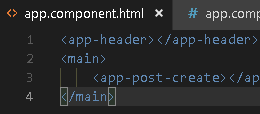
Creando un toolbar (header)

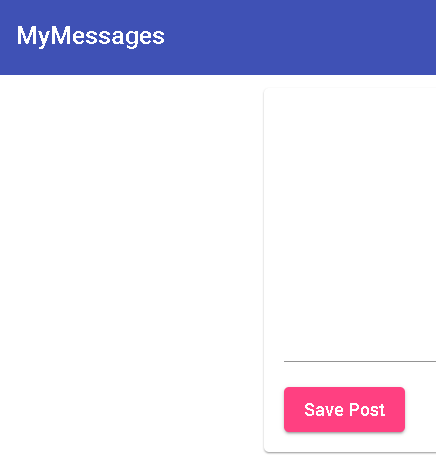
Se crea unnuevo folder y componente , header

En el html se implementa el tag mat-toolbar, que debe ser importado en el app.module.



Desde el app.component

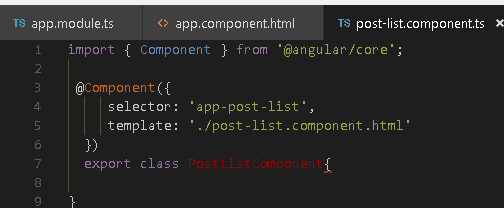




Mostrando los post.

Se crea una subcarpeta, post-list con su ts y html, recordar incluirlo en declarations.

Y luego en el app.html con su tag

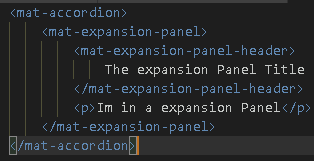


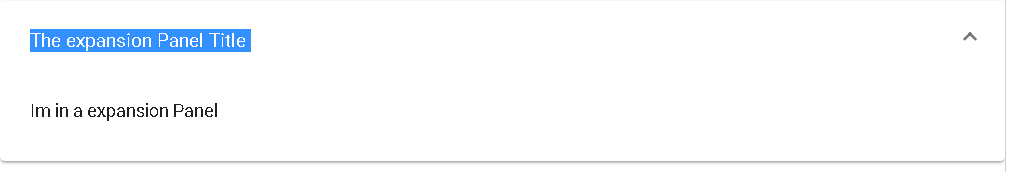


Ahora importamos un nuevo modulo de material,

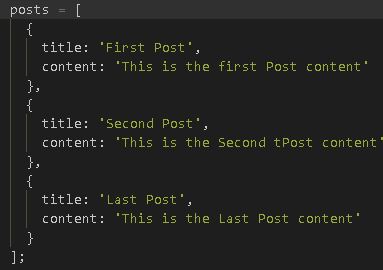


Ahora en el html se utiliza una nueva estructura:

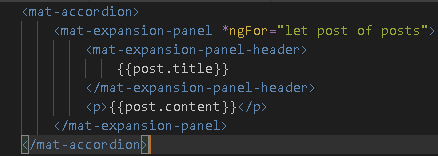




Creando una lista de posts.



E irteandola en el html:



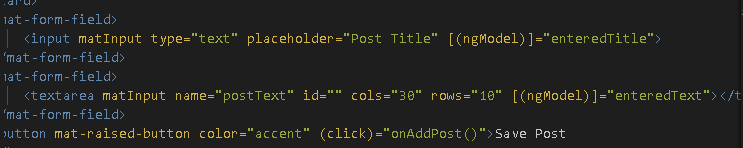
Creando la estructura que reciba los post desde el input, se declara el arreglo posts vacio.

Y la estructura en el html cambia.



Para ingresar un post según la estructura definida se necesita un titulo para el post, asi que el post input cambia añadiéndole otro input.

Se crea el imput en el html y usando el [(ngModel)] se enlaza a las variables creadas en el ts.



Pasando la información de un nuevo post con un emit. Desde el créate component al app.component. Es necesario crear una nueva variable de tipo EventEmitter()  
poner el decorador Output para que este pueda ser escuchado desde el exterior del componente.  
  
Luego dentro del método crear un objeto con los valores ya entrados y enviarlo con la propiedad emit de la variable evento creada.



Desde el app component html, se enlaza con el evento que se creo para que escuche y apenas se emita el evento se llame a una función que llenara un arreglo local de posts para luego ser utilizado.  




Ahora recordando el arreglo posts que tenemos en post List, este se puede enlazar con otro valor que este fuera del componente, utilizando el decorador de Input().

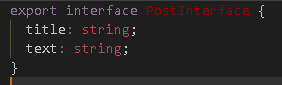


Y en el html del app component se enlaza el valor del arreglo local de posts, con el del post-list que se tiene como un decorador imput.  


En este momento se están utilizando post desde: el créate, list, app component

Es necesario crear un modelo de post, para asignárselo a las variables para que esten mas estructuradas.

Creamos entonces un model.ts para luego asignárselo a las variables de este tipo.

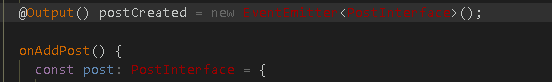




en el list.component

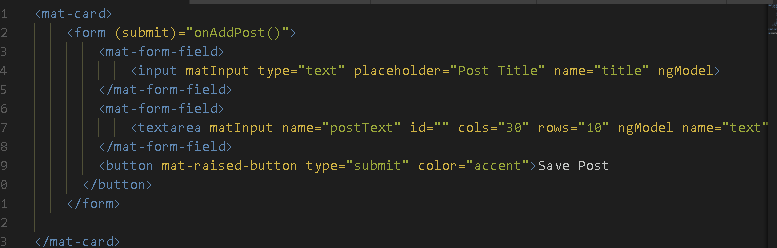


Y en el créate Post



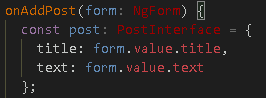
Añadiendo forms:

Para no utilizar el ngModel podemos manipular las entradas del usuario con un formulario, se declara y dentro se pone el contenido de los inputs:

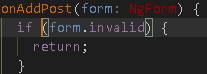


Es necesario crear una referencia del formulario para poder acceder a los inputs.



Y ahora desde el post create .ts recibimos como parámetro el NgForm (Que debemos importar) para luego poder acceder a los valores que retorna:  


Se pueden añadir validaciones desde el html añadiéndole un required a los inputs, y desde la función comprobar si son validos los datos ingresados en los inputs:



Se puede añadir un mensaje de error al html



**Implementando un servicio**

Cambiando un poco la metodología de enlazar los datos entre componentes, ya que puede resultar muy engorroso si exiten muchos componentes que necesiten el mismo dato. Es mejor practica utilizar un servicio.

ng g s posts/post

Al poner injectable no es necesario introducirlo en el app.module



Desde el método de getPosts se utiliza la nomenclatura de [ ] para hacer una copia del arreglo, y asi no se vea afectado su contenido desde el exterior con estemetodo.

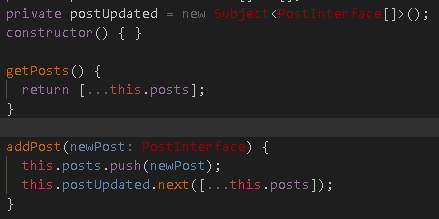


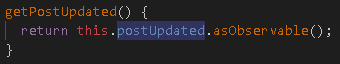
Ahora podemos implementarlo en cualquier componente, importándolo y declarándolo desde el constructor.



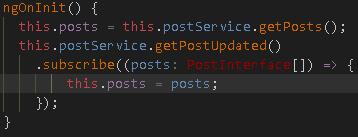


Ahora desde el servicio hay que importar Subject desde rxjs para trabajar con observables y asi poder escuchar cuando se introduce un nuevo post y mostrarlo



Ahora para obtenerlo desde afuera se declara una función que va a retornar un observable de este arreglo declarado en el servicio.  


Desde el componente que queremos escuchar se llama a este método , y nos subscribimos a su respuesta.



Al momento de subscribirnos, existe la posibilidad que este componente no forme parte de la vista en algún momento, en este caso hay que cancelar la subscripcion para que no quede en memoria.



Se le asigna el valor dl observable, añadimos el onDestroy a la clase y añadimos el método .

