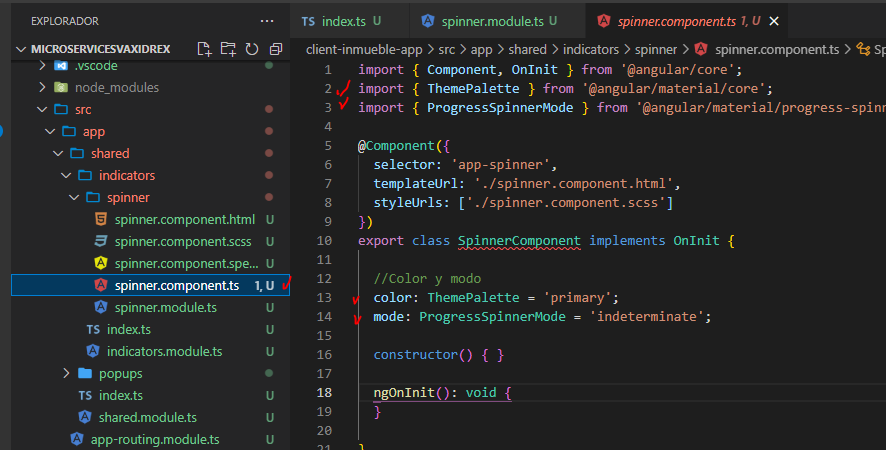
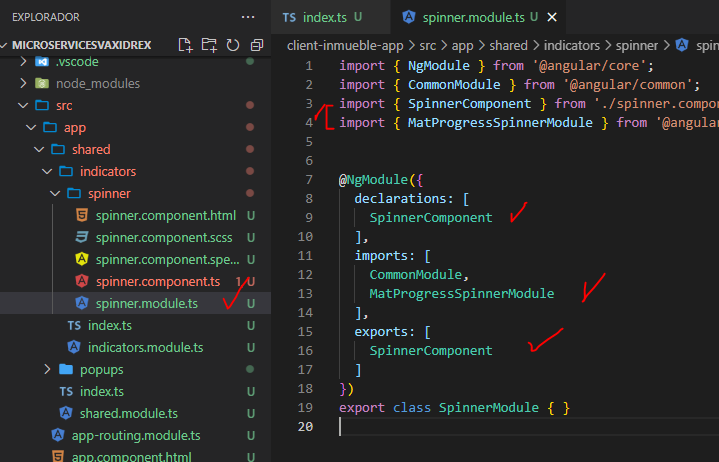
16. Creacion de Controles Shared

Creacion de Spinner

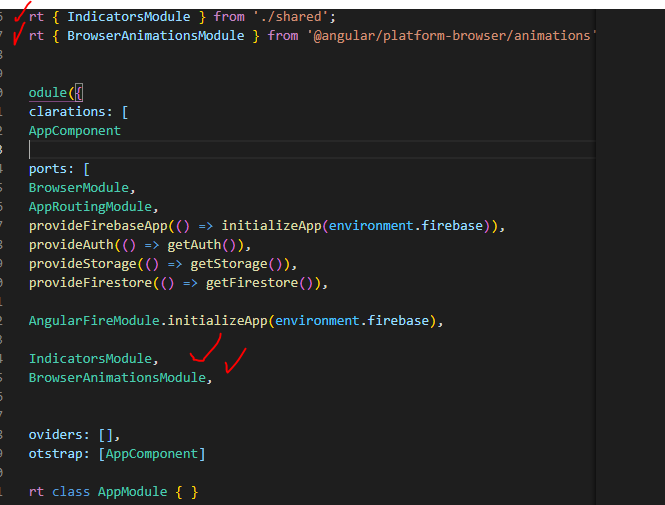
Controles compartidos dentro dentro de shared

* Indicators
* Spinners

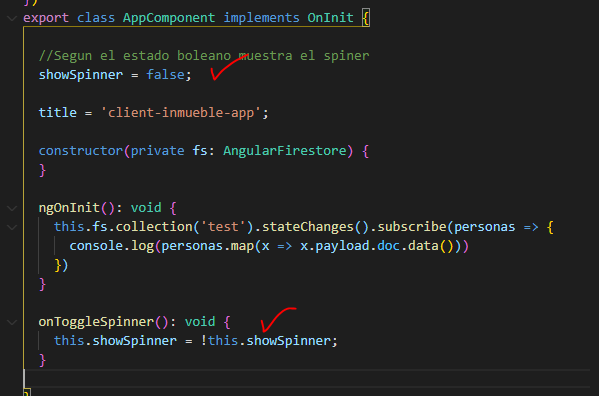
Configurando el spiner – realizo lños ajustes en el componente html y css

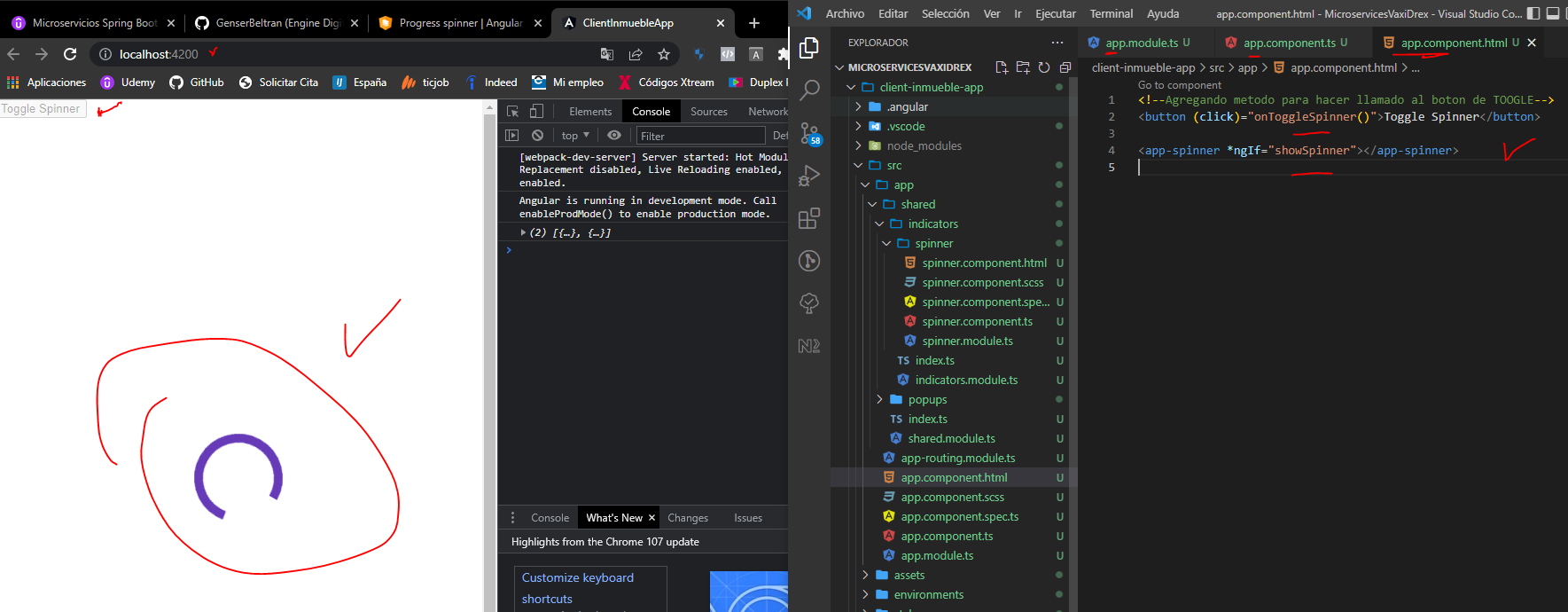


Para visualizarlo… IMPORTO la característica al app module.ts



Creacion del método para cambiar el estado del espiner de ese modo previsualizar



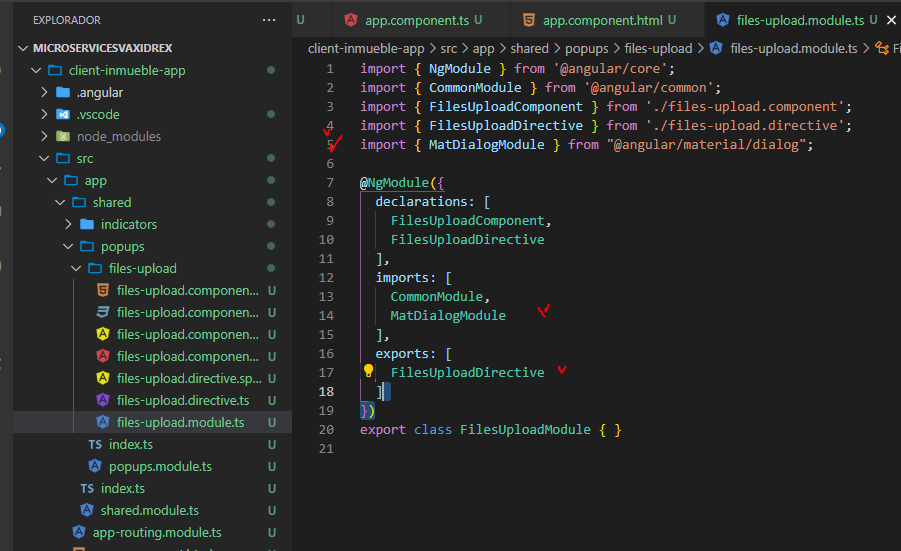


Estructura de Control Upload

Creo directivas para la carpeta files-upload

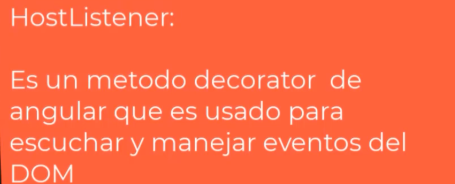
ng g directive shared/popups/files-upload/files-upload

Para tener el control de upload



La carpeta componente representa los componentes gráficos

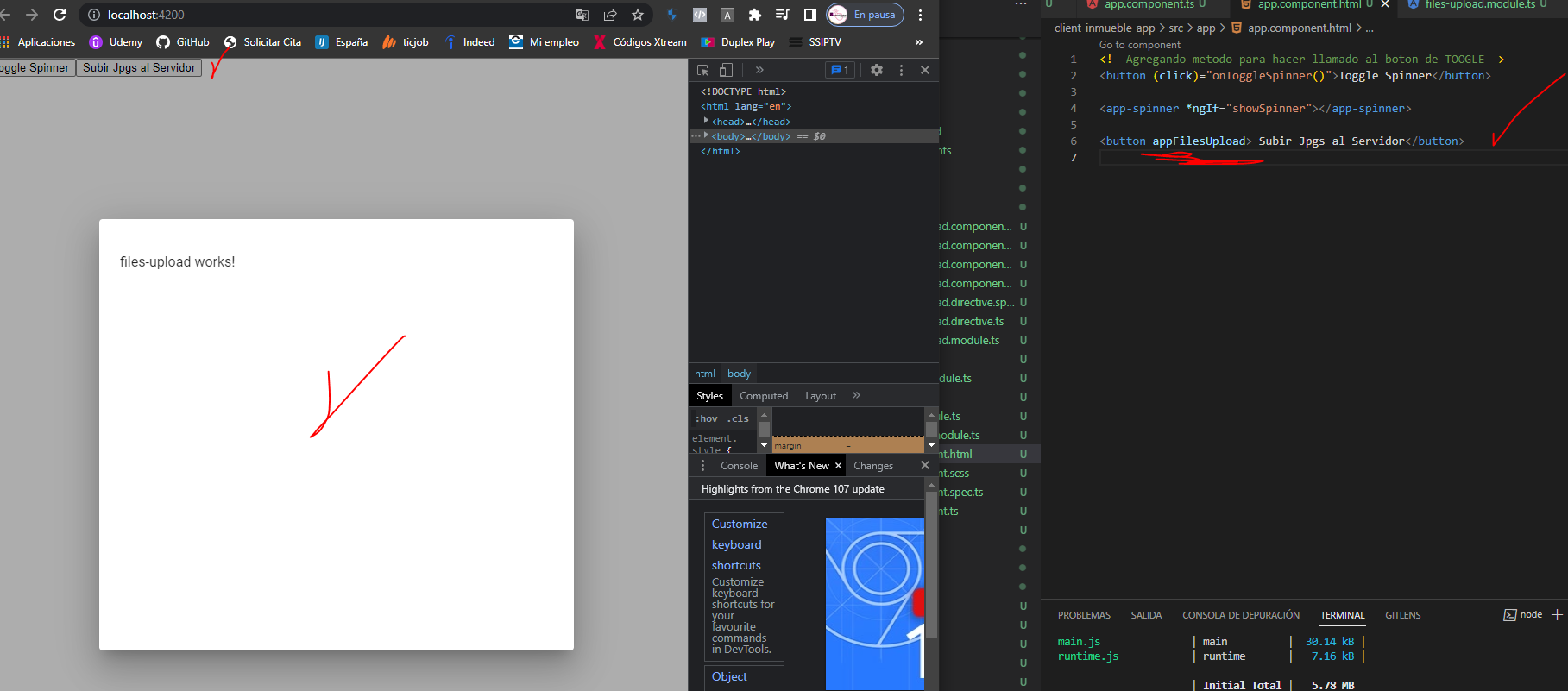
Dialog en Angular



Funciones generales para el upload   //Popiedades importantes Control de Multiples archivo o solo Uno - Crop edicion de la imagen, en el archivo FilesUploadDirective

Como siempre ya esta listo para utilizar el modulo, agrego el modulo al app.module.ts y luego agrego la línea de código al app.html

Prueba



 Template de dialog

Ajusto los archivos.

* files-upload.component.scss (Cuidado cuando importo las rutas PADRES)
* files-upload.component.html

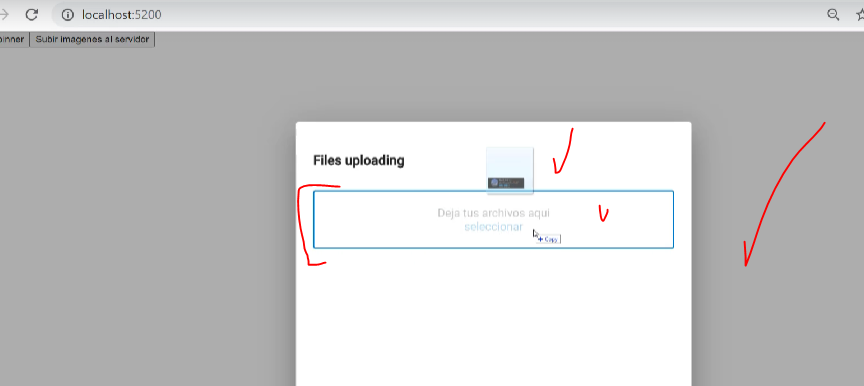
Directiva drag and drop de archivos

Directiva que manejo los eventos de drag an drop, en la siguiente ruta se crea la directiva

ng g directive shared/popups/files-upload/directives/drop-zone/drop-zone

Y configure el archive drop-zone.directive.ts para agregar la directiva al files-upload.component.html y luego realizar los ajustes de estado en files-upload.component.ts, e indicamos que exporte el drop-zone por medio del drop-zone.index.ts

Prueba

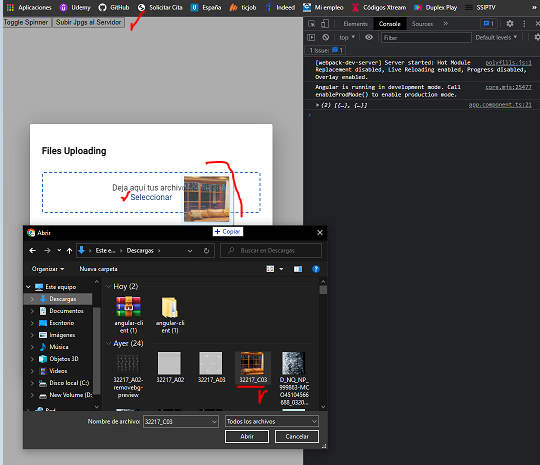
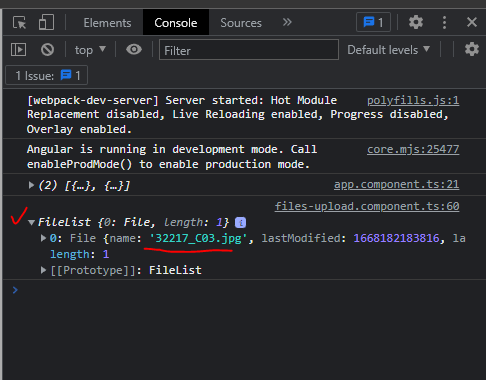


Almacenar archivos temporales en angular

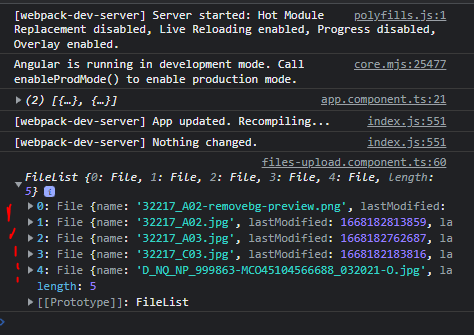
Capturar la data en files-upload.component.ts para luego modificar files-upload.component.html y agrego las propiedades de la data files-upload.directive.ts.

* Para manejar los mensajes de error los agrego al html y procedo a darle los estilos desde
* Crear método para que dispare las imágenes que se enviaran al servidor , se crea en files-upload.component.ts

Prueba

Como vemos Subo la imagen con el nombre y en consola se almacena este. Esto puede ser para una jpg o multiples jpg (ajuste en app.component.html con el atributo de multiple)



 Subir archivos en Firebase

Creamos u n componente en la carpeta components ng g c shared/popups/files-upload/components/upload reviso que en files-upload.module.ts se haya registrado el componente si no lo registro.

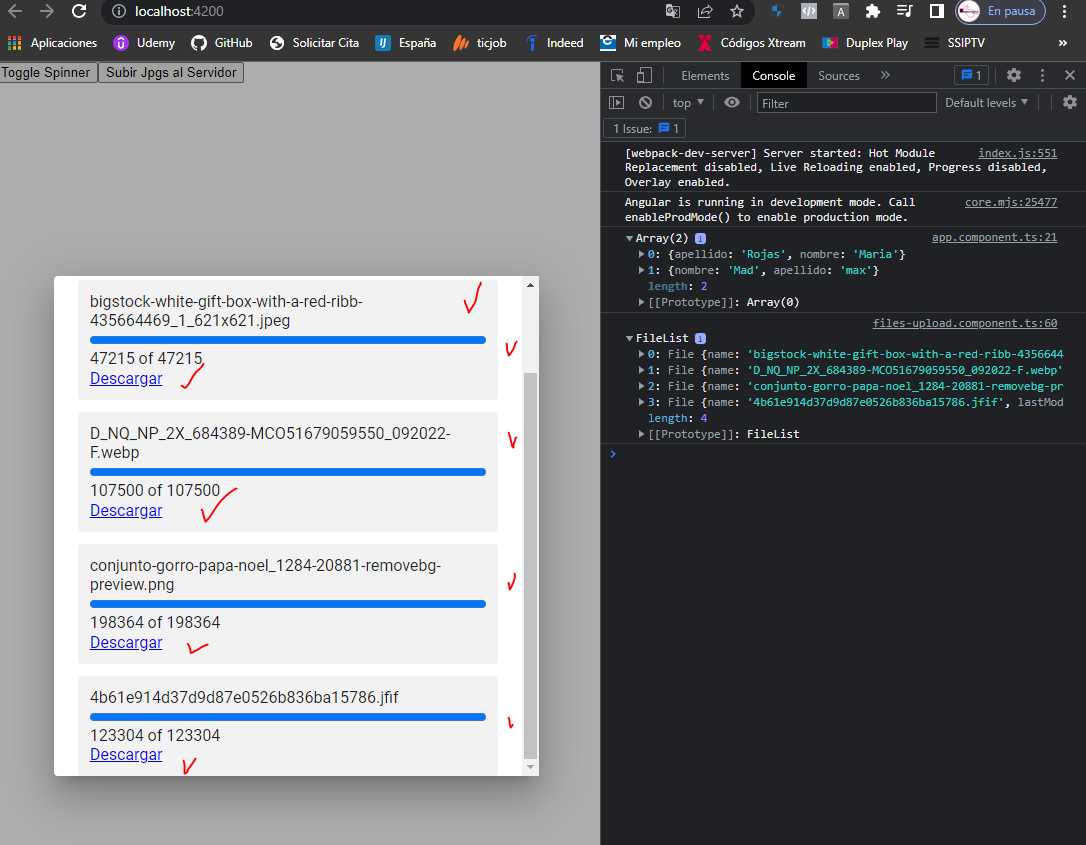
Configuro upload.component.ts implemento OnDestroy.

Clase interesante (Para identificar qwue tipo de archivo estamos almacenando jpg, Word, png … etc)

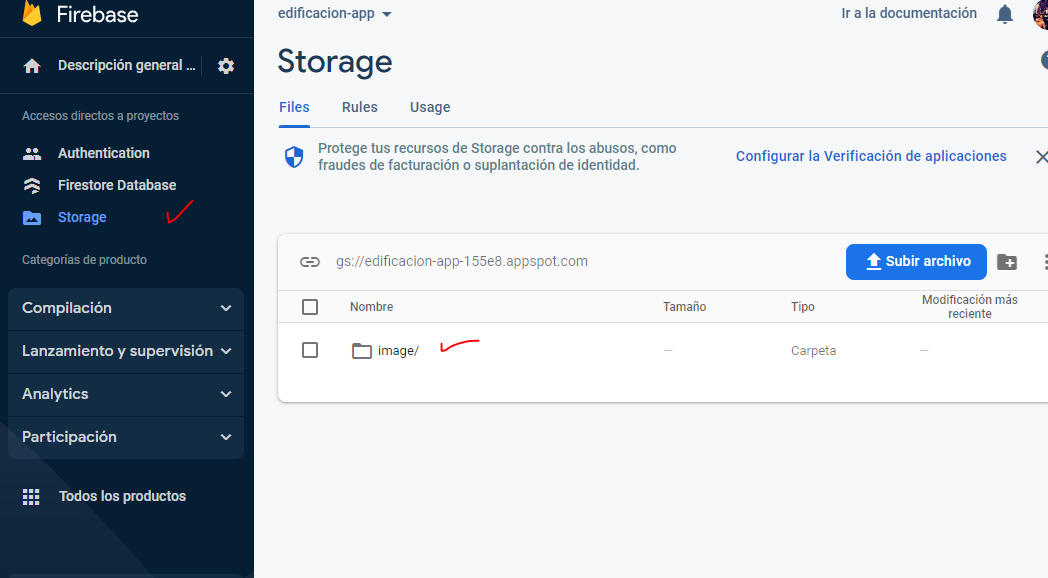
Por nomenclatura un observable siempre se le coloca signo de DÓLAR al final, luego de configurar la logis procedo a realizar el html

* Procedo a realizar los austes de scss upload.component.scss
* Configuro nuevamente el files-upload.component.html para agregarle la condición que que el container contenga archivos. Tambien otro container para que muestre la lista de archivos (files ) que se subirán a firestore
* Creo el método onLoadCOmplete para el archivo files-upload.component.ts
* Modifico el ng-templete de upload.component.html para que me muestre la URL AL FINALIZAR la carga

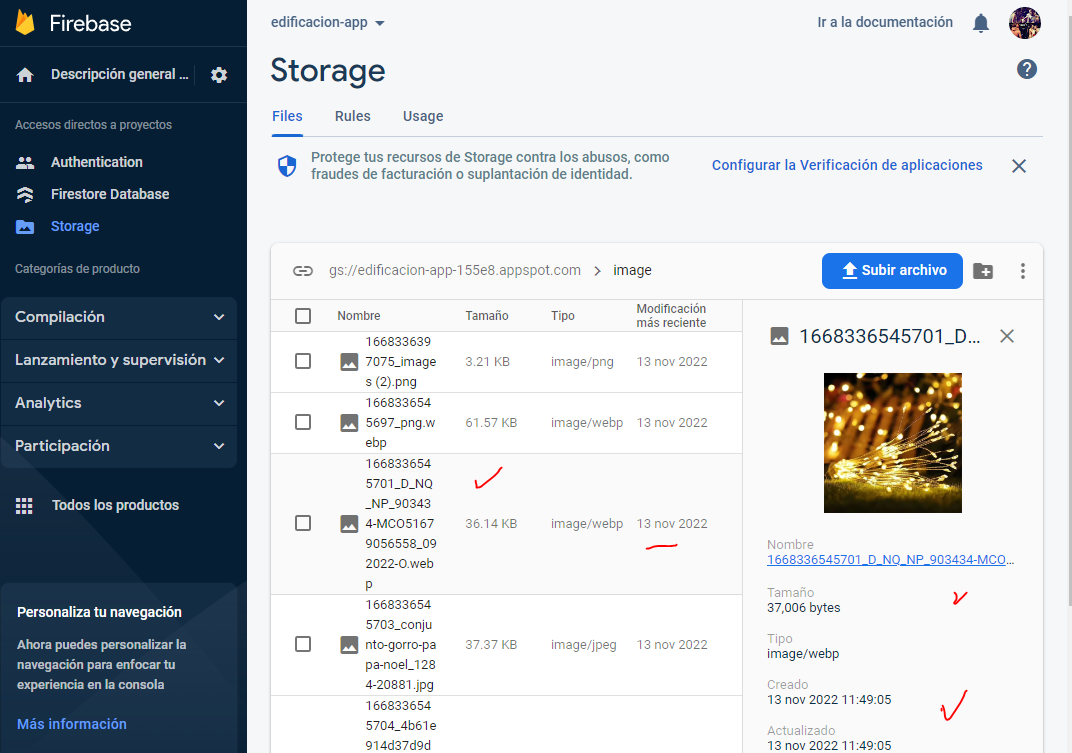
Pruebas



Y en firestore ya están quedando las imágenes con su url



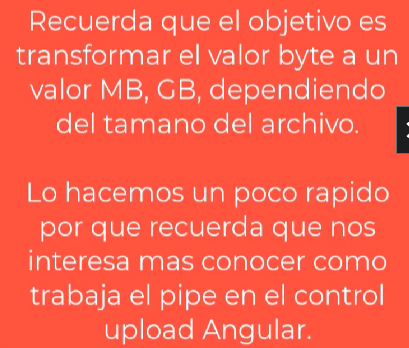
Clase Op



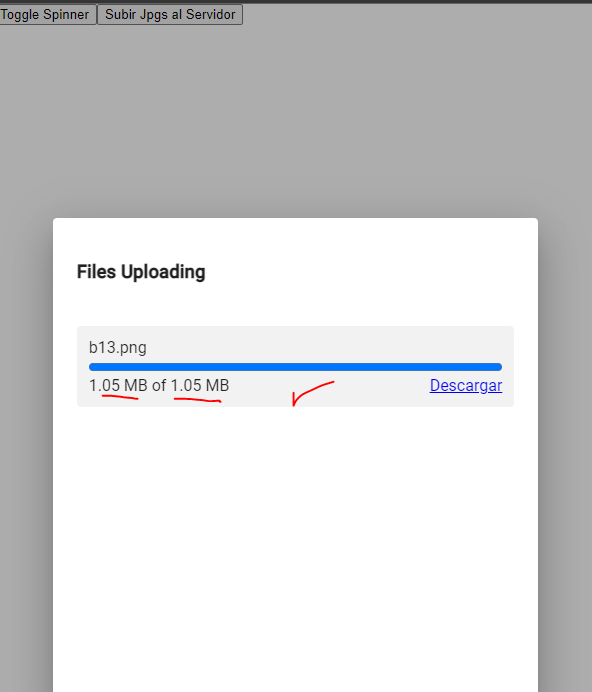
Transformar Label con Pipe

El peso del archivo se esta subiendo en bytes y lo pasaremos a MegaBytes

* Crear pipe ng g p shared/popups/files-upload/pipes/file-size/file-size y lo configuro file-size.pipe.ts



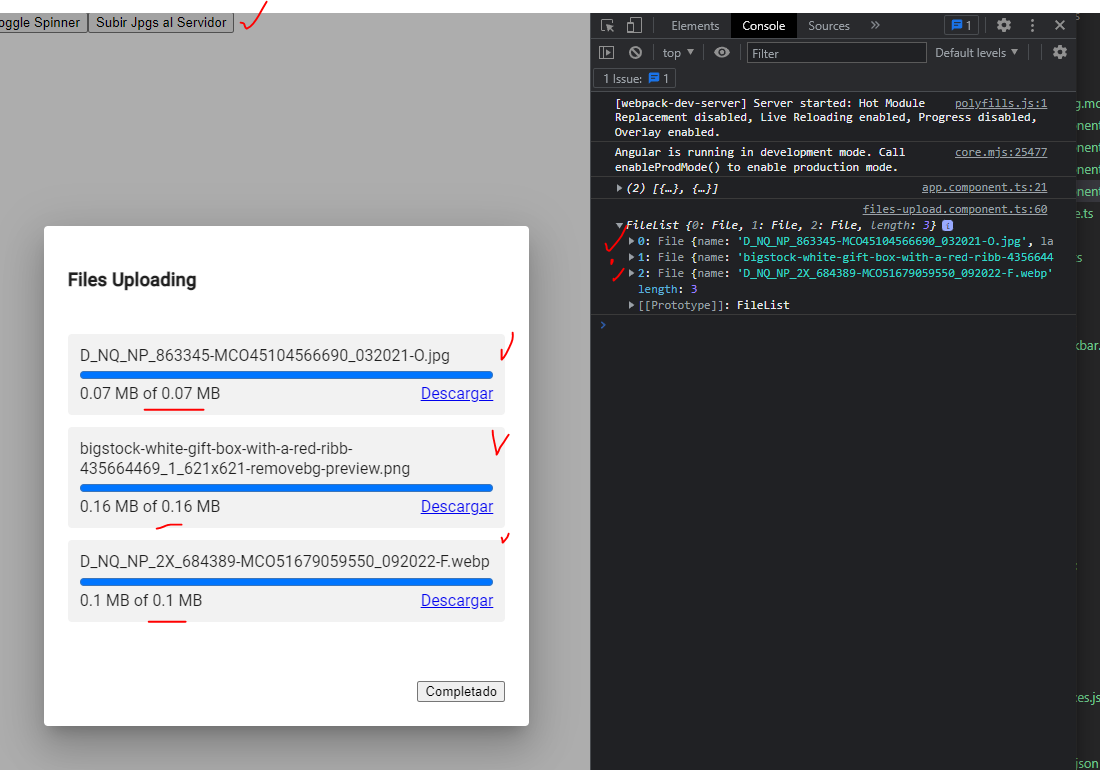
* Ajusto el upload.component.html ern la línea 11 el comando sera convertido al pipe filezise
* Prueba



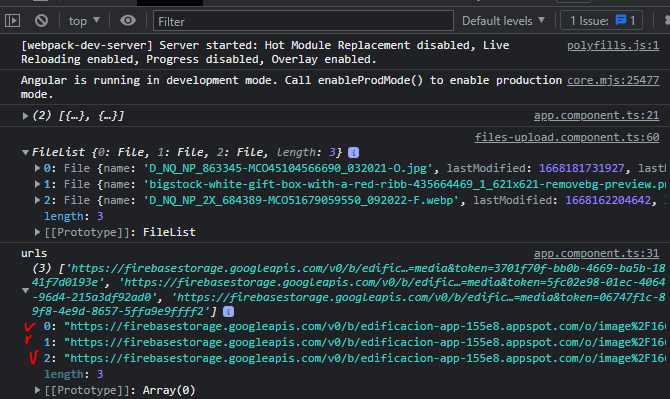
Retornar URL de imagen

EL objetivo es que uando cierre la venta nos retorne la url de la jpg

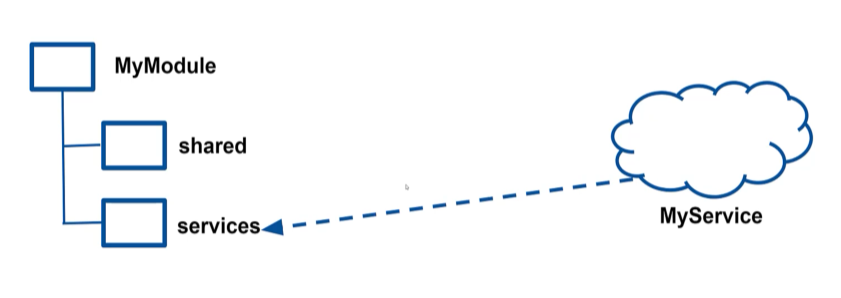
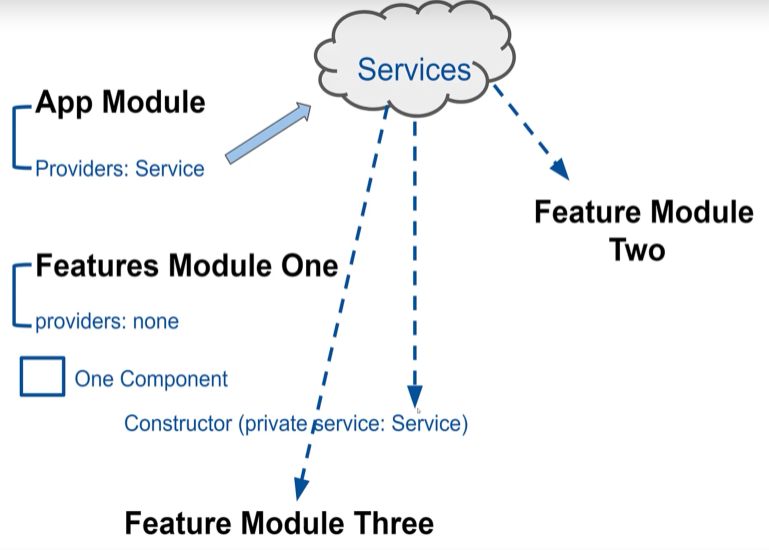
* Modifico files-upload.component.html
* Agrego los cambios a files-upload.component.ts
* Modifico la directiva files-upload.directive.ts
* Agregar el evento que se dispara cada vez que se cierre el modal app.component.html
* El evento tiene todas las url que se han subido a Firebase, por medio de un método creado en app.component.ts extraeremos dichas url
* Pruebas – Agregando las imágenes ya se habilita el botón de completado para almacenar en firebase



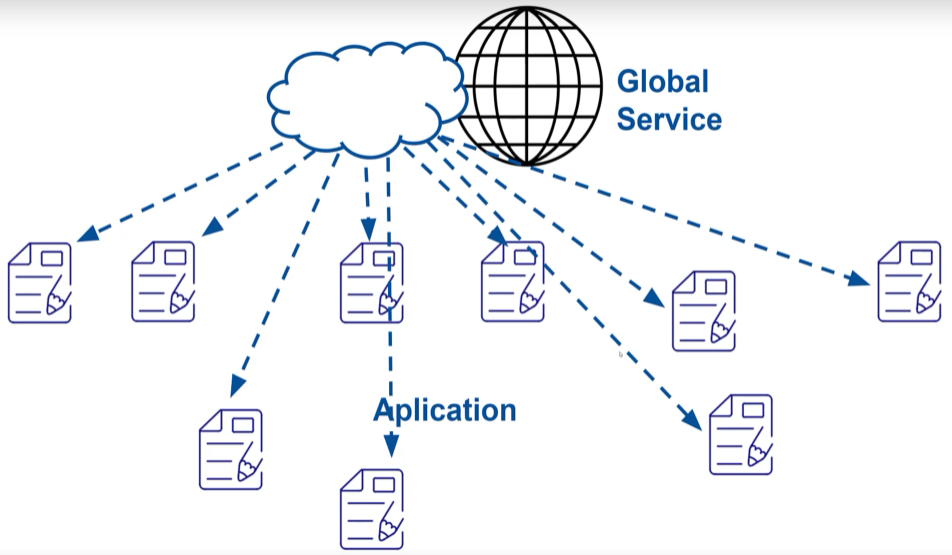
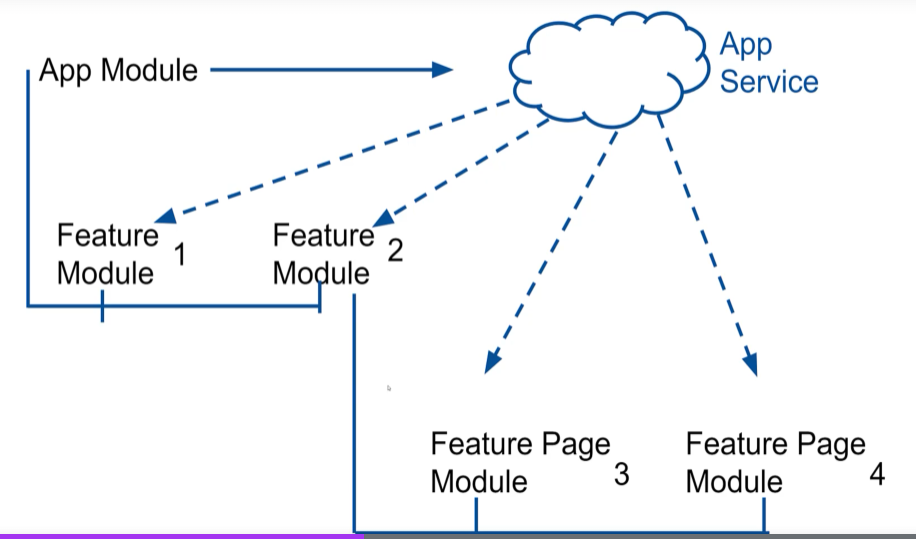
Cuando precioso el botón de completado llega el disparo con la URL de cada imagen lista para almacenar en nuestra DB

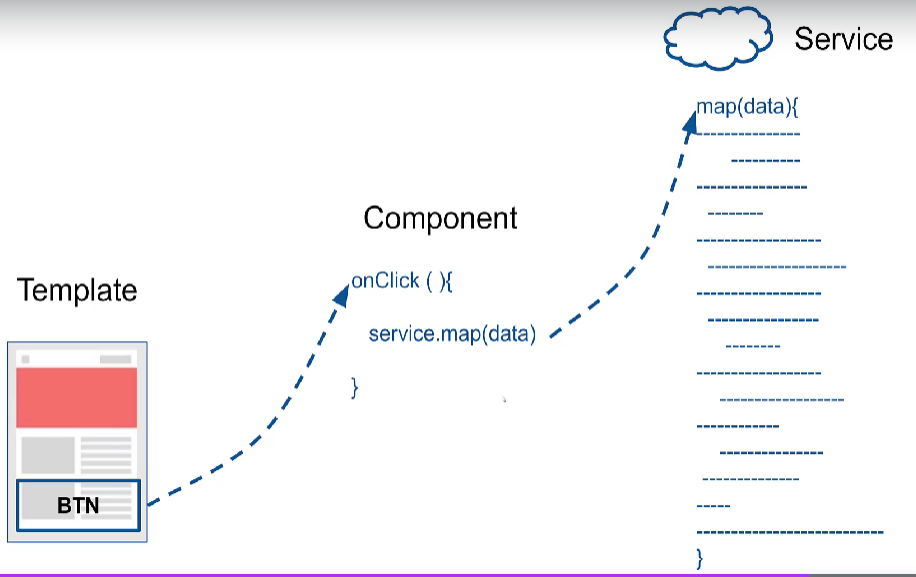
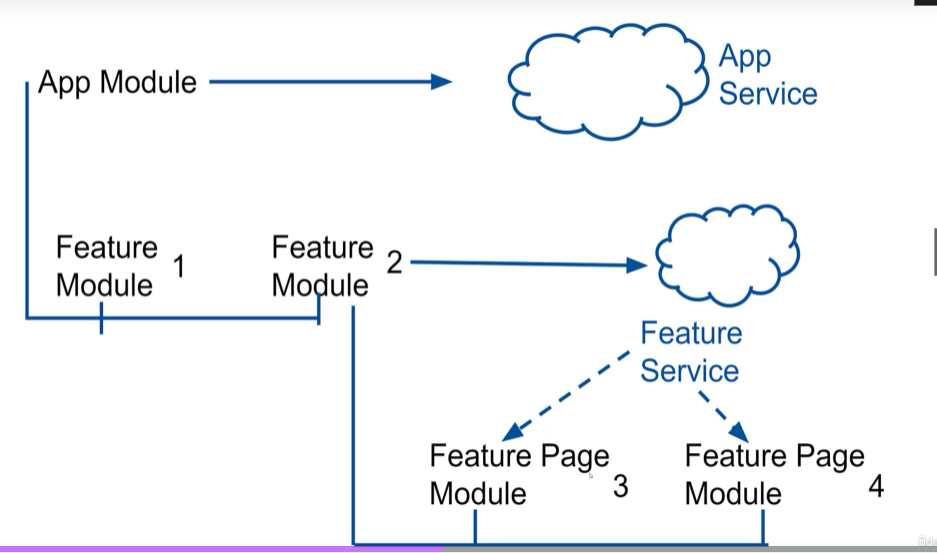


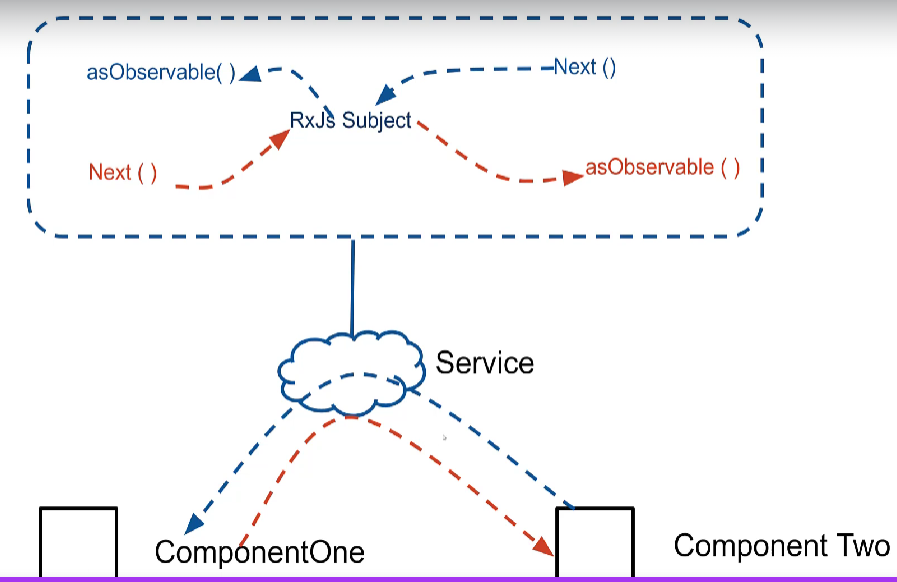
Services en Angular



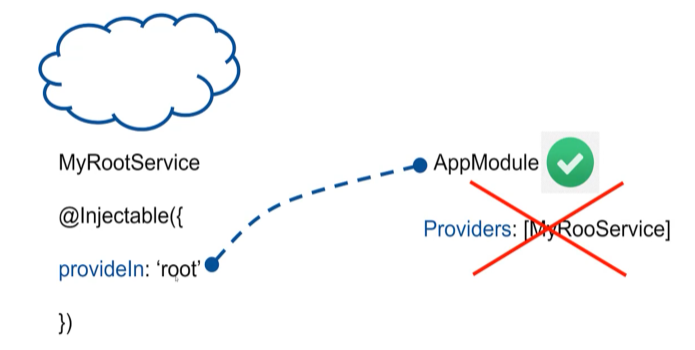
Podemos construir servicies a nivel global, local(Conocido como complementari) y comunication(Bidirecional comunicación entre dos componentes)

)

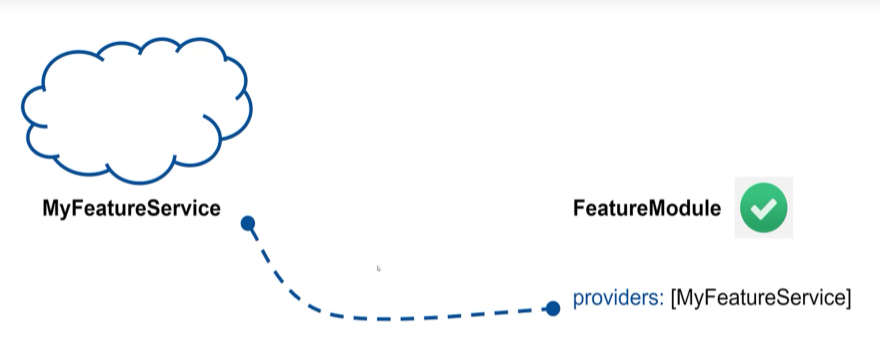




SI creo el service por comando se agrega en automático al appModule



Pero cuando es local dse debe remover del Modulo general y pasarlo al local



Creacion de Snackbars - Mensajes del sistema

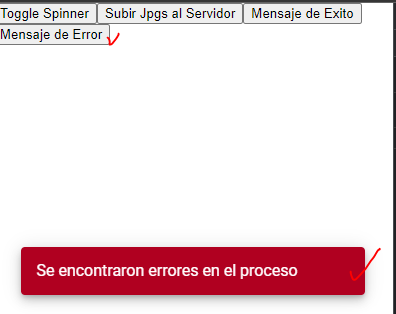
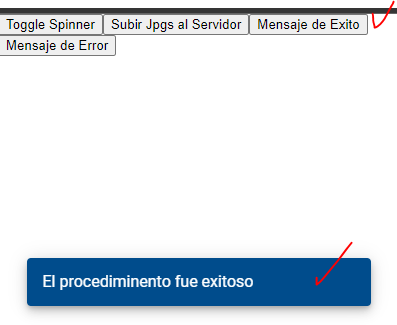
Logico y componetne grafico para mostrar snabars de mensajes luego de una acción.

* Creo la carpeta servicios app.services
* Creo un modulo nuevo dentro de la carpeta services ng g m services/notificación
* Creo el servicio ng g s services/notificacion/notificación
* Configurando notificacion.service.ts como es un servicio local en el ijectable le quito el root ysolo queda injectable()
* Configurar notificacion.module.ts para que solos e habilite el servicio una vez iniciado el proyecto
* Exportar el service y el modulo en Notificación.index.ts
* Importo el notificaionModule a app.module.ts

Procedo a crear el componetne grafico para mostrar el mensaje según la acción carpeta, notificación.component y creo el componente ng g c services/notificacion/components/notificación

* Hago exportable el componente creado en NOTIFICACION/components.index.ts
* Agreggo el component creado a notificacion.module.ts y en el impor agrego el MatSnackBarModule
* En notificacion.service.ts agrego los métodos que necesito para desplegar un mensaje de éxito
* Modifico el notificacion.component.ts que recibirá los parámetros del mensaje y será una interface
* Procedo hacer los ajustes en notificacion.component.html
* Modifico el app.component.ts
* Procedo a realizar los ajustes en app.component.html

Pruebas



Lo interesante es que los artefacto se hicieron en un servicio global y se pueden utilizar para cualquier parte del proyecto