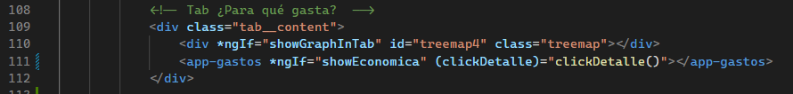
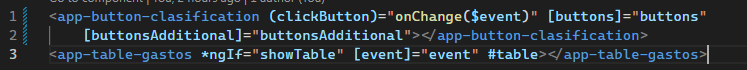
­­Documentación gastos

detalle-presupuesto.component

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

ckeckbox

Texto

Descripción generada automáticamente

gastos.component

Como en el HTML usamos \*ngIf="showTable" cuando lo ponemos a false se elimina el componente del DOM y cuando lo ponemos a true se vuelve a crear el componente y se ejecuta el ngOnInit.

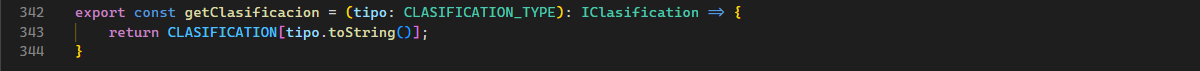
**Si recargamos la app los datos de los services se pierden.**

Texto

Descripción generada automáticamente

buttons se definen en data-table.ts

Y se recuperan llamando a la función getClasification()



button-clasification.component

Texto

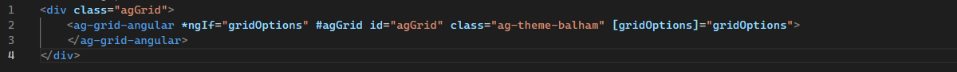
Descripción generada automáticamentePantalla de un computador

Descripción generada automáticamente con confianza media

Tabla

Descripción generada automáticamente con confianza baja

table-gastos.component

Texto

Descripción generada automáticamente

**State managment**

Trato de encontrar el método más adecuado para manejar el state en la app.  
 Las opciones que tengo son:  
 1.- Utilizar un service que maneje cada propiedad:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamenteEn el componente que necesite usarlo, en este caso table-gastos.component.ts, lo inyecto en el constructor y grabo los cambios en el store:

Captura de pantalla de computadora

Descripción generada automáticamente con confianza media Al seleccionar un row en el grid actualizo el valor a True:

En el componente que quiero usar la propiedad almacenada, en este caso button-clasification.component.ts, inyecto el service en el constructor, creo una subscription (para poder después unsubscribe) y asigno el valor a una propiedad pública

Para poder usarla en el HTML.

Captura de pantalla de computadora

Descripción generada automáticamente

Captura de pantalla de computadora

Descripción generada automáticamente con confianza mediaEn el HTML uso la propiedad pública

2.- Utilizar código más elaborado siguiendo este articulo:  
 <https://dev.to/avatsaev/simple-state-management-in-angular-with-only-services-and-rxjs-41p8>

3.- Utilizar @ngrx/component-store de forma básica, solamente crear el store y grabar y recuperar los elementos.

<https://stackblitz.com/edit/angular-ivy-bv5uz6?file=src/app/component2.component.ts>

La ventaja es que usamos un component que se supone esta bien mantenido y que únicamente hay que importar.

El código esta en: <https://github.com/ngrx/platform/blob/13.0.1/modules/component-store/src/component-store.ts#L52>

Como desventaja usamos código que no es propio.

4.- Utilizar de forma completa @ngrx/component-store siguiendo este articulo:

<https://medium.com/ngconf/using-ngrx-component-store-exploration-fd9bd8354653>

El código lo tengo en:   
 [C:\Users\Usuario\Google Drive\Angular ejemplos\ComponentStore-exploration](file:///C:\Users\Usuario\Google%20Drive\Angular%20ejemplos\ComponentStore-exploration)