# Objetivo

Desarrollar un portal web para gestionar los siguients pilarres de it

* Inventario de documentos
* Inventario de activos
* Ciclo de vida de Requerimiento de Cambios, solicitud🡪test🡪implementación en prod (con envidencia capturas de pantalla)
* Ciclo de vida de Riesgos: inventarios de riesgo con calcula de impacto, exposición, etc
* Ciclo de vida de incidentes: Alta de incidents, comentarios, cierre
* Gestion de logs, Guardar envidencia de los logs de sistemas, y agregar comentarios en caso de detectar un warning o incidente
* Repositorio de documentos y aprobación, subir documentos y generar workflows de aprobación
* Inventario de backups
* Inventario de acceso físico o virtual a zonas de seguridad: Guardar una vitacora de accesos a lugares clasificados como zonas restringidas, guardando fecha y hora de accesos, y quien autorizo el mismo

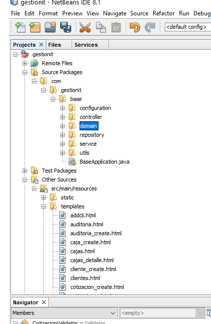
# Tecnología:

Spring boot, jpa, thymeleaf capa de presentación Bootstrap

* Spring inicial: usamos una base de datos envevida en memoria
* Spring 2: conectamos a mariaDB

# Estructura aplicación

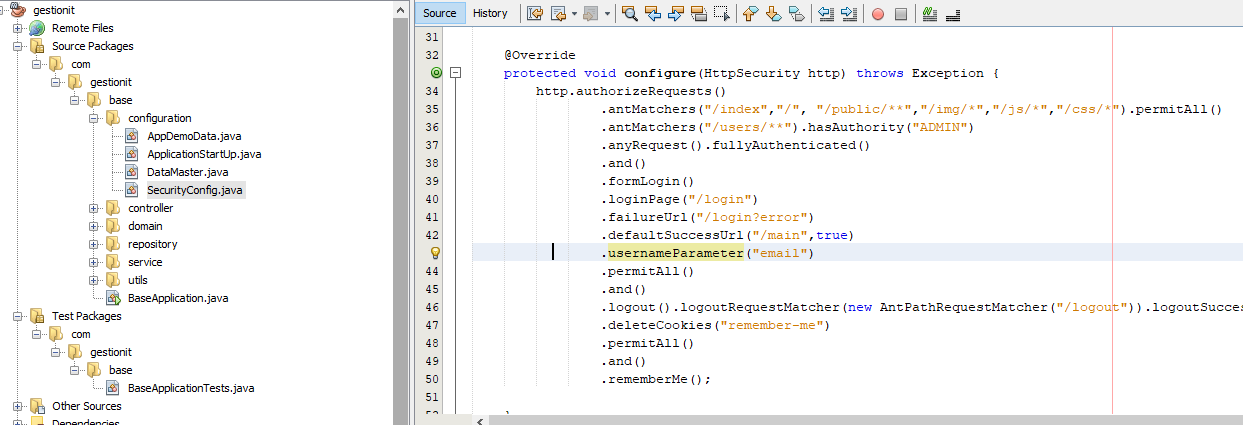
## Paquetes



Templates: Archivos html, con thymeleaf capa presentación

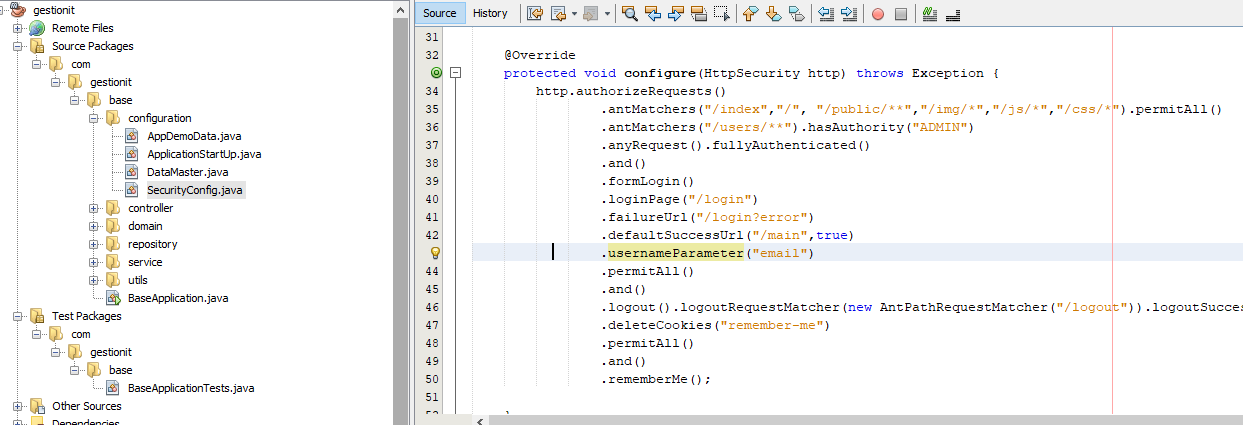
## Agregar usuarios o seguridad

En la clase security config podemos agragar usuario o manejar la seguridad



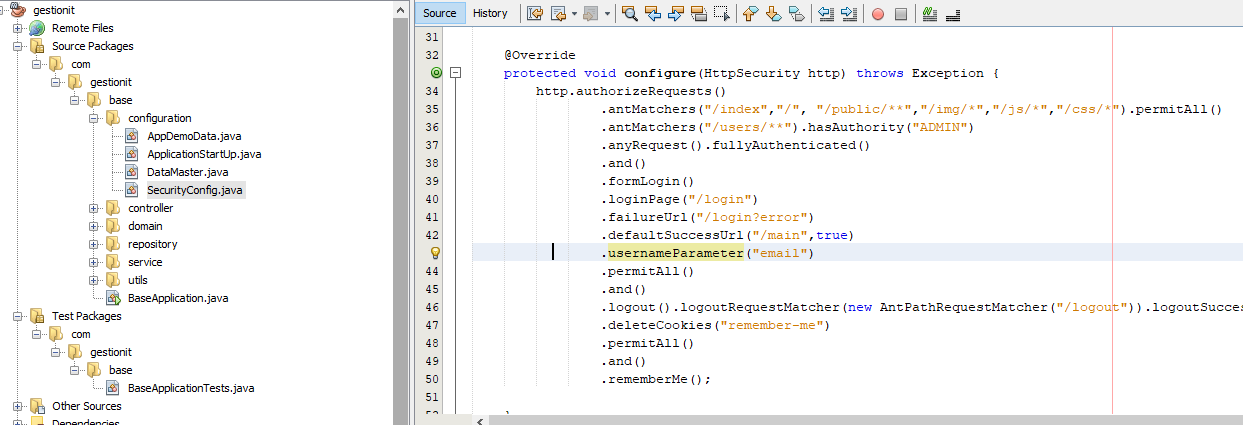
## Hacer un ABM

1-Podemos implemtar la interfaz CrudControllerInterfaz, que por defecto ya tiene mapeado los Request de un CRUD, y además 2 DTO uno para búsquedas y otro para el Objeto sobre el cual queremos hacer el CRUD



Opción 2:

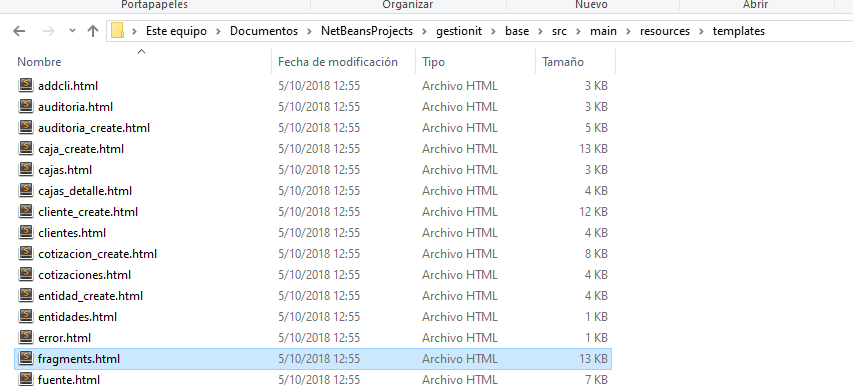
Podriamos no implemtar la interfaz y definir nuestros propios request mappings



2-Una vez que definimos el controller creamos la pagina html (el nombre tiene que ser igual al que devuelvo en el método del controller)



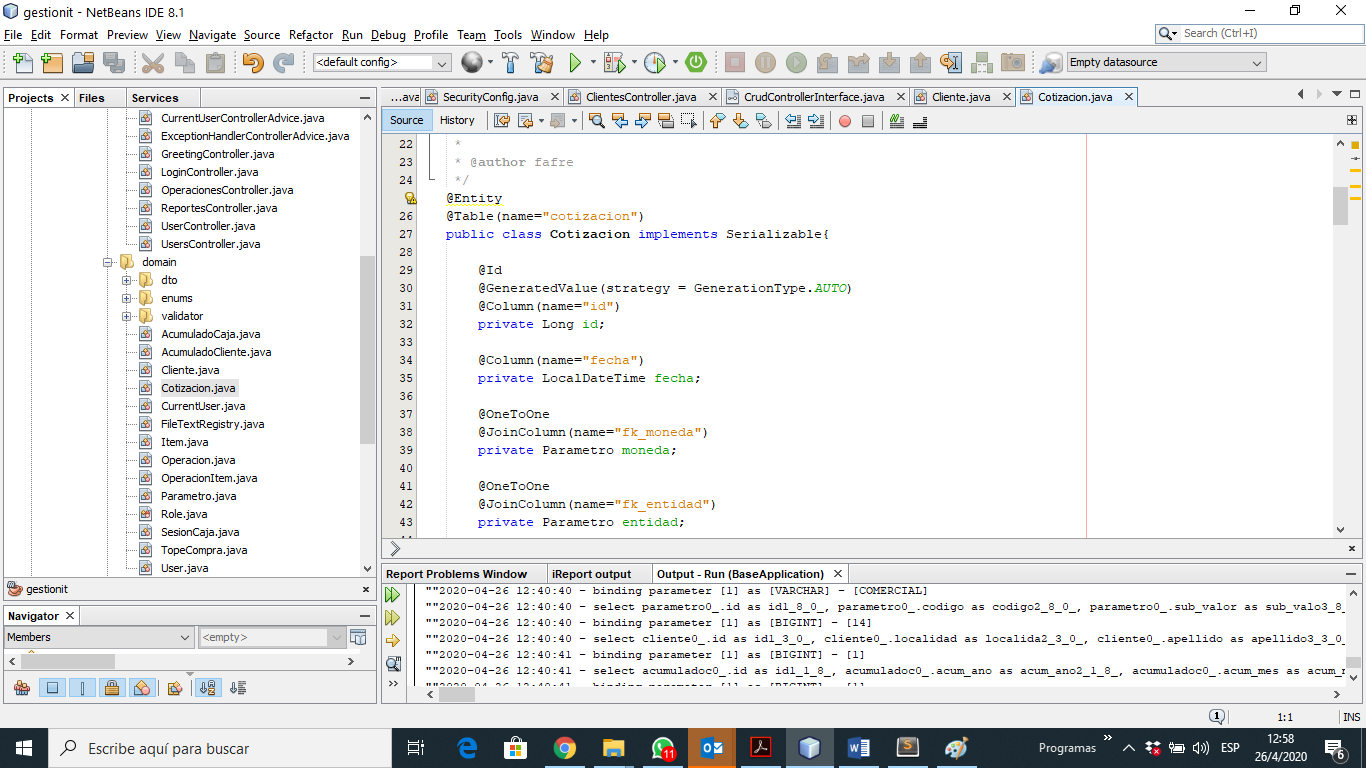
Bootstrap divide la pagina en 12 columnas, como se ve, en las primeras 3 pongo un meno lateral que sirve de búsqueda, ese menú lateral lo cree como fragmento y lo pego dentro del html llamado fragments



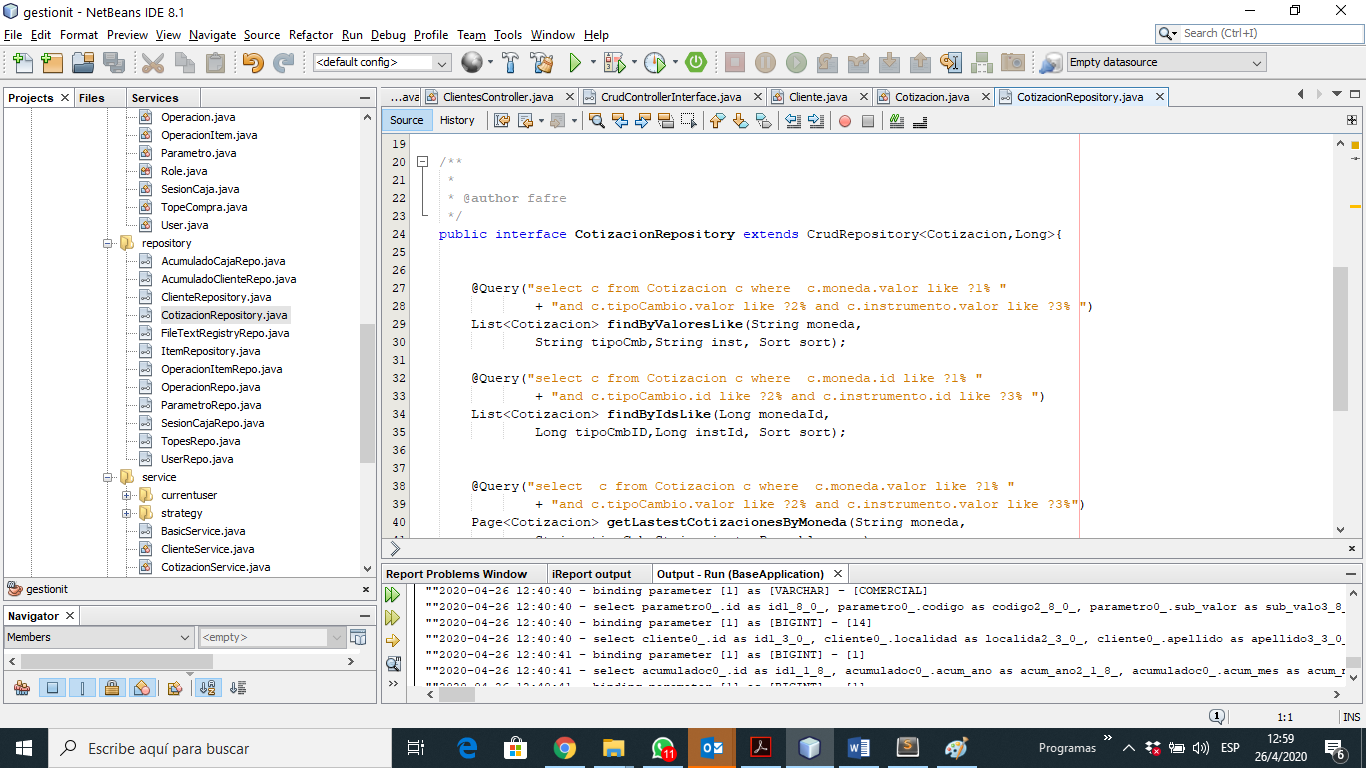
Y en las 9 columnas restantes de la derecha creo mi tabla de resultados , con un botón de nuevo encima

3-Creo la entidad y el servicio para persistirla, y el DTO para realizar búsquedas

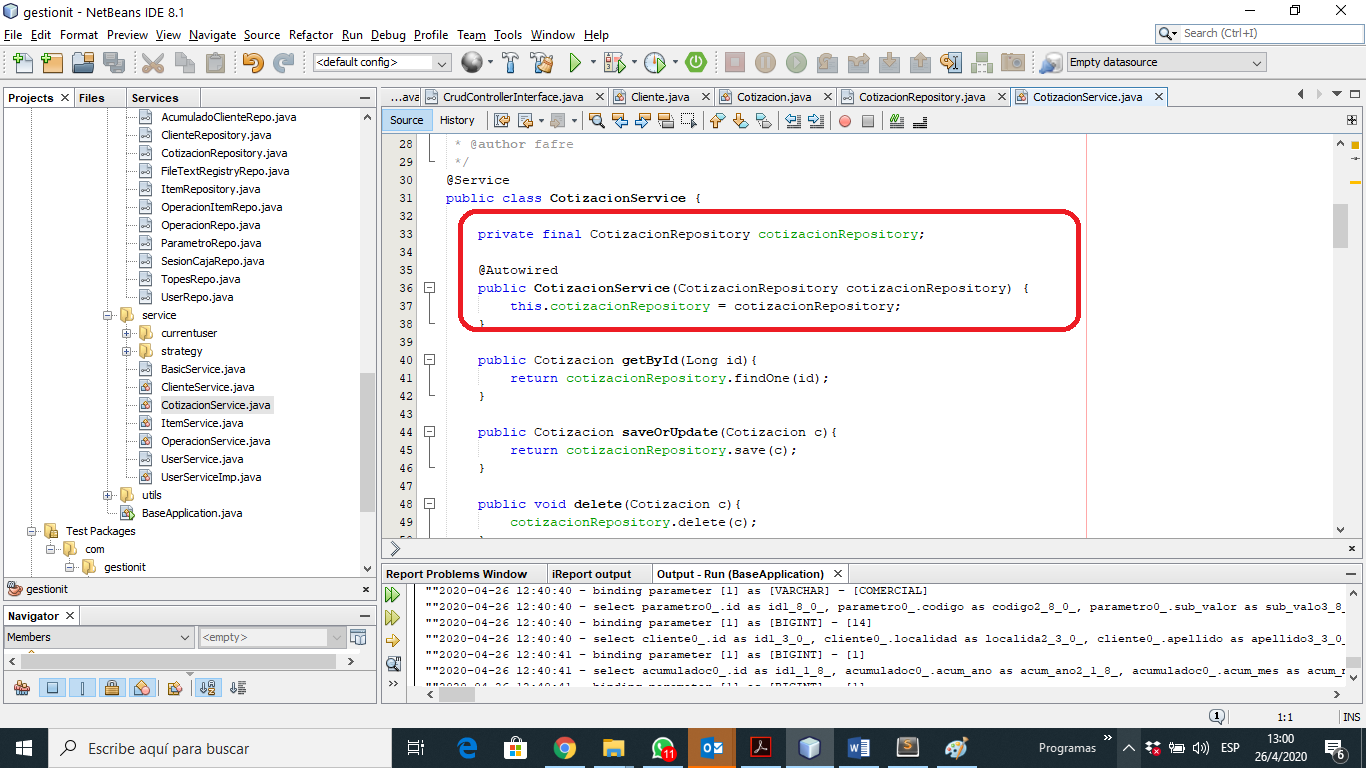
**a-Entidad**



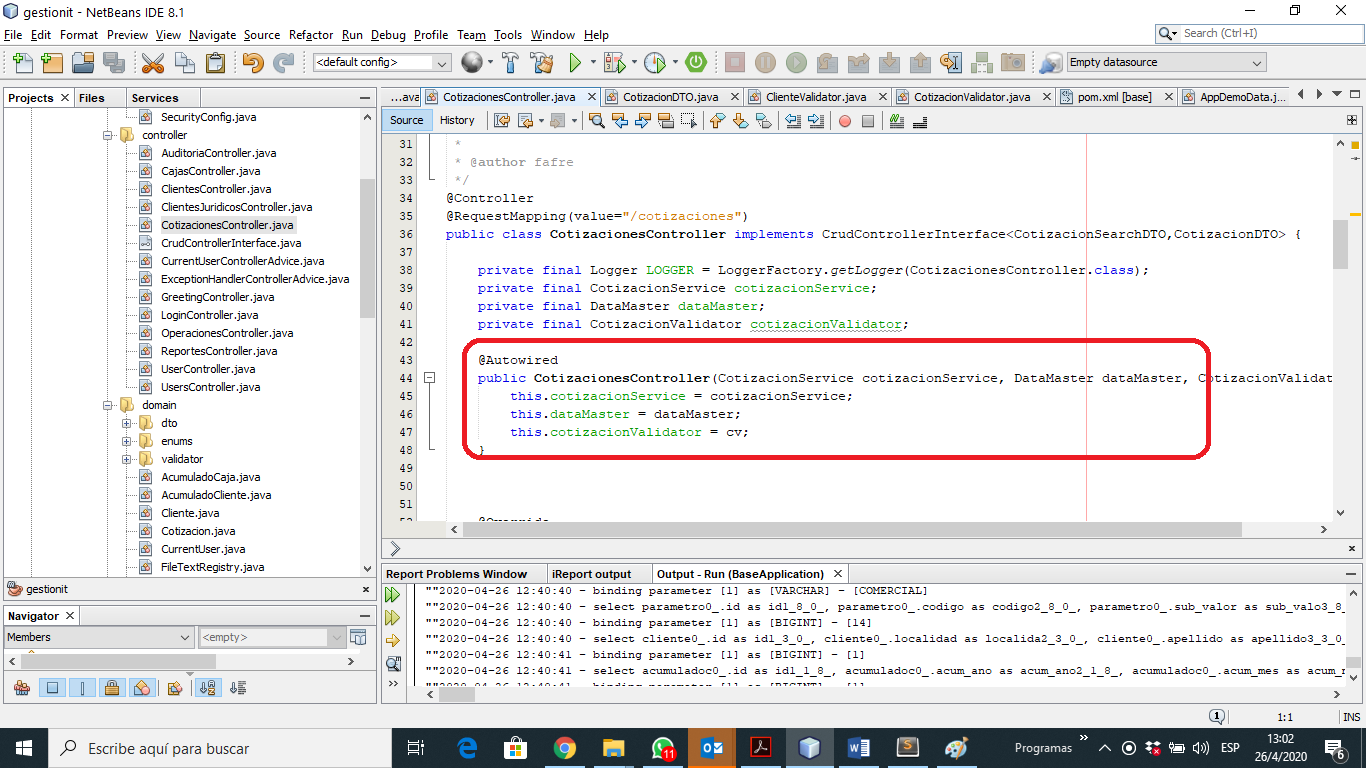
**b-Repositorio (o clase de persistencia) en este caso jpa**



**C-Servico:** Simplemten lo que hace es implementar la clase anterior de repositorio

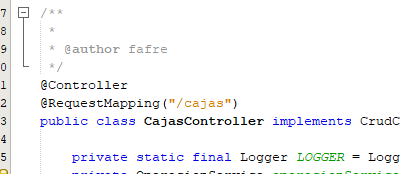


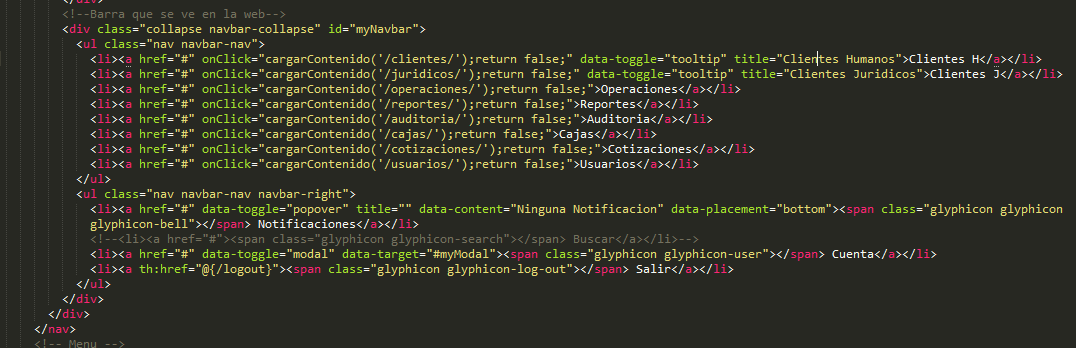
**-d-La mapeo en el controller**

****

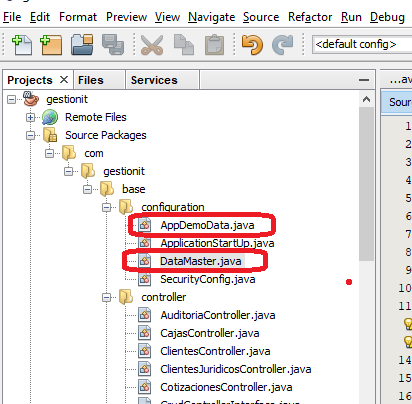
**Nota:** Realizo los mismos pasos con el objeto SearchDto (el que uso para traer las opciones de filtro de búsqueda de la pantalla)

## Agregar modificar Menu superior

Hay que ir a la pagina fragments.html y cambiar nombres, y en caso de que haya cambiado el request mapping del controller (En este caso “/cajas”)



## Configuración de inicio app



**AppDemoData:** ahí cargo información para testear la aplicación, como aun no hay base de datos poque se usa la embebida en memoria, cargo clientes de prueba etc

**DataMaster:** Esta es una clase singleton, que tiene información que se usa en todas las pantallas, como por ejemplo provincias, tipos de algo etc, y que por ahora se cargan por base de datos es decir, no hay desarrollado abms para cargarlo, entonces no pueden cambiar solo si se realiza un update en la base, por eso lo que hago es cargarlas en memoria apenas arranca la aplicación para que sea rápido acceder