**SEjercicios de graphql. 2 pto**

Haz un servidor de graphql con una entidad alumno, con nombre, id, y un listado de asignaturas con sus notas.

Define una query de get de todos los alumnos Define una mutación para añadir alumnos sin notas.

En postman o en http de intellij haz una petición a los alumnos con todas sus asignaturas, y otra solo para los nombres de los alumnos

Haz la petición de mutación de añadir alumnos.

**Ejercicio 2 (2 Pto)**

Una empresa de detectives tiene casos en los que trabajan dos detectives, uno por la mañana y otro por la noche.

El de por la mañana genera informes que el de por la noche contrasta.

La empresa solo tiene un ordenador para todos los empleados y tiene que inventar un mecanismo para que los informes del detective de la mañana solo los pueda ver el de la tarde.

Los detectives no se conocen solo saben que uno trabaja por la mañana y otro por la noche.

Inventa un mecanismo para que los informes solo los puedan ver ellos, y el de por la noche este completamente seguro de que el informe lo ha hecho el de la mañana.

Haz un main para guardar informe y otro para leer informe.

**Ejercicio 3 (6 puntos)**

Haz un **rest** con **spring**. Con las siguientes características.

* Se tendrán que **guardar** los **usuarios** con sus **contraseñas** y **roles** para que puedan hacer login.
* Las entidades a devolver serán informes, con nombre , fecha del informe, role de seguridad que puede verlo y ratones con nombre y edad.
* Los **roles** posibles para acceder a los informes serán
  + **NIVEL1**,
  + **NIVEL2** que tiene acceso al nivel 1 y al suyo
  + **ADMIN** que tendría acceso a todo.

Peticiones:

* Petición de **login** para validar al usuario. Las siguientes peticiones se validarán con un **token** de JWT **que caducara a los 10 segundos**.
* Petición de **ver todos** los ratones posibles (puedes tener cualquier role).
* Petición de **añadir informe** que **solo** lo podrá hacer el rol **ADMIN**.
* Petición de **añadir ratones** que **solo** lo podrá hacer el rol **ADMIN**.
* Petición de **get informes** donde según el role se le dará los informes correspondientes. Los informes tienen dos tipos de roles NIVEL1, y NIVEL2, si alguien tiene el role NIVEL2 tb tiene permiso para los de NIVEL1. Se filtrarán los informes según el nivel del usuario.
* Petición de **get** de un **informe** concreto **por id**, filtrado por el role del usuario que lo pida.
* Petición para comprobar el mecanismo **cuando el token expira**, que será simplemente que **avisará** al usuario de token expirado **con un 401**.
* Todas estas peticiones se guardan en una colección de postman o http de intellij para cada petición a realizar.

Entidades:

**Usuario:**

* ID (identificador único)
* Nombre de usuario
* Contraseña encriptada
* Roles asignados (puede ser una lista de roles)

**Informe:**

* ID (identificador único)
* Nombre del informe
* Fecha del informe
* Rol de seguridad que puede acceder (puede ser un único rol o una lista de roles)
* Lista de ratones asociados (puede ser una lista de objetos Ratón)

**Ratón:**

* ID (identificador único)
* Nombre del ratón
* Edad del ratón