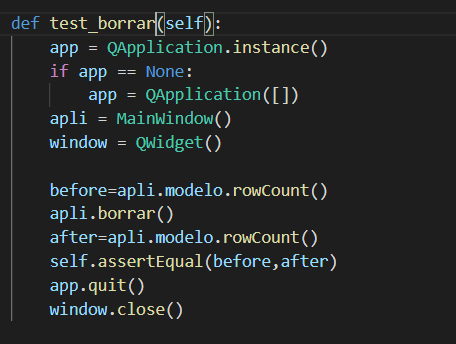
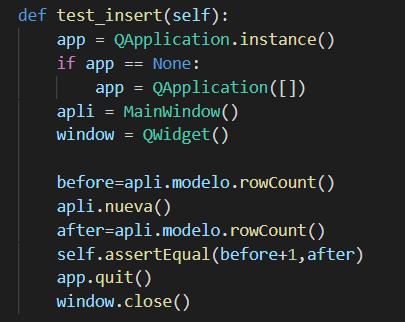
# Pruebas unitarias

Test unitarios realizados sobre las operaciones básicas de insertado y borrado, además también sobre la función generadora de informes, comprobando si ha generado dicho informe posterior a su ejecución



# pruebas de integración de los distintos elementos

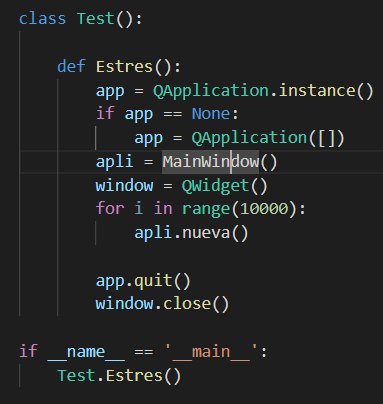
Luego de añadir El Boton Animado a nuestra aplicación, sus test unitarios seguirían funcionando igual que antes

# pruebas de regresión

Las pruebas unitarias realizadas antes seguirían teniendo el mismo resultado luego de añadir el botón

# pruebas de volumen y estrés

Insertaríamos 10.000 nuevas tuplas para comprobar su correcto funcionamiento en situaciones de estrés



# pruebas de seguridad

A la hora de hacer la prueba de Seguridad hemos observado que nuestra base de datos no esta cifrada, ni tiene protección de ningún tipo, y los datos que guardamos son un tanto sensibles ya que contiene información personal de los alumnos.

Para solucionar esto he propuesto utilizar la herramienta de Python **pysqlitecipher** que nos permite crear un objeto sobre el que realizaremos las operaciones actuando como modelo, y a su vez cifraría la base de datos.

Como implementar esta herramienta:

Instalamos su libreria

-pip install pysqlitecipher==0.11

La importamos en nuestro proyecto

-from pysqlitecipher import sqlitewrapper

Creamos el objeto/modelo indicándole la ruta de la DB y su password

-obj = sqlitewrapper.SqliteCipher(dataBasePath="pysqlitecipher.db" , checkSameThread=False , password=None)

Sobre el objeto:

Para crear nuevas tablas

-obj.createTable(tableName , colList , makeSecure=False ,commit=True)

En la que colList estaría compuesta por

colList = [

[colname , datatype] ,  
[colname2 , datatype] ,

]

Insertar Datos

-obj.insertIntoTable(tableName , insertList , commit = True)

Insertlist deberá tener el mismo formato que colList pero con sus respectivos datos ya rellenos tal que:

colList = [

[colname , colname2]

]

Para obtener datos de una tabla:

- obj.getDataFromTable(tableName , raiseConversionError = True , omitID = False)

Para borrar datos de una tabla:

obj.deleteDataInTable(tableName , iDValue , commit = True , raiseError = True , updateId = True)

Para actualizarla:

obj.updateInTable(tableName , iDValue , colName , colValue , commit = True , raiseError = True)

# pruebas de uso de recursos por parte de la aplicación

Los recursos utilizados por la aplicación cuando se esta ejecutando son mínimos y tendrían poco impacto en el sistema

