

Estudio Independiente #6

Nombre: Manolo Sebastián Iñiguez Ramírez

Código: 00212562

Fecha: 27 noviembre 2022

1. Resumir con sus palabras el nivel de aislamiento de las transacciones (transaction isolation level).

Se define niveles de aislamiento de transacciones para lidiar con los posibles problemas de lectura de una base de datos, con esto se restringe y se controla, por ejemplo, posibles problemas:

Una "Dirty read" que puede procesar información incorrecta o una "Nonrepeatable read" que encuentra modificaciones en la relectura, o una "Phantom read" que ocurre en una relectura y se encuentra nuevas filas que fueron insertadas desde la última lectura.

Para controlar, la base de datos creará los locks necesarios para lograr el nivel de aislamiento deseado por el programador ya que el nivel es libre, sin embargo, cuanto más restrictivo sea el nivel, menor será el rendimiento.

Los niveles de aislamiento de transacciones son:

Un "Read-uncommitted isolation level" que permite que ocurran todos los posibles problemas mencionados. (Dirty read, Nonrepeatable read, Phantom read)

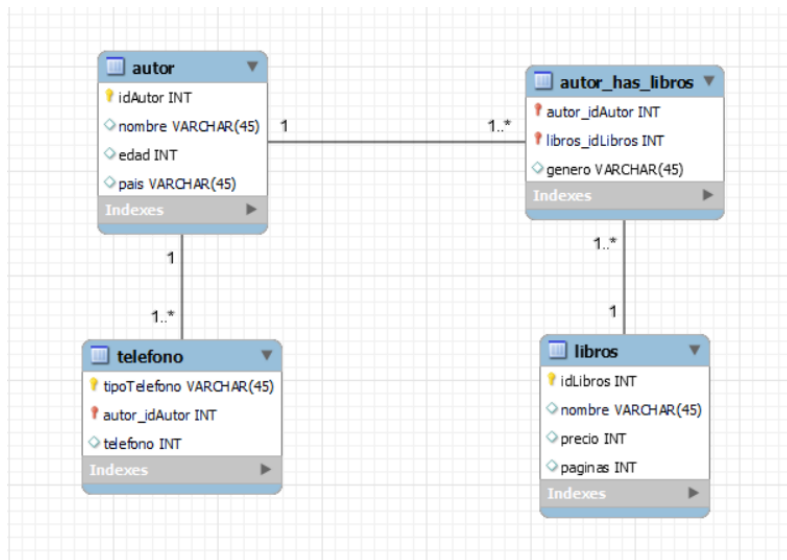
Un "Read-committed isolation level" permite que ocurran las Nonrepeatable y Phantom reads pero no permite las Dirty reads.

Un "Repeatable-read isolation level" permite que ocurran Phantom reads pero no permite las otras dos.

Un "Serializable isolation read" no permite que ocurra ninguno de los problemas de lecturas de datos.

2. Implementar un ejemplo de aplicación de varias capas (ver Tutorial Mysql en la web) y presentar el código de conexión e interacción con la base de datos.

Base de Datos MySQL:



Autor:

	idAutor	nombre	edad	pais
▶	1	Manolo	23	Ecuador
	2	Julio	40	Colombia
	3	Pedro	89	Portugal
	4	Ana	24	Mexico
	5	Maria	36	USA
	6	Noel	30	Ecuador
*	NULL	NULL	NULL	NULL

Libros:

	idLibros	nombre	precio	paginas
▶	111	Harry Potter	87	788
	222	El señor de los anillos	37	345
	333	El alquimista	34	123
	666	Crepusculo	27	264
	777	Libro 34	97	734
	999	Libro 4	54	112
	1000	Libro 12	24	843
	1111	LibroX	99	120
	1222	LibroY	26	872
	1333	LibroW	12	45
*	NULL	NULL	NULL	NULL

Libros_has_autores:

	autor_idAutor	libros_idLibros	genero
▶	1	111	Fantasia
	1	222	Fantasia
	1	777	Romance
	2	333	Romance
	3	999	Romance
	4	222	Fantasia
	4	1000	Fantasia
	5	111	Fantasia
	5	666	Misterio
*	NULL	NULL	NULL

Teléfono:

	tipoTelefono	autor_idAutor	telefono
▶	casa	1	2111
	casa	2	2222
	celular	1	9111
	celular	2	9555
	celular	3	9222
	celular	4	9333
	celular	5	9444
	trabajo	1	3111
	trabajo	5	3222
*	NULL	NULL	NULL

Interfaz Gráfica y Código de conexión con la base de datos:

La interfaz fue desarrollada con QtDesigner y la conexión en lenguaje Python.

Se instaló lo apropiado para que funcione la conexión entre el Visual Studio y el MySQL.

Y se crea la función para conectar ambos.

Se usa la importación: `from mysql.connector import MySQLConnection, Error`

Y la conexión es con: `conn = MySQLConnection(**db_config)`

```
def connect():
    """ Connect to MySQL database """

    db_config = read_db_config()
    conn = None
    try:
        print('Connecting to MySQL database...')
        conn = MySQLConnection(**db_config)

        if conn.is_connected():
            print('Connection established.')
        else:
            print('Connection failed.')

    except Error as error:
        print(error)

    finally:
        if conn is not None and conn.is_connected():
            conn.close()
            print('Connection closed.')
```

Donde read_db_config() es una función importada:

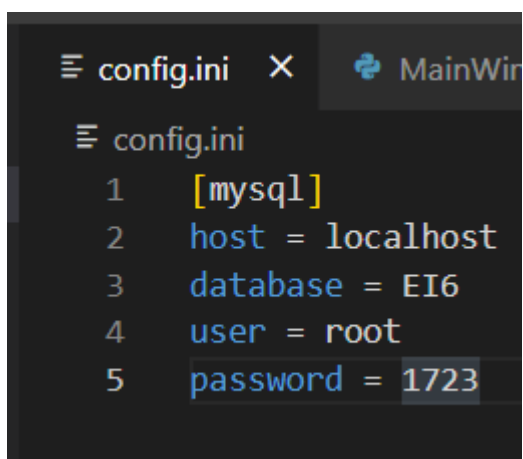
```
def read_db_config(filename='config.ini', section='mysql'):

    # create parser and read ini configuration file
    parser = ConfigParser()
    parser.read(filename)

    # get section, default to mysql
    db = {}
    if parser.has_section(section):
        items = parser.items(section)
        for item in items:
            db[item[0]] = item[1]
        #print(db)
    else:
        raise Exception('{0} not found in the {1} file'.format(section, filename))

    return db
```

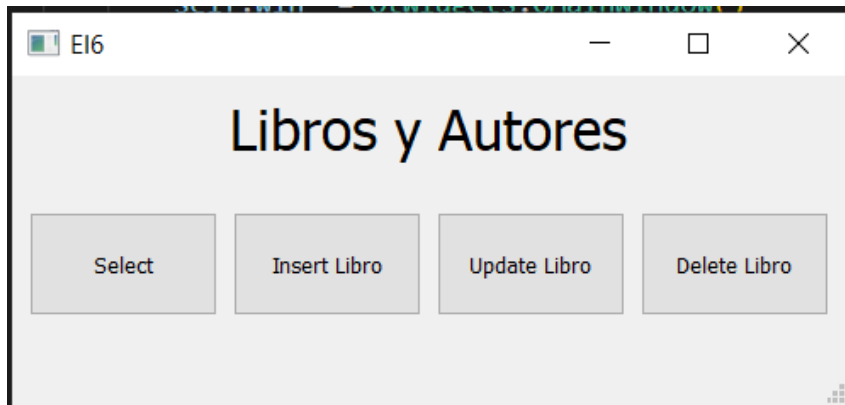
Esta permite leer el archivo de configuración que contiene la información de la base de datos a la que nos conectaremos y devuelve un objeto de diccionario.



```
config.ini
1  [mysql]
2  host = localhost
3  database = EI6
4  user = root
5  password = 1723
```

Archivo de configuración.

Main Window:



Esta es la ventana principal (Main Window)

Creacion de la interfaz de esta ventana, tiene labels y botones, tambien las conexiones para que los botones se conecten a las funciones que llevan a las otras ventanas.

```
def setupUi(self, MainWindow):
    MainWindow.setObjectName("MainWindow")
    MainWindow.resize(493, 194)

    self.centralwidget = QtWidgets.QWidget(MainWindow)
    self.centralwidget.setObjectName("centralwidget")

    self.texto1 = QtWidgets.QLabel(self.centralwidget)
    self.texto1.setGeometry(QtCore.QRect(0, 10, 491, 41))
    font = QtGui.QFont()
    font.setPointSize(20)
    self.texto1.setFont(font)
    self.texto1.setScaledContents(False)
    self.texto1.setAlignment(QtCore.Qt.AlignCenter)
    self.texto1.setWordWrap(True)
    self.texto1.setObjectName("texto1")

    self.botonInsert = QtWidgets.QPushButton(self.centralwidget)
    self.botonInsert.setGeometry(QtCore.QRect(130, 80, 111, 61))
    self.botonInsert.setObjectName("botonInsert")
    self.botonInsert.clicked.connect(self.AbrirInsertWindow)

    self.botonUpdate = QtWidgets.QPushButton(self.centralwidget)
    self.botonUpdate.setGeometry(QtCore.QRect(250, 80, 111, 61))
    self.botonUpdate.setObjectName("botonUpdate")
    self.botonUpdate.clicked.connect(self.AbrirUpdateWindow)

    self.botonSelect = QtWidgets.QPushButton(self.centralwidget)
    self.botonSelect.setGeometry(QtCore.QRect(10, 80, 111, 61))
    self.botonSelect.setObjectName("botonSelect")
    self.botonSelect.clicked.connect(self.AbrirSelectWindow)

    self.botonDelete = QtWidgets.QPushButton(self.centralwidget)
    self.botonDelete.setGeometry(QtCore.QRect(370, 80, 111, 61))
    self.botonDelete.setObjectName("botonDelete")
    self.botonDelete.clicked.connect(self.AbrirDeleteWindow)

    MainWindow.setCentralWidget(self.centralwidget)

    self.menubar = QtWidgets.QMenuBar(MainWindow)
```

Las funciones para abrir las otras ventanas desde la ventana principal y posteriormente se importa todas las ventanas.

```
def AbrirSelectWindow(self):
    self.win = QtWidgets.QMainWindow()
    self.ui = Ui_SelectWindow()
    self.ui.setupUi(self.win)
    self.win.show()

def AbrirInsertWindow(self):
    self.win = QtWidgets.QMainWindow()
    self.ui = Ui_InsertWindow()
    self.ui.setupUi(self.win)
    self.win.show()

def AbrirUpdateWindow(self):
    self.win = QtWidgets.QMainWindow()
    self.ui = Ui_UpdateWindow()
    self.ui.setupUi(self.win)
    self.win.show()

def AbrirDeleteWindow(self):
    self.win = QtWidgets.QMainWindow()
    self.ui = Ui_DeleteWindow()
    self.ui.setupUi(self.win)
    self.win.show()
```

```
from PyQt5 import QtCore, QtGui, QtWidgets
from mysql.connector import MySQLConnection, Error
from python_mysql_dbconfig import read_db_config
from SelectWindow import Ui_SelectWindow
from InsertWindow import Ui_InsertWindow
from UpdateWindow import Ui_UpdateWindow
from DeleteWindow import Ui_DeleteWindow
```

El main donde se inicia todo y se enlaza la conexión con la base de datos

```
if __name__ == "__main__":
    import sys
    app = QtWidgets.QApplication(sys.argv)
    MainWindow = QtWidgets.QMainWindow()
    ui = Ui_MainWindow()
    ui.setupUi(MainWindow)
    MainWindow.show()
    Ui_MainWindow.connect()
    sys.exit(app.exec_())
```

La ventana principal tiene botones para:

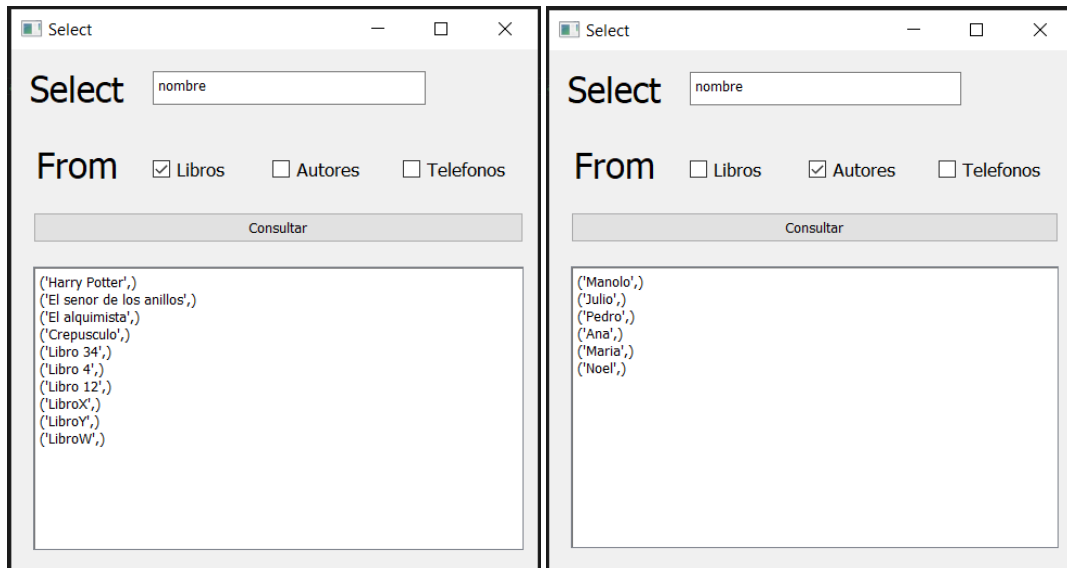
- Select →
Permite realizar una búsqueda cómoda en la base de datos, con un EditText para ingresar el select y CheckBoxes para escoger de la tabla que deseamos la consulta.
- Insert Libro →
Pide todos los parámetros necesarios para ingresar un libro

(Hice solo de libros para no alargar el deber)

- Update Libro →
Permite actualizar cualquier libro por medio del Id del libro
- Delete Libro →
Permite eliminar cualquier libro por medio del Id del libro

Select Window:

Se escribe que se desea consultar (id, nombres, precios, paginas, teléfonos, etc) y se marca el checkbox de que tabla, en la pantalla de abajo sale el resultado.



Clase UI_SelectWindow →

Creacion de la interfaz de esta ventana, tiene labels, botones, checkboxes.

No sale toda la función en la captura, pero es similar lo que no sale, es interfaz gráfica y conexión de checkboxes y botón a las funciones que les darán funcionalidad.

```

def setupUi(self, SelectWindow):
    SelectWindow.setObjectName("SelectWindow")
    SelectWindow.resize(484, 481)

    self.label = QtWidgets.QLabel(SelectWindow)
    self.label.setGeometry(QtCore.QRect(0, 10, 121, 51))
    font = QtGui.QFont()
    font.setPointSize(20)
    self.label.setFont(font)
    self.label.setLayoutDirection(QtCore.Qt.LeftToRight)
    self.label.setAlignment(QtCore.Qt.AlignCenter)
    self.label.setObjectName("label")

    self.label_2 = QtWidgets.QLabel(SelectWindow)
    self.label_2.setGeometry(QtCore.QRect(0, 80, 121, 51))
    font = QtGui.QFont()
    font.setPointSize(20)
    self.label_2.setFont(font)
    self.label_2.setLayoutDirection(QtCore.Qt.LeftToRight)
    self.label_2.setAlignment(QtCore.Qt.AlignCenter)
    self.label_2.setObjectName("label_2")

    self.textEdit = QtWidgets.QTextEdit(SelectWindow)
    self.textEdit.setGeometry(QtCore.QRect(130, 20, 251, 31))
    self.textEdit.setObjectName("textEdit")

    self.checkBoxAutores = QtWidgets.QCheckBox(SelectWindow)
    self.checkBoxAutores.setGeometry(QtCore.QRect(240, 100, 81, 20))
    font = QtGui.QFont()
    font.setPointSize(10)
    self.checkBoxAutores.setFont(font)
    self.checkBoxAutores.setObjectName("checkBoxAutores")
    self.checkBoxAutores.stateChanged.connect(self.cambioCheck)

    self.checkBoxLibros = QtWidgets.QCheckBox(SelectWindow)
    self.checkBoxLibros.setGeometry(QtCore.QRect(130, 100, 81, 20))
    font = QtGui.QFont()
    font.setPointSize(10)
    self.checkBoxLibros.setFont(font)
    self.checkBoxLibros.setObjectName("checkBoxLibros")
    self.checkBoxLibros.stateChanged.connect(self.cambioCheck)

```

Cuando se marca un checkbox, se asigna el nombre de la tabla a la variable tabla para realizar la consulta con ese string.

```

def cambioCheck(self):
    tabla = ""
    if self.checkBoxAutores.isChecked()==True:
        tabla = "autor"
    if self.checkBoxLibros.isChecked()==True:
        tabla = "libros"
    if self.checkBoxTelefonos.isChecked()==True:
        tabla = "telefono"
    return tabla

```

Una vez que se presiona el botón para consultar, se asignan las variables y se realiza la consulta en la base de datos.

Se hace un fetchall() para la obtención de todas las tuplas de las tabla.

Si la base de datos es mas grande se recomienda usar fetchmany().

Se ejecuta el cursor con dicha consulta.

Este proceso es parecido para toda el programa.

```
def ClickConsultar(self):
    columna = str(self.textEdit.toPlainText())
    tabla = self.cambioCheck()

    try:
        dbconfig = read_db_config()
        conn = MySQLConnection(**dbconfig)
        cursor = conn.cursor()
        cursor.execute("SELECT " + columna + " FROM " + tabla)
        rows = cursor.fetchall() #guarda todas las filas como un arreglo

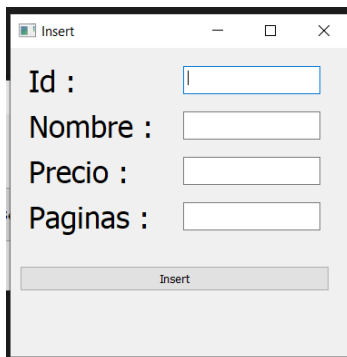
        print('Total Row(s):', cursor.rowcount)
        for row in rows:
            self.listaSelect.setText('\n'.join(map(str, rows)))

    except Error as e:
        print(e)

    finally:
        cursor.close()
        conn.close()
```

Insert Window:

Al insertar un libro sale una confirmación de que el libro fue insertado y se puede hacer una consulta en Select para ver el resultado.

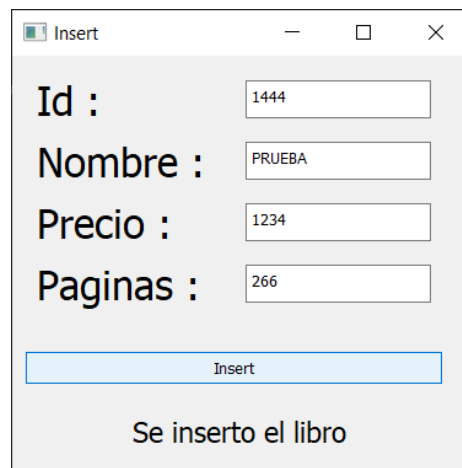


Id :

Nombre :

Precio :

Paginas :



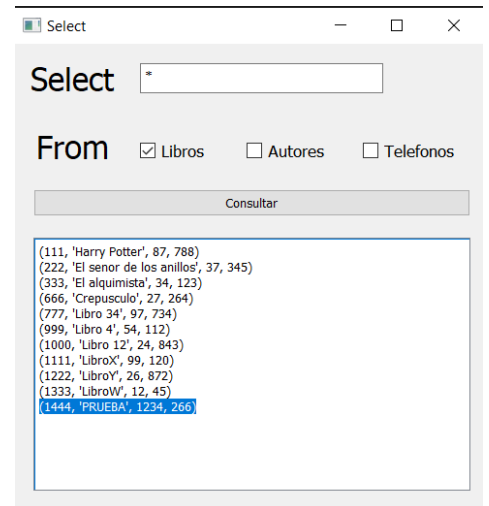
Id :

Nombre :

Precio :

Paginas :

Se inserto el libro



Select

From ☒ Libros ☐ Autores ☐ Telefonos

(111, 'Harry Potter', 87, 788)
(222, 'El señor de los anillos', 37, 345)
(333, 'El alquimista', 34, 123)
(666, 'Crepusculo', 27, 264)
(777, 'Libro 34', 97, 734)
(999, 'Libro 4', 54, 112)
(1000, 'Libro 12', 24, 843)
(1111, 'LibroX', 99, 120)
(1222, 'LibroY', 26, 872)
(1333, 'LibroW', 12, 45)
(1444, 'PRUEBA', 1234, 266)

Clase UI_InsertWindow →

Creacion de la interfaz de esta ventana, tiene labels, EditTexts y un botón.

No sale toda la función en la captura, pero es similar lo que no sale, es interfaz gráfica junto con las conexiones de los recuadros de texto y botón a las funciones que les darán funcionalidad.

```

def setupUi(self, InsertWindow):

    InsertWindow.setObjectName("InsertWindow")
    InsertWindow.resize(375, 340)

    self.textEditInsertId = QtWidgets.QTextEdit(InsertWindow)
    self.textEditInsertId.setGeometry(QtCore.QRect(190, 20, 151, 31))
    self.textEditInsertId.setObjectName("textEditInsertId")

    self.label = QtWidgets.QLabel(InsertWindow)
    self.label.setGeometry(QtCore.QRect(20, 10, 71, 51))
    font = QtGui.QFont()
    font.setPointSize(20)
    self.label.setFont(font)
    self.label.setLayoutDirection(QtCore.Qt.LeftToRight)
    self.label.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
    self.label.setObjectName("label")

    self.textEditInsertNombre = QtWidgets.QTextEdit(InsertWindow)
    self.textEditInsertNombre.setGeometry(QtCore.QRect(190, 70, 151, 31))
    self.textEditInsertNombre.setObjectName("textEditInsertNombre")

    self.label_2 = QtWidgets.QLabel(InsertWindow)
    self.label_2.setGeometry(QtCore.QRect(20, 60, 161, 51))
    font = QtGui.QFont()
    font.setPointSize(20)
    self.label_2.setFont(font)
    self.label_2.setLayoutDirection(QtCore.Qt.LeftToRight)
    self.label_2.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
    self.label_2.setObjectName("label_2")

    self.textEditInsertPrecio = QtWidgets.QTextEdit(InsertWindow)
    self.textEditInsertPrecio.setGeometry(QtCore.QRect(190, 120, 151, 31))
    self.textEditInsertPrecio.setObjectName("textEditInsertPrecio")

    self.label_3 = QtWidgets.QLabel(InsertWindow)
    self.label_3.setGeometry(QtCore.QRect(20, 110, 161, 51))
    font = QtGui.QFont()
    font.setPointSize(20)
    self.label_3.setFont(font)
    self.label_3.setLayoutDirection(QtCore.Qt.LeftToRight)
    self.label_3.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)

```

Al hacer click en el botón para insertar, se asigna los valores de los EditTexts (los recuadros de texto llenos de información) y con esto se hace la consulta.

Igualmente sale el mensaje de confirmación después de insertar

```
def clickInsertar(self):
    id = int(self.textEditInsertId.toPlainText())
    nombre = str(self.textEditInsertNombre.toPlainText())
    precio = int(self.textEditInsertPrecio.toPlainText())
    paginas = int(self.textEditInsertPaginas.toPlainText())

    query = "INSERT INTO libros(idLibros, nombre, precio, paginas) " \
           "VALUES(%s,%s,%s,%s)"
    args = (id, nombre, precio, paginas)

    try:
        db_config = read_db_config()
        conn = MySQLConnection(**db_config)

        cursor = conn.cursor()
        cursor.execute(query, args)
        #cursor.executemany(arreglo) #Para insertar varias cosas al mismo tiempo

        self.textoActualizacionInsert.setText("Se inserto el libro")

        if cursor.lastrowid:
            print('last insert id', cursor.lastrowid)
        else:
            print('last insert id not found')

        conn.commit()
    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()
```

Update Window:

Al actualizar un libro sale una confirmación de que el libro fue actualizado y se puede hacer una consulta en Select para ver el resultado.

Update window with empty input fields:

- Id :
- Nombre :
- Precio :
- Paginas :
- Update button

Update window with filled input fields and confirmation:

- Id :
- Nombre :
- Precio :
- Paginas :
- Update button
- Se actualizo el libro

Select window showing a list of books:

Select:

From: ☒ Libros ☐ Autores ☐ Telefonos

Consultar button

List of books (updated record highlighted):

- (111, 'Harry Potter', 87, 788)
- (222, 'El señor de los anillos', 37, 345)
- (333, 'El alquimista', 34, 123)
- (666, 'Crepusculo', 27, 264)
- (777, 'Libro 34', 97, 734)
- (999, 'Libro 4', 54, 112)
- (1000, 'Libro 12', 24, 843)
- (1111, 'LibroX', 99, 120)
- (1222, 'LibroY', 26, 872)
- (1333, 'LibroW', 12, 45)
- (1444, 'Prueba', 67, 98)**

Clase UI_UpdateWindow →

La ventana de Update tiene la misma interfaz de Insert, así mismo las mismas conexiones de los elementos de la interfaz con las funciones.

Lo que cambia es la consulta, se actualiza el nombre, precio y páginas del ID del libro que se inserte. Y asimismo sale el mensaje de confirmación.

```
def clickUpdate(self):
    id = int(self.textEditUpdateId.toPlainText())
    nombre = str(self.textEditUpdateNombre.toPlainText())
    precio = int(self.textEditUpdatePrecio.toPlainText())
    paginas = int(self.textEditUpdatePaginas.toPlainText())

    query = """UPDATE libros
                SET nombre = %s,
                    precio = %s,
                    paginas = %s,
                WHERE idLibros = %s """
    args = (nombre, precio, paginas, id)

    try:
        db_config = read_db_config()
        conn = MySQLConnection(**db_config)

        cursor = conn.cursor()
        cursor.execute(query, args)
        #cursor.executemany(arreglo) #Para insertar varias cosas al mismo tiempo

        self.textoActualizacionUpdate.setText("Se actualizo el libro")

        if cursor.lastrowid:
            print('last insert id', cursor.lastrowid)
        else:
            print('last insert id not found')

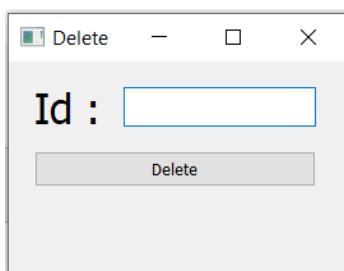
        conn.commit()
    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()
```

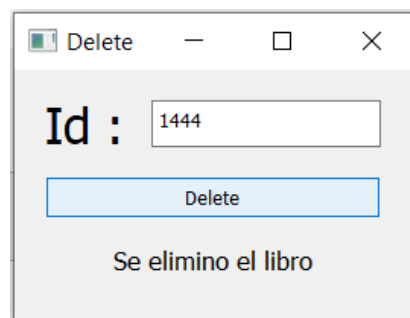
Delete Window:

Al eliminar un libro sale una confirmación de que el libro fue eliminado y se puede hacer una consulta en Select para ver el resultado.

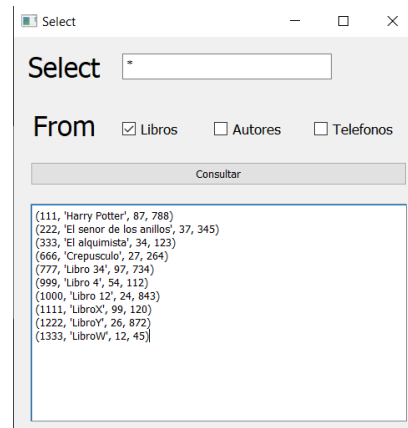
Consiste en ingresar el Id del libro que se quiere eliminar y presionar el botón.



A screenshot of a window titled "Delete". It contains a label "Id :" followed by an empty text input field. Below the input field is a button labeled "Delete".



A screenshot of the "Delete" window after a successful deletion. The "Id :" label is followed by a text input field containing the value "1444". Below the input field is a button labeled "Delete". At the bottom of the window, the text "Se elimino el libro" is displayed.



A screenshot of a window titled "Select". It has a search bar at the top. Below it, there are checkboxes for "Libros" (checked), "Autores", and "Telefonos". A "Consultar" button is located below these checkboxes. The main area of the window displays a list of books in a text box, each entry containing an ID, the book title, and its page count in parentheses. The list includes: (111, 'Harry Potter', 87, 788), (222, 'El señor de los anillos', 37, 345), (333, 'El alquimista', 34, 123), (666, 'Crepusculo', 27, 264), (777, 'Libro 34', 97, 734), (999, 'Libro 4', 54, 112), (1000, 'Libro 12', 24, 843), (1111, 'LibroX', 99, 120), (1222, 'LibroY', 26, 872), and (1333, 'LibroW', 12, 45).

Clase UI_DeleteWindow →

Creacion de la interfaz de esta ventana, tiene labels y un boton, también la conexión del recuadro de texto y botón a las funciones que les darán funcionalidad.

```
def setupUi(self, DeleteWindow):
    DeleteWindow.setObjectName("DeleteWindow")
    DeleteWindow.resize(265, 166)

    self.textEditId = QtWidgets.QTextEdit(DeleteWindow)
    self.textEditId.setGeometry(QtCore.QRect(90, 20, 151, 31))
    self.textEditId.setObjectName("textEditId")

    self.label = QtWidgets.QLabel(DeleteWindow)
    self.label.setGeometry(QtCore.QRect(20, 10, 71, 51))
    font = QtGui.QFont()
    font.setPointSize(20)
    self.label.setFont(font)
    self.label.setLayoutDirection(QtCore.Qt.LeftToRight)
    self.label.setAlignment(QtCore.Qt.AlignLeading|QtCore.Qt.AlignLeft|QtCore.Qt.AlignVCenter)
    self.label.setObjectName("label")

    self.botonUpdateLibro = QtWidgets.QPushButton(DeleteWindow)
    self.botonUpdateLibro.setGeometry(QtCore.QRect(20, 70, 221, 28))
    self.botonUpdateLibro.setObjectName("botonUpdateLibro")
    self.botonUpdateLibro.clicked.connect(self.clickDelete)

    self.textoActualizacionDelete = QtWidgets.QLabel(DeleteWindow)
    self.textoActualizacionDelete.setGeometry(QtCore.QRect(10, 110, 241, 31))
    font = QtGui.QFont()
    font.setPointSize(10)
    self.textoActualizacionDelete.setFont(font)
    self.textoActualizacionDelete.setText("")
    self.textoActualizacionDelete.setAlignment(QtCore.Qt.AlignCenter)
    self.textoActualizacionDelete.setObjectName("textoActualizacionDelete")

    self.retranslateUi(DeleteWindow)
    QtCore.QMetaObject.connectSlotsByName(DeleteWindow)
```

Finalmente al hacer click en el botón delete, se asigna el valor del recuadro de texto a la variable id que se le pasa a la consulta para eliminar el libro.

```
def clickDelete(self):
    id = int(self.textEditId.toPlainText())

    db_config = read_db_config()

    query = "DELETE FROM libros WHERE idLibros = %s"

    try:
        # connect to the database server
        conn = MySQLConnection(**db_config)

        # execute the query
        cursor = conn.cursor()
        cursor.execute(query, (id,))

        self.textoActualizacionDelete.setText("Se elimino el libro")
        # accept the change
        conn.commit()

    except Error as error:
        print(error)

    finally:
        cursor.close()
        conn.close()
```