МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ
«ОРЛОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИМЕНИ И. С. ТУРГЕНЕВА»

Кафедра программной инженерии

ОТЧЕТ

по лабораторной работе № 7
на тему: «Разработка приложения, взаимодействующего с базой данных»
по дисциплине: «Программирование на языке Python»

Выполнил: Евдокимов Н.А.
Институт приборостроения, автоматизации и информационных технологий
Направление: 09.03.04 «Программная инженерия»
Группа: 71-ПГ
Проверила: Захарова О.В.

Отметка о зачете:

Дата: «____» _____ 2019 г.

Орёл, 2019

Задание

Разработать приложение (можно web-приложение), взаимодействующее с базой данных. Приложение должно иметь удобный графический интерфейс. Базу данных разработать в соответствии с темой своего варианта (варианты представлены ниже). База данных должна состоять из 2-3 связанных таблиц; одна таблица основная.

Функционал приложения:

- добавление информации в основную таблица;
- удалении информации из основной таблицы;
- отображение информации из основной таблицы.

Добавление и отображение информации должно быть реализовано в читаемой для пользователя форме (внешние ключи не отображать, вместо них отображать пользователю понятную информацию).

Код

```
from PyQt5 import QtWidgets, uic
from PyQt5.QtWidgets import QAction
from lab3.Book import Book
import sys
import pickle as p
import mysql.connector


class UI(QtWidgets.QMainWindow):
    def __init__(self):
        super(UI, self).__init__()
        uic.loadUi('lab7.ui', self)

        self.handler = Handler(self)

        self.btnClear = self.findChild(QtWidgets.QPushButton, 'btnClear')
        self.btnAdd = self.findChild(QtWidgets.QPushButton, 'btnAdd')
        self.btnDelete = self.findChild(QtWidgets.QPushButton, 'btnDelete')
        self.btnMessage = self.findChild(QtWidgets.QPushButton, 'btnMessage')
        self.btnSearch = self.findChild(QtWidgets.QPushButton, 'btnSearch')

        self.leName = self.findChild(QtWidgets.QLineEdit, 'leName')
        self.leIsbn = self.findChild(QtWidgets.QLineEdit, 'leIsbn')
        self.leAuthors = self.findChild(QtWidgets.QLineEdit, 'leAuthors')
        self.lePublisher = self.findChild(QtWidgets.QLineEdit, 'lePublisher')
        self.leSearch = self.findChild(QtWidgets.QLineEdit, 'leSearch')

        self.sbNPages = self.findChild(QtWidgets.QSpinBox, 'sbNPages')
        self.sbPrice = self.findChild(QtWidgets.QDoubleSpinBox, 'dsbPrice')
        self.sbPublicationYear = self.findChild(QtWidgets.QSpinBox, 'sbPublicationYear')

        self.cbMessageType = self.findChild(QtWidgets.QComboBox, 'cbMessageType')

        self.cbWithIcon = self.findChild(QtWidgets.QCheckBox, 'cbWithIcon')

        self.twBooks = self.findChild(QtWidgets.QTableView, 'twBooks')
        self.twBooks.setSelectionBehavior(QtWidgets.QAbstractItemView.SelectRows)

        self.btnAdd.clicked.connect(self.btn_add_clicked)
        self.btnClear.clicked.connect(self.clear_input_fields)
        self.btnDelete.clicked.connect(self.btn_delete_clicked)
        self.btnMessage.clicked.connect(self.btn_message_clicked)

        self.setWindowTitle('Lab7')
```

```
        self.handler.read_books_file()
        self.show()

    def closeEvent(self, event) -> None:
        answer = QtWidgets.QMessageBox.question(self, 'Выход', 'Вы точно хотите выйти?')
        if answer == QtWidgets.QMessageBox.No:
            event.ignore()
        else:
            super(QtWidgets.QMainWindow, self).closeEvent(event)

    def show_about_msg(self):
        message = QtWidgets.QMessageBox(self)
        message.setText('Евдокимов Н.А. 71-ПГ')
        message.setWindowTitle('Сообщение')
        message.setIcon(QtWidgets.QMessageBox.Information)

        message.exec()

    def read_books_file(self):
        self.handler.read_books_file()
        self.repaint_table()

    def btn_delete_clicked(self, event):
        if len(self.twBooks.selectedIndexes()) <= 0:
            return

        index = [idx.row() for idx in self.twBooks.selectionModel().selectedRows()][0]
        book_name = self.handler.book_list[index].name
        self.handler.remove_book(book_name)
        self.repaint_table()

    def repaint_table(self):
        self.twBooks.setRowCount(0)
        self.twBooks.setColumnCount(7)

        for book in self.handler.book_list:
            row_pos = self.twBooks.rowCount()
            self.twBooks.insertRow(row_pos)
            self.twBooks.setItem(row_pos, 0, QtWidgets.QTableWidgetItem(book.name))
            self.twBooks.setItem(row_pos, 1, QtWidgets.QTableWidgetItem(str(book.publication_year)))
            self.twBooks.setItem(row_pos, 2, QtWidgets.QTableWidgetItem(str(book.n_pages)))
            self.twBooks.setItem(row_pos, 3, QtWidgets.QTableWidgetItem(book.isbn))
            self.twBooks.setItem(row_pos, 4, QtWidgets.QTableWidgetItem(','.join(book.authors)))
            self.twBooks.setItem(row_pos, 5, QtWidgets.QTableWidgetItem(book.publisher))
            self.twBooks.setItem(row_pos, 6, QtWidgets.QTableWidgetItem(str(book.price)))

    def btn_message_clicked(self, event):
        message_type = str(self.cbMessageType.currentText())

        is_sad = message_type == 'Грустное сообщение'
        if is_sad:
            message_str = ':('
        else:
            message_str = ':)'

        message = QtWidgets.QMessageBox()
        message.setText(message_str)
        message.setWindowTitle('Сообщение')

        if self.cbWithIcon.isChecked() and is_sad:
            message.setIcon(QtWidgets.QMessageBox.Critical)
        elif self.cbWithIcon.isChecked() and not is_sad:
            message.setIcon(QtWidgets.QMessageBox.Information)
```

```
        message.exec()

    def btn_add_clicked(self, event):
        name = self.leName.text()
        pub_year = self.sbPublicationYear.value()
        n_pages = self.sbNPages.value()
        isbn = self.leIsbn.text()
        authors = self.leAuthors.text().replace(' ', '').upper().split(',')
        publisher = self.lePublisher.text()
        price = self.sbPrice.value()

        if len(name) <= 0 or n_pages <= 0 or len(isbn) <= 0 or len(authors) <= 0 or len(publisher) <= 0 or price <= 0:
            message = QtWidgets.QMessageBox()
            message.setText('Недостаточно данных о книге')
            message.setIcon(QtWidgets.QMessageBox.Critical)
            message.setWindowTitle('Книга не добавлена')
            message.exec()

            return

        for book in self.handler.book_list:
            if book.name == name:
                message = QtWidgets.QMessageBox()
                message.setText('Такая книга уже есть')
                message.setIcon(QtWidgets.QMessageBox.Critical)
                message.setWindowTitle('Книга не добавлена')
                message.exec()

                return

        self.handler.add_book(name, pub_year, n_pages, isbn, authors, publisher, price)
        self.repaint_table()

    def clear_input_fields(self):
        self.leName.setText('')
        self.lePublisher.setText('')
        self.leAuthors.setText('')
        self.leIsbn.setText('')
        self.sbPublicationYear.setValue(0)
        self.sbPrice.setValue(0)
        self.sbNPages.setValue(0)


class Handler:
    def __init__(self, ui):
        self.window = ui
        self.file_path = None
        self.book_list = []
        self.db = mysql.connector.connect(
            host='localhost',
            user='root',
            passwd='password',
            database='python_lab_7'
        )

    def read_books_file(self):
        self.book_list = []

        cursor = self.db.cursor()
        sql_statement = 'SELECT books.id, books.num_pages, books.pub_year, books.price, books.name, ' \
                        'books.isbn_id, books.publisher_id FROM books'
        cursor.execute(sql_statement)
```

```
        selected_books = cursor.fetchall()

        for selected_book in selected_books:
            # get isbn
            sql_statement = 'SELECT isbn.name FROM isbn WHERE isbn.id = %s'
            cursor.execute(sql_statement, (selected_book[5], ))
            isbn = cursor.fetchall()[0][0]

            # get publisher
            sql_statement = 'SELECT publisher.name FROM publisher WHERE publisher.id = %s'
            cursor.execute(sql_statement, (selected_book[6], ))
            publisher = cursor.fetchall()[0][0]

            # get_authors
            sql_statement = 'SELECT author.name FROM books ' \
                            'INNER JOIN author_has_books ON author_has_books.books_id = books.id ' \
                            'INNER JOIN author ON author_has_books.author_id = author.id ' \
                            'WHERE books.id = %s'
            cursor.execute(sql_statement, (selected_book[0], ))
            authors = [author_name[0] for author_name in cursor.fetchall()]

            book = Book(selected_book[4], selected_book[2], selected_book[1],
                        isbn, authors, publisher, selected_book[3])
            self.book_list.append(book)

        cursor.close()
        self.window.repaint_table()

    def add_book(self, name, publication_year, n_pages,
                 isbn, authors, publisher, price):

        book = Book(name, publication_year, n_pages, isbn, authors, publisher, price)
        self.book_list.append(book)

        # Check for ISBN
        cursor = self.db.cursor()
        sql_statement = 'SELECT isbn.id, isbn.name FROM isbn WHERE isbn.name = %s'
        cursor.execute(sql_statement, (isbn, ))
        select_result = cursor.fetchall()
        is_isbn_stored = len(select_result) > 0
        if not is_isbn_stored:
            sql_statement = 'INSERT INTO isbn (`name`) VALUES (%s)'
            cursor.execute(sql_statement, (isbn, ))
            self.db.commit()
            isbn_id = cursor.getlastrowid()
        else:
            isbn_id = select_result[0][0]

        # Check for publisher
        sql_statement = 'SELECT publisher.id, publisher.name FROM publisher WHERE publisher.name = %s'
        cursor.execute(sql_statement, (publisher, ))
        select_result = cursor.fetchall()
        is_publisher_stored = len(select_result) > 0
        if not is_publisher_stored:
            sql_statement = 'INSERT INTO publisher (`name`) VALUES (%s)'
            cursor.execute(sql_statement, (publisher,))
            self.db.commit()
            publisher_id = cursor.getlastrowid()
        else:
            publisher_id = select_result[0][0]

        # check for authors
        author_ids = []
```

```
        sql_statement = 'SELECT author.id from author WHERE author.name = %s'
        for author in authors:
            cursor.execute(sql_statement, (author, ))
            select_result = cursor.fetchall()
            if len(select_result) <= 0:
                sql = 'INSERT INTO author (`name`) VALUES (%s)'
                cursor.execute(sql, (author, ))
                self.db.commit()
                author_ids.append(cursor.getlastrowid())
            else:
                author_ids.append(select_result[0][0])

        sql_statement = 'INSERT INTO books (num_pages, pub_year, price, `name`, isbn_id, publisher_id) ' \
                        'VALUES (%s, %s, %s, %s, %s, %s)'
        cursor.execute(sql_statement,
                       (book.n_pages, book.publication_year, book.price, book.name, isbn_id, publisher_id))
        self.db.commit()

        book_id = cursor.getlastrowid()
        for author_id in author_ids:
            sql_statement = 'INSERT INTO author_has_books (author_id, books_id) VALUES (%s, %s)'
            cursor.execute(sql_statement, (author_id, book_id))
            self.db.commit()

        cursor.close()

    def remove_book(self, book_name):
        cursor = self.db.cursor()

        # get book id
        sql_statement = 'SELECT books.id FROM books WHERE books.name = %s'
        cursor.execute(sql_statement, (book_name, ))
        book_id = cursor.fetchall()[0][0]

        # delete from author_has_books
        sql_statement = 'DELETE FROM author_has_books WHERE author_has_books.books_id = %s'
        cursor.execute(sql_statement, (book_id, ))
        self.db.commit()

        sql_statement = 'DELETE FROM books WHERE books.name = %s'
        cursor.execute(sql_statement, (book_name, ))
        cursor.close()
        self.db.commit()

        cursor.close()
        self.read_books_file()


if __name__ == '__main__':
    app = QtWidgets.QApplication(sys.argv)
    window = UI()
    app.exec()

# read_books_file()
# repaint_table()
```