**

Колледж космического машиностроения и технологии

**КУРСОВОЙ ПРОЕКТ**

**По дисциплине «Прикладное программирование»**

**Тема: «Разработка приложения «Библиотека»»**

Выполнил студент

Рубановский О.А

Группа П1-17

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

Проверил преподаватель

Гусятинер Леонид Борисович

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(подпись)

\_\_\_\_\_\_\_\_\_\_\_ (Дата сдачи работы)

Королев, 2020

**Введение**

Целью данного курсового проекта является создание программы по ведению ведению базы данных библиотеки. Он рассчитан на упрощение ведения учета и подсчета книг .

В эпоху автоматизации данная программа позволит облегчить статестический подсчет работников библиотек. Так же в данном проекте будет простой для понимания интерфейс и небольшой порог вхождения.

В первой части будет рассмотрена предметная область данной темы, а также несколько продуктов по данной теме.

Во второй части будут рассмотрены инструменты и модули, которые были разработаны, структура программной части и листинги ключевых частей программных модулей.

В третьей части будет рассмотрено руководство для пользователей.

В заключительной части будет приведен To-do лист с планами по доработки программы, а также сделаны общие выводы о получившемся проекте.

# **Теоретическая часть**

## **Описание предметной области**

Библиоте́ка — учреждение, собирающее и хранящее произведения печати и письменности для общественного пользования, а также осуществляющее справочно-библиографическую работу. В настоящее время всё большее количество книг оцифровываются и хранятся на электронных носителях. Для хранения и чтения используют форматы FB2, EPUB, MOBI, PDF, DjVu, RTF, TXT и другие.

Всего на сегодня в библиотеках находится примерно 130 миллионов Наимонований книг (по версии Google)  
Библиотеки бывают:

Государственные

Бюджетные

Муниципальные

Частные

Личные (семейные)

Учебные

Социальные виды универсальных библиотек:

Публичная

Для слепых

Детская

Юношеская

Вузовская

Академическая

Отраслевые

Специальные отраслевые библиотеки бывают:

Медицинские

Сельскохозяйственные

Технические

Художественные

И другие

**Национальная библиотека** призвана обеспечить сохранность и доступность всей печатной и смежной продукции, выпущенной и выпускаемой данным государством или имеющей к нему то или иное отношение, могущей быть востребованной читателями. Для обеспечения полноты фонда национальной библиотеки во многих странах используется система обязательного экземпляра. В России функции национальной библиотеки выполняют Российская государственная библиотека (бывш. имени Ленина) в Москве и Российская национальная библиотека в Санкт-Петербурге.  


Национальная библиотека Белоруссии

**Региональные библиотеки** выполняют роль филиалов национальной библиотеки, что особенно актуально для отдалённых регионов страны. В России особенно важную роль играют несколько региональных библиотек Урала и Сибири, наряду с двумя национальными библиотеками наделённых правом получения обязательного экземпляра.



Региональная библиотека Болгарии

**Публичные библиотеки** обеспечивают читателей наиболее употребительными и популярными изданиями.  
  


Первая публичная библиотека Уфы

**Специальные библиотеки** собирают издания определённого типа (нотные издания, книги для слепых, государственные стандарты, патенты, предсказания на пальмовых листьях и т. п.) или определённой тематики. Необходимость специальных библиотек в ряде случаев вызывается особыми условиями хранения изданий и пользования ими, но по большей части связана с невозможностью сосредоточить слишком большое количество изданий в одном помещении и обеспечить работу в одном учреждении высококвалифицированных специалистов по слишком разным отраслям книжного дела. В России в последние десятилетия особенно важную роль стала играть Всероссийская Государственная библиотека иностранной литературы, взявшая на себя ряд периферийных для библиотеки функций и превратившаяся благодаря этому в крупный культурный центр.



**Библиотеки для слепых** обеспечивают доступ к информации для слепых и слабовидящих читателей. Такие библиотеки содержат книги, набранные рельефным шрифтом Брайля и аудиокниги на разных носителях. Крупнейшая в России библиотека для слепых — Российская Государственная библиотека для слепых. Помимо книг, набранных рельефным шрифтом и аудиокниг, она содержит большую коллекцию рельефно-объёмных моделей, позволяющих слепым узнать облик различных объектов.

**Университетские, институтские, школьные библиотеки** нацелены, главным образом, на обеспечение учащихся литературой, необходимой для учебного процесса и по составу фонда приближаются к специальным. То же можно сказать и о ведомственных библиотеках. Однако, в отличие от специальных библиотек, институтские и ведомственные библиотеки не являются общедоступными и обслуживают только читателей, относящихся к соответствующему учебному заведению или ведомству. Впрочем, реализация этого принципа зависит от национальной традиции и конкретных социокультурных условий: так, в США ряд университетских библиотек обеспечивает свободный доступ к своим фондам для всех желающих.

## **Описание существующих разработок**

В это разделе рассмотрены некоторые уже имеющиеся программы для управления библиотекой, а также цели таких программ и требования.

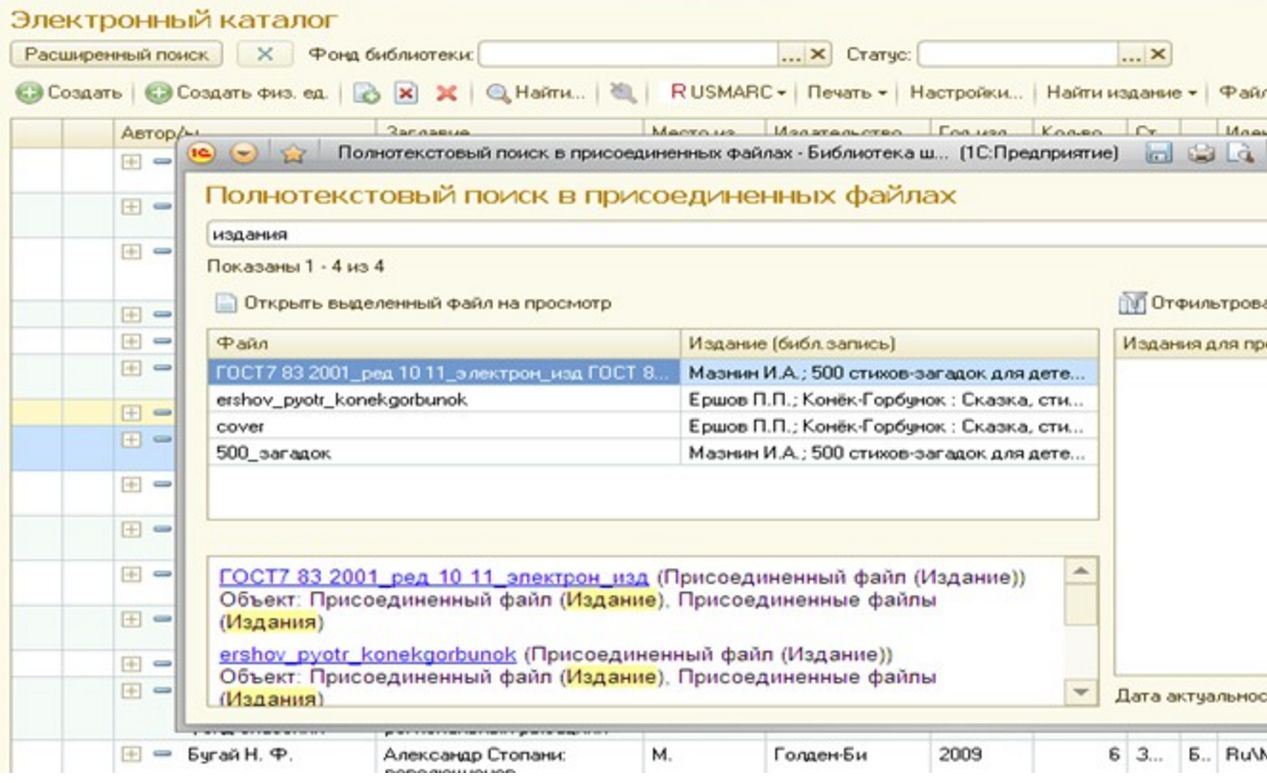
Интерфейс таких программ для управления отелями чаще всего простой и понятный для обычных пользователей.

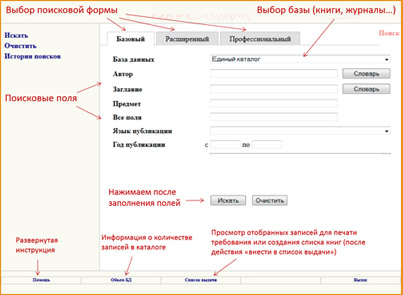
Цели использования системы управления отелем:

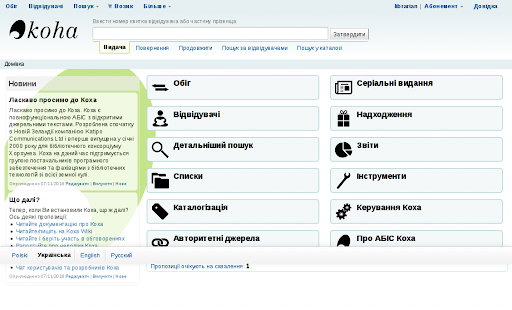
1. Автоматизация работы персонала.
2. Синхронизация с системами трекинга книг.
3. Ведение бухгалтерии.

Основные требования:

1. Ведение отчетности в количестве позиций и наименований книг.
2. Размещение книг по местоположению .
3. Оплата и трекинг книг .

**1.2.1 1С:Библиотека**Программа для автоматизации деятельности библиотек любого типа и назначения. Продукт позволяет автоматизировать рабочие процессы библиотеки, в зависимости от ее назначения, типа, состава фондов, может быть интегрирован с другими типовыми решениями фирмы "1С". Есть конфигурации для школы, ВУЗа, колледжа.  
  
 **1.2.2** OPAC-Global

Aвтоматизированная библиотечная информационная система, основанная на облачных технологиях. Версии для отдельной библиотеки и для сети библиотек. Имеет возможность загрузки авторитетных данных из внешних файлов и позволяет создавать Сводный каталог через онлайн-каталогизацию в едином каталоге.  


**1.2.3** **Koha**Open-source автоматизированная библиотечная информационная система (АБИС). Интерфейс для библиотекарей и читателей (посетителей). Поиск. Оборот книг и управление читателями. Модуль каталогизации со встроенным клиентом Z39.50. Система периодики для журналов или газет.  


# **Глава 2. Проектная часть**

## **2.1. Диаграмма Прецедентов**

В ходе разработки была построена диаграмма для “библиотекаря ”и непосредственно “Посетителя”  
На данных диаграммах мы можем сопоставить функционал для каждого пользователя

Диаграмма прецедентов (рис. 7). Показывает, что рабочий имеет множество выборов действия: просмотр данных по посетителям, Писателям, Книгам, получение статистических данных, внесение данных в любую из предоставленных таблиц. Посетитель(покупатель), будет иметь доступ только к просмотром книг которые есть в наличии и просмотра данных о писателе.

## Выбор инструментов

При выборе инструментов я исходил из некоторых критериев, который я считал самыми главными в разработке.

Важность критерия я выбирал из: низкая, средняя, высокая.

Таблица №1 важность критерия:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Критерий | Участие в корпоративном проекте | Простота сопровождения | Наличие библиотек связанных с выбранной Базой данных | Наличие документации на русском языке | Скорость разработки |
| Важность критерия | Низкая | Средняя | Низкая | Низкая | Высокая |
| Критерий | Простота разработки графического интерфейса | Скорость обучения |  |  |  |
| Важность критерия | Высокая | Высокая |  |  |  |

Исходя из этих критериев, я сравнил 3 языка программирования от 0 до 10 баллов за критерий исходя из своих предпочтений и знаний.

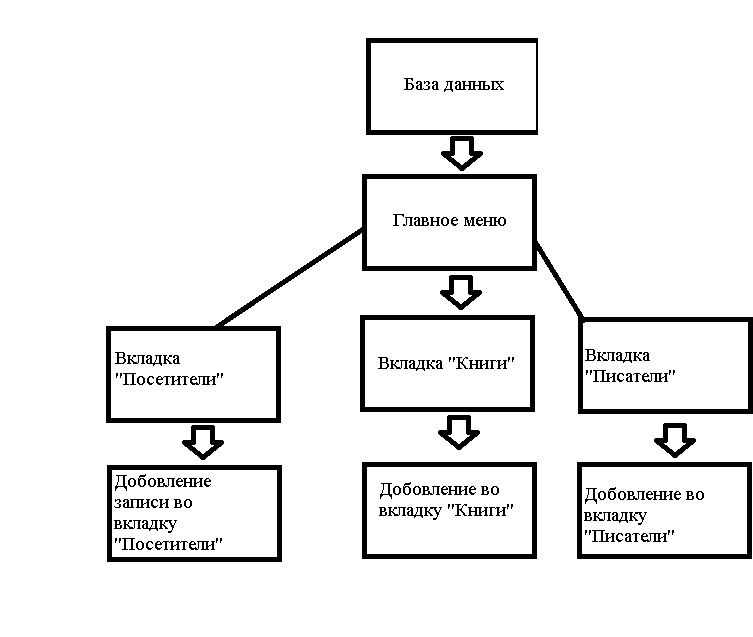
Таблица №2 сравнение языков программирования по критериям:

|  |  |  |  |
| --- | --- | --- | --- |
|  | C++ | C# | Python |
| Участие в корпоративном проекте | 4 | 11 | 6 |
| Простота сопровождения | 5 | 5 | 7 |
| Наличие библиотек | 7 | 3 | 10 |
| Наличие документации на русском языке | 8 | 7 | 8 |
| Скорость разработки | 6 | 6 | 10 |
| Простота разработки графического интерфейса | 5 | 8 | 10 |
| Скорость обучения | 10 | 5 | 10 |
| Итого баллов | 50 | 46 | 63 |

По результатам сравнения мной был выбран язык программирования Python.

## Проектирование сценария

Данная программа ориентирована на такой сценарий (рис. 8).



Сценарий программы

После запуска программы пользователь имеет несколько вариантов действий

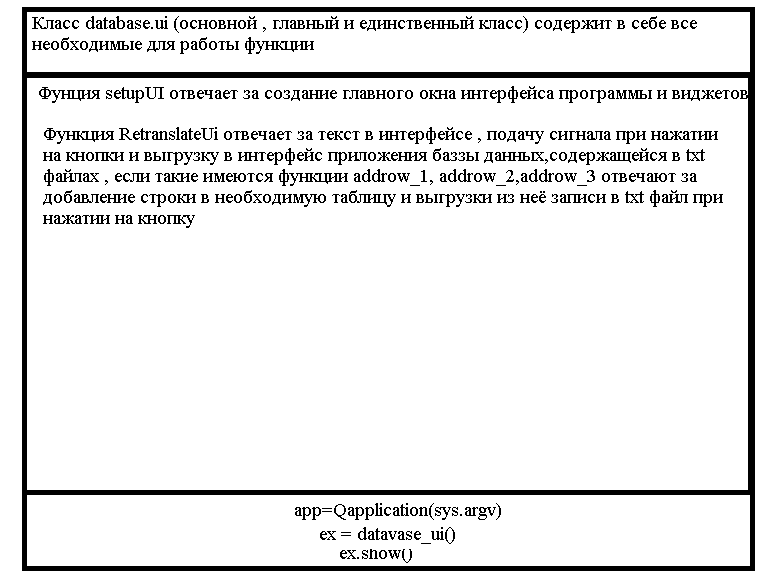
1. Зайти во вкладку «Посетители»
2. Зайти во вкладку «Книги»
3. Зайти во вкладку «Писатели»
4. Добавить запись в любую из перечисленных выше вкладок

При добавлении записи, она автоматически помещается в созданный файл типа txt.

При выборе пункта 4, информация будет выведена на консоль.

## 2.4 Построение диаграммы классов

В данном параграфе представлены все классы, которые используются в программе, а также их взаимосвязи .



Сценарий программы

## 2.5 Описание главного модуля

В главный модуль разработки входит один класс, который отвечает за взаимосвязь со всеми компонентами базы данных, а также за их включение.

Данный модуль состоит из одного кода, который приведен в листинге 1.

Листинг 1:

class database\_ui(QtWidgets.QWidget):  
  
 def \_\_init\_\_(self):  
 QtWidgets.QWidget.\_\_init\_\_(self)  
 self.setupUi(self)  
  
 def setupUi(self, UI\_Tool):  
 UI\_Tool.setObjectName("TPayne\_MySQL\_Tool")  
 UI\_Tool.resize(610, 431)  
 self.gridLayout = QtWidgets.QGridLayout(UI\_Tool)  
 self.gridLayout.setObjectName("gridLayout")  
 self.verticalLayout = QtWidgets.QVBoxLayout()  
 self.verticalLayout.setObjectName("verticalLayout")  
 self.label = QtWidgets.QLabel(UI\_Tool)  
 font = QtGui.QFont()  
 font.setPointSize(11)  
 self.label.setFont(font)  
 self.label.setLayoutDirection(QtCore.Qt.LeftToRight)  
 self.label.setScaledContents(True)  
 self.label.setAlignment(QtCore.Qt.AlignCenter)  
 self.label.setObjectName("label")  
 self.verticalLayout.addWidget(self.label)  
 self.horizontalLayout = QtWidgets.QHBoxLayout()  
 self.horizontalLayout.setObjectName("horizontalLayout")  
 self.tabWidget = QtWidgets.QTabWidget(UI\_Tool)  
 self.tabWidget.setObjectName("tabWidget")  
 self.tab = QtWidgets.QWidget()  
 self.tab.setObjectName("tab")  
 self.verticalLayout\_5 = QtWidgets.QVBoxLayout(self.tab)  
 self.verticalLayout\_5.setObjectName("verticalLayout\_5")  
 self.tableWidget\_1 = QtWidgets.QTableWidget(self.tab)  
 self.tableWidget\_1.setObjectName("tableWidget\_1")  
 self.tableWidget\_1.setColumnCount(3)  
 self.tableWidget\_1.setRowCount(1)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_1.setVerticalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_1.setHorizontalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_1.setHorizontalHeaderItem(1, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_1.setHorizontalHeaderItem(2, item)  
 self.tableWidget\_1.horizontalHeader().setCascadingSectionResizes(True)  
 self.tableWidget\_1.horizontalHeader().setSortIndicatorShown(False)  
 self.tableWidget\_1.horizontalHeader().setStretchLastSection(False)  
 self.tableWidget\_1.verticalHeader().setVisible(False)  
 self.verticalLayout\_5.addWidget(self.tableWidget\_1)  
 self.pushButton\_3 = QtWidgets.QPushButton(self.tab)  
 self.pushButton\_3.setObjectName("pushButton\_3")  
 self.verticalLayout\_5.addWidget(self.pushButton\_3)  
 self.tabWidget.addTab(self.tab, "")  
 self.tab\_2 = QtWidgets.QWidget()  
 self.tab\_2.setObjectName("tab\_2")  
 self.verticalLayout\_4 = QtWidgets.QVBoxLayout(self.tab\_2)  
 self.verticalLayout\_4.setObjectName("verticalLayout\_4")  
 self.tableWidget\_2 = QtWidgets.QTableWidget(self.tab\_2)  
 self.tableWidget\_2.setObjectName("tableWidget\_2")  
 self.tableWidget\_2.setColumnCount(3)  
 self.tableWidget\_2.setRowCount(1)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_2.setVerticalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_2.setHorizontalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_2.setHorizontalHeaderItem(1, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_2.setHorizontalHeaderItem(2, item)  
 self.tableWidget\_2.verticalHeader().setVisible(False)  
 self.verticalLayout\_4.addWidget(self.tableWidget\_2)  
 self.pushButton\_4 = QtWidgets.QPushButton(self.tab\_2)  
 self.pushButton\_4.setObjectName("pushButton\_4")  
 self.verticalLayout\_4.addWidget(self.pushButton\_4)  
 self.tabWidget.addTab(self.tab\_2, "")  
 self.tab\_3 = QtWidgets.QWidget()  
 self.tab\_3.setObjectName("tab\_3")  
 self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.tab\_3)  
 self.verticalLayout\_3.setObjectName("verticalLayout\_3")  
 self.tableWidget\_3 = QtWidgets.QTableWidget(self.tab\_3)  
 self.tableWidget\_3.setObjectName("tableWidget\_3")  
 self.tableWidget\_3.setColumnCount(4)  
 self.tableWidget\_3.setRowCount(1)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setVerticalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setHorizontalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setHorizontalHeaderItem(1, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setHorizontalHeaderItem(2, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setHorizontalHeaderItem(3, item)  
 self.tableWidget\_3.verticalHeader().setVisible(False)  
 self.verticalLayout\_3.addWidget(self.tableWidget\_3)  
 self.pushButton\_5 = QtWidgets.QPushButton(self.tab\_3)  
 self.pushButton\_5.setObjectName("pushButton\_5")  
 self.verticalLayout\_3.addWidget(self.pushButton\_5)  
 self.tabWidget.addTab(self.tab\_3, "")  
 self.horizontalLayout.addWidget(self.tabWidget)  
 self.verticalLayout.addLayout(self.horizontalLayout)  
 self.gridLayout.addLayout(self.verticalLayout, 0, 0, 1, 1)  
 self.pushButton = QtWidgets.QPushButton(UI\_Tool)  
 self.pushButton.setAutoDefault(False)  
 self.pushButton.setDefault(False)  
 self.pushButton.setFlat(False)  
 self.pushButton.setObjectName("pushButton")  
 self.gridLayout.addWidget(self.pushButton, 1, 0, 1, 1)  
 self.pushButton\_2 = QtWidgets.QPushButton(UI\_Tool)  
 self.pushButton\_2.setObjectName("pushButton\_2")  
 self.gridLayout.addWidget(self.pushButton\_2, 2, 0, 1, 1)  
  
 self.retranslateUi(UI\_Tool)  
 self.tabWidget.setCurrentIndex(0)  
 QtCore.QMetaObject.connectSlotsByName(UI\_Tool)  
  
 def retranslateUi(self, TPayne\_MySQL\_Tool):  
 \_translate = QtCore.QCoreApplication.translate  
 TPayne\_MySQL\_Tool.setWindowTitle(\_translate("TPayne\_MySQL\_Tool", "CEPDB Manager"))  
 self.label.setText(\_translate("TPayne\_MySQL\_Tool", "База данных книг"))  
 item = self.tableWidget\_1.verticalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "New Row"))  
 item = self.tableWidget\_1.horizontalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код клиента"))  
 item = self.tableWidget\_1.horizontalHeaderItem(1)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "ФИО"))  
 item = self.tableWidget\_1.horizontalHeaderItem(2)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Паспорт"))  
 self.pushButton\_3.setText(\_translate("TPayne\_MySQL\_Tool", "Добавить ряд для записи"))  
 self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), \_translate("TPayne\_MySQL\_Tool", "Клиенты"))  
 item = self.tableWidget\_2.verticalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "1"))  
 item = self.tableWidget\_2.horizontalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код валюты"))  
 item = self.tableWidget\_2.horizontalHeaderItem(1)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Название"))  
 item = self.tableWidget\_2.horizontalHeaderItem(2)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Курс продажи"))  
 self.pushButton\_4.setText(\_translate("TPayne\_MySQL\_Tool", "Добавить ряд для записи"))  
 self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_2), \_translate("TPayne\_MySQL\_Tool", "Валюта"))  
 item = self.tableWidget\_3.verticalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "New Row"))  
 item = self.tableWidget\_3.horizontalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код сделки"))  
 item = self.tableWidget\_3.horizontalHeaderItem(1)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код клиента"))  
 item = self.tableWidget\_3.horizontalHeaderItem(2)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код валюты"))  
 item = self.tableWidget\_3.horizontalHeaderItem(3)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Сумма"))  
 self.pushButton\_5.setText(\_translate("TPayne\_MySQL\_Tool", "Добавить ряд для записи"))  
 self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_3), \_translate("TPayne\_MySQL\_Tool", "Сделки"))  
 self.pushButton.setText(\_translate("TPayne\_MySQL\_Tool", "Определить долю сделок по каждой валюте"))  
 self.pushButton\_2.setText(\_translate("TPayne\_MySQL\_Tool", "Определить максимальный размер сделки в рублях"))  
  
 db\_file\_clients = open('database\_clients.txt', 'r')  
 db\_file\_curr = open('database\_curr.txt', 'r')  
 db\_file\_deals = open('database\_deals.txt', 'r')  
  
 for line\_clients in db\_file\_clients.readlines():  
 splited = line\_clients.split()  
 for i in range(0, 2):  
 self.tableWidget\_1.setItem(0, i, QTableWidgetItem(splited[i]))  
 self.tableWidget\_1.setItem(0, 2, QTableWidgetItem(splited[2]))  
 self.tableWidget\_1.insertRow(0)  
  
 for line\_curr in db\_file\_curr.readlines():  
 splited = line\_curr.split()  
 for i in range(0, 2):  
 self.tableWidget\_2.setItem(0, i, QTableWidgetItem(splited[i]))  
 self.tableWidget\_2.setItem(0, 2, QTableWidgetItem(splited[2]))  
 self.tableWidget\_2.insertRow(0)  
  
 for line\_deals in db\_file\_deals.readlines():  
 splited = line\_deals.split()  
 for i in range(0, 2):  
 self.tableWidget\_3.setItem(0, i, QTableWidgetItem(splited[i]))  
 self.tableWidget\_3.setItem(0, 2, QTableWidgetItem(splited[2]))  
 self.tableWidget\_3.setItem(0, 3, QTableWidgetItem(splited[3]))  
 self.tableWidget\_3.insertRow(0)  
  
 db\_file\_clients.close()  
 db\_file\_curr.close()  
 db\_file\_deals.close()  
  
 self.pushButton.pressed.connect(self.totaldeals)  
 self.pushButton\_2.pressed.connect(self.maxdeal)  
 self.pushButton\_3.pressed.connect(self.addrow\_1)  
 self.pushButton\_4.pressed.connect(self.addrow\_2)  
 self.pushButton\_5.pressed.connect(self.addrow\_3)  
  
 def addrow\_1(self):  
 db\_file\_clients = open('database\_clients.txt', 'a')  
 for i in range(0, 3):  
 temp = self.tableWidget\_1.item(0, i).text()  
 db\_file\_clients.write(temp)  
 db\_file\_clients.write(' ')  
 db\_file\_clients.write('\n')  
 db\_file\_clients.close()  
 self.tableWidget\_1.insertRow(0)  
  
 def addrow\_2(self):  
 db\_file\_curr = open('database\_curr.txt', 'a')  
 for i in range(0, 3):  
 temp = self.tableWidget\_2.item(0, i).text()  
 db\_file\_curr.write(temp)  
 db\_file\_curr.write(' ')  
 db\_file\_curr.write('\n')  
 db\_file\_curr.close()  
 self.tableWidget\_2.insertRow(0)  
  
 def addrow\_3(self):  
 db\_file\_deals = open('database\_deals.txt', 'a')  
 for i in range(0, 4):  
 temp = self.tableWidget\_3.item(0, i).text()  
 db\_file\_deals.write(temp)  
 db\_file\_deals.write(' ')  
 db\_file\_deals.write('\n')  
 db\_file\_deals.close()  
 self.tableWidget\_3.insertRow(0)  
  
 def totaldeals(self):  
  
 db\_file\_curr = open('database\_curr.txt', 'r')  
 db\_file\_deals = open('database\_deals.txt', 'r')  
 exit\_list = []  
 deals\_lines = []  
 curr\_lines = []  
 overall = 0  
  
 for line\_deals in db\_file\_deals.readlines():  
 splited\_deals = line\_deals.split()  
 deals\_lines.append(splited\_deals)  
  
 for line\_curr in db\_file\_curr.readlines():  
 splited\_curr = line\_curr.split()  
 curr\_lines.append(splited\_curr)  
 curr\_code = splited\_curr[0]  
 curr\_price = splited\_curr[2]  
 for i in range(len(deals\_lines)):  
 if curr\_code == splited\_deals[2]:  
 val = float(curr\_price) \* float(splited\_deals[3])  
 overall = overall + val  
  
 for i in range(len(curr\_lines)):  
 for j in range(len(deals\_lines)):  
 if curr\_lines[i][0] == deals\_lines[j][2]:  
 exit\_list.append(curr\_lines[i][1])  
 exit\_list.append(float(curr\_lines[i][2])\*float(deals\_lines[j][3])/float(overall))  
 print(\*exit\_list)  
  
 db\_file\_curr.close()  
 db\_file\_deals.close()  
  
 def maxdeal(self):  
  
 db\_file\_curr = open('database\_curr.txt', 'r')  
 db\_file\_deals = open('database\_deals.txt', 'r')  
 exit\_list = []  
 deals\_lines = []  
 curr\_lines = []  
  
 for line\_deals in db\_file\_deals.readlines():  
 splited\_deals = line\_deals.split()  
 deals\_lines.append(splited\_deals)  
  
 for line\_curr in db\_file\_curr.readlines():  
 splited\_curr = line\_curr.split()  
 curr\_lines.append(splited\_curr)  
  
 for i in range(len(curr\_lines)):  
 for j in range(len(deals\_lines)):  
 if curr\_lines[i][0] == deals\_lines[j][2]:  
 exit\_list.append(float(curr\_lines[i][2])\*float(deals\_lines[j][3]))

## **2.6 Описание спецификаций к модулям**

В данной главе будут описаны публичные члены модулей курсового проекта.

В главном модуле публичным объектом являются setupUI, который представляет из себя главную форму модуля, на который закреплены объекты, предназначенные для взаимосвязи с базой данных.

В модуле отвечающим за заполнение списков, и сортировку соответствующих таблиц так же присутствуют функции, которые будут описаны в следующем параграфе.

## Описание модулей

В данной главе мы разберем три главных модуля.

* Центральный модуль

Данный модуль представляет из себя набор обязательных элементов базы данных таких как таблицы, запросы и транзакции. И центральную функцию, которая представляет из себя настройку базы данных.

* Модуль работы с базами данных.

Модуль представляет из себя несколько графических форм каждая из которых обращается в центральный модуль за своей таблицей из базы данных для дальнейшей работы с ней и подключает свою навигационную систему этой таблице что обеспечивает удобную работу. Так же этот модуль для удобства ввода имеет скрипт для подстановки соответствующих значений в выпадающие списки.

* Модуль просмотра таблиц

Модуль представляет из себя несколько графических форм каждая из которых обращается в центральный модуль за своим запросом из базы данных для дальнейшей его вывода, так же подключает навигацию к полученному запросу. Так же этот модуль имеет два скрипта которые представляют из себя

* 1. Подстановку в выпадающий список соответствующе значения
  2. При выборе значения из списка производит изменение скрипта запроса и выводит новую отсортированную таблицу.

Листинг модуля просмотра таблицы:

def setupUi(self, UI\_Tool):  
 UI\_Tool.setObjectName("TPayne\_MySQL\_Tool")  
 UI\_Tool.resize(610, 431)  
 self.gridLayout = QtWidgets.QGridLayout(UI\_Tool)  
 self.gridLayout.setObjectName("gridLayout")  
 self.verticalLayout = QtWidgets.QVBoxLayout()  
 self.verticalLayout.setObjectName("verticalLayout")  
 self.label = QtWidgets.QLabel(UI\_Tool)  
 font = QtGui.QFont()  
 font.setPointSize(11)  
 self.label.setFont(font)  
 self.label.setLayoutDirection(QtCore.Qt.LeftToRight)  
 self.label.setScaledContents(True)  
 self.label.setAlignment(QtCore.Qt.AlignCenter)  
 self.label.setObjectName("label")  
 self.verticalLayout.addWidget(self.label)  
 self.horizontalLayout = QtWidgets.QHBoxLayout()  
 self.horizontalLayout.setObjectName("horizontalLayout")  
 self.tabWidget = QtWidgets.QTabWidget(UI\_Tool)  
 self.tabWidget.setObjectName("tabWidget")  
 self.tab = QtWidgets.QWidget()  
 self.tab.setObjectName("tab")  
 self.verticalLayout\_5 = QtWidgets.QVBoxLayout(self.tab)  
 self.verticalLayout\_5.setObjectName("verticalLayout\_5")  
 self.tableWidget\_1 = QtWidgets.QTableWidget(self.tab)  
 self.tableWidget\_1.setObjectName("tableWidget\_1")  
 self.tableWidget\_1.setColumnCount(3)  
 self.tableWidget\_1.setRowCount(1)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_1.setVerticalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_1.setHorizontalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_1.setHorizontalHeaderItem(1, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_1.setHorizontalHeaderItem(2, item)  
 self.tableWidget\_1.horizontalHeader().setCascadingSectionResizes(True)  
 self.tableWidget\_1.horizontalHeader().setSortIndicatorShown(False)  
 self.tableWidget\_1.horizontalHeader().setStretchLastSection(False)  
 self.tableWidget\_1.verticalHeader().setVisible(False)  
 self.verticalLayout\_5.addWidget(self.tableWidget\_1)  
 self.pushButton\_3 = QtWidgets.QPushButton(self.tab)  
 self.pushButton\_3.setObjectName("pushButton\_3")  
 self.verticalLayout\_5.addWidget(self.pushButton\_3)  
 self.tabWidget.addTab(self.tab, "")  
 self.tab\_2 = QtWidgets.QWidget()  
 self.tab\_2.setObjectName("tab\_2")  
 self.verticalLayout\_4 = QtWidgets.QVBoxLayout(self.tab\_2)  
 self.verticalLayout\_4.setObjectName("verticalLayout\_4")  
 self.tableWidget\_2 = QtWidgets.QTableWidget(self.tab\_2)  
 self.tableWidget\_2.setObjectName("tableWidget\_2")  
 self.tableWidget\_2.setColumnCount(3)  
 self.tableWidget\_2.setRowCount(1)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_2.setVerticalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_2.setHorizontalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_2.setHorizontalHeaderItem(1, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_2.setHorizontalHeaderItem(2, item)  
 self.tableWidget\_2.verticalHeader().setVisible(False)  
 self.verticalLayout\_4.addWidget(self.tableWidget\_2)  
 self.pushButton\_4 = QtWidgets.QPushButton(self.tab\_2)  
 self.pushButton\_4.setObjectName("pushButton\_4")  
 self.verticalLayout\_4.addWidget(self.pushButton\_4)  
 self.tabWidget.addTab(self.tab\_2, "")  
 self.tab\_3 = QtWidgets.QWidget()  
 self.tab\_3.setObjectName("tab\_3")  
 self.verticalLayout\_3 = QtWidgets.QVBoxLayout(self.tab\_3)  
 self.verticalLayout\_3.setObjectName("verticalLayout\_3")  
 self.tableWidget\_3 = QtWidgets.QTableWidget(self.tab\_3)  
 self.tableWidget\_3.setObjectName("tableWidget\_3")  
 self.tableWidget\_3.setColumnCount(4)  
 self.tableWidget\_3.setRowCount(1)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setVerticalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setHorizontalHeaderItem(0, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setHorizontalHeaderItem(1, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setHorizontalHeaderItem(2, item)  
 item = QtWidgets.QTableWidgetItem()  
 self.tableWidget\_3.setHorizontalHeaderItem(3, item)  
 self.tableWidget\_3.verticalHeader().setVisible(False)  
 self.verticalLayout\_3.addWidget(self.tableWidget\_3)  
 self.pushButton\_5 = QtWidgets.QPushButton(self.tab\_3)  
 self.pushButton\_5.setObjectName("pushButton\_5")  
 self.verticalLayout\_3.addWidget(self.pushButton\_5)  
 self.tabWidget.addTab(self.tab\_3, "")  
 self.horizontalLayout.addWidget(self.tabWidget)  
 self.verticalLayout.addLayout(self.horizontalLayout)  
 self.gridLayout.addLayout(self.verticalLayout, 0, 0, 1, 1)  
 self.pushButton = QtWidgets.QPushButton(UI\_Tool)  
 self.pushButton.setAutoDefault(False)  
 self.pushButton.setDefault(False)  
 self.pushButton.setFlat(False)  
 self.pushButton.setObjectName("pushButton")  
 self.gridLayout.addWidget(self.pushButton, 1, 0, 1, 1)  
 self.pushButton\_2 = QtWidgets.QPushButton(UI\_Tool)  
 self.pushButton\_2.setObjectName("pushButton\_2")  
 self.gridLayout.addWidget(self.pushButton\_2, 2, 0, 1, 1)  
  
 self.retranslateUi(UI\_Tool)  
 self.tabWidget.setCurrentIndex(0)  
 QtCore.QMetaObject.connectSlotsByName(UI\_Tool)  
  
def retranslateUi(self, TPayne\_MySQL\_Tool):  
 \_translate = QtCore.QCoreApplication.translate  
 TPayne\_MySQL\_Tool.setWindowTitle(\_translate("TPayne\_MySQL\_Tool", "CEPDB Manager"))  
 item = self.tableWidget\_1.verticalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "New Row"))  
 item = self.tableWidget\_1.horizontalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код посетителя"))  
 item = self.tableWidget\_1.horizontalHeaderItem(1)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "ФИО"))  
 item = self.tableWidget\_1.horizontalHeaderItem(2)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Паспорт"))  
 self.pushButton\_3.setText(\_translate("TPayne\_MySQL\_Tool", "Добавить ряд для записи"))  
 self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab), \_translate("TPayne\_MySQL\_Tool", "Посетители"))  
 item = self.tableWidget\_2.verticalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "1"))  
 item = self.tableWidget\_2.horizontalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код писателя"))  
 item = self.tableWidget\_2.horizontalHeaderItem(1)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Название"))  
 item = self.tableWidget\_2.horizontalHeaderItem(2)  
 self.pushButton\_4.setText(\_translate("TPayne\_MySQL\_Tool", "Добавить ряд для записи"))  
 self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_2), \_translate("TPayne\_MySQL\_Tool", "Писатель"))  
 item = self.tableWidget\_3.verticalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "New Row"))  
 item = self.tableWidget\_3.horizontalHeaderItem(0)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код сделки"))  
 item = self.tableWidget\_3.horizontalHeaderItem(1)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код Посетителя"))  
 item = self.tableWidget\_3.horizontalHeaderItem(2)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Код писателя"))  
 item = self.tableWidget\_3.horizontalHeaderItem(3)  
 item.setText(\_translate("TPayne\_MySQL\_Tool", "Наименование"))  
 self.pushButton\_5.setText(\_translate("TPayne\_MySQL\_Tool", "Добавить ряд для записи"))  
 self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab\_3), \_translate("TPayne\_MySQL\_Tool", "Книги"))

## Описание тестовых наборов модулей

В этом разделе показано умение применять средства отладки.

В ходе написания курсового проекта при попытке запустить скрипт, окно программы было принуждённо закрыто и было получено данное сообщение:

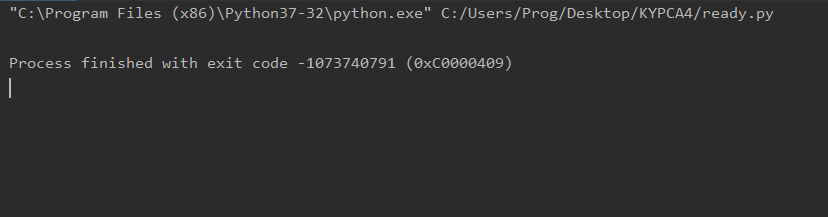


Рисунок 10 До применения средств отладки

После получения данного сообщения были просмотрены 275 строка модуля ready.py и была обнаружена ошибка, которая впоследствии была устранена, а после попытки запуска скрипта получено данное сообщение:

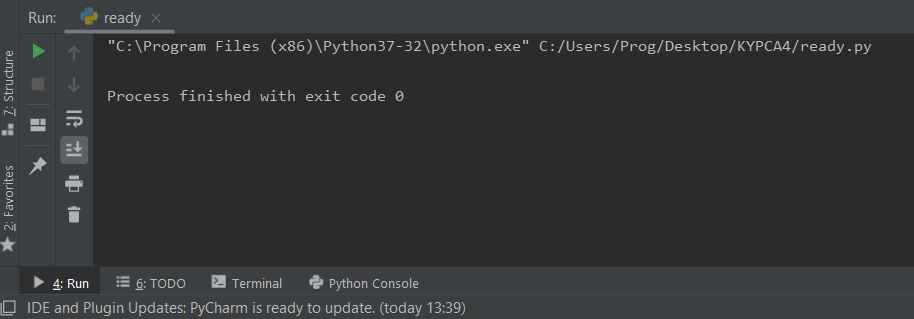


Рисунок 11. После применения средств отладки

Это означает что ошибка была устранена и скрипт запустился.