



Государственное бюджетное образовательное учреждение высшего образования
Московской области

ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ

Колледж космического машиностроения и технологий

КУРСОВОЙ ПРОЕКТ

По МДК.01.02 «Прикладное программирование»

Тема: «Разработка приложения "Библиотека"»

Выполнил студент

Рубановский О.А

Группа П1-17

_____ (Подпись)

_____ (Дата сдачи работы)

Проверил преподаватель

Гусятинер Л.Б.

_____ (Оценка)

_____ (Подпись)

Королёв 2020 г.

Оглавление

Введение	3
1. Теоретическая часть	4
1.1. Описание предметной области	4
1.2. Описание существующих разработок.....	9
1.2.1. 1С:Библиотека.....	9
1.2.2. OPAC-Global	10
1.2.3. Коһа	11
Глава 2. Проектная часть.....	12
2.1. Диаграмма прецедентов	12
2.2. Выбор инструментов	13
2.3. Проектирование сценария	15
2.4. Построение диаграммы классов	16
2.5. Описание главного модуля	17
2.6. Описание модулей.....	19
2.7. Описание тестовых наборов модулей	23
2.8. Анализ оптимальности использования памяти и быстродействия	24
3. Эксплуатационная часть	25
3.1. Руководство оператора	25
Заключение.....	32
Список литературы и интернет-источников.....	33
Приложение 1. Код программы.....	34

Введение

Целью данного курсового проекта является написание программы «Библиотека» для упрощения их работы. Эта тема является актуальной на данный момент, потому что в наше время все стремятся все автоматизировать или упростить. Данный курсовой проект позволит облегчить работу библиотекарей, Так же в данном проекте будет простой для понимания интерфейс и требовать будет программа 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.

1. Теоретическая часть

1.1. Описание предметной области

Библиотека — 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 наименований книг (по версии Google)

Библиотеки бывают: государственные, бюджетные, муниципальные, местные, личные (семейные), учебные

Социальные виды универсальных библиотек: публичная; для слепых; детская; юношеская; вузовская; академическая; отраслевые

Специальные отраслевые библиотеки бывают: медицинские; сельскохозяйственные; технические; художественные; и так далее

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 (бывший. имени Ленина) в Москве и Российская национальная библиотека в Санкт-Петербурге.

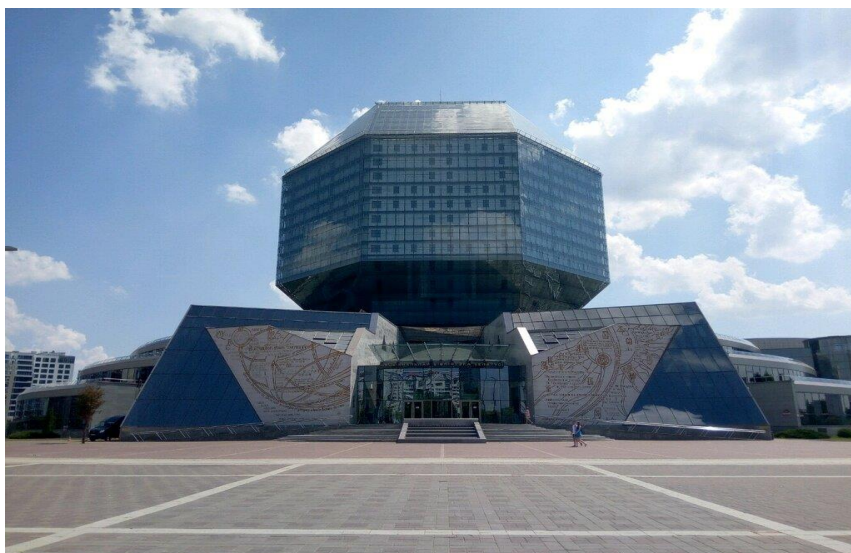


Рисунок 1.Национальная библиотека Белоруссии

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.



Рисунок 2.Региональная библиотека Болгарии

Публичные библиотеки обеспечивают читателей наиболее употребительными и популярными изданиями.



Рисунок 3.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.



Рисунок 4 Библиотека института имени Баумана

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.

1.2. 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.

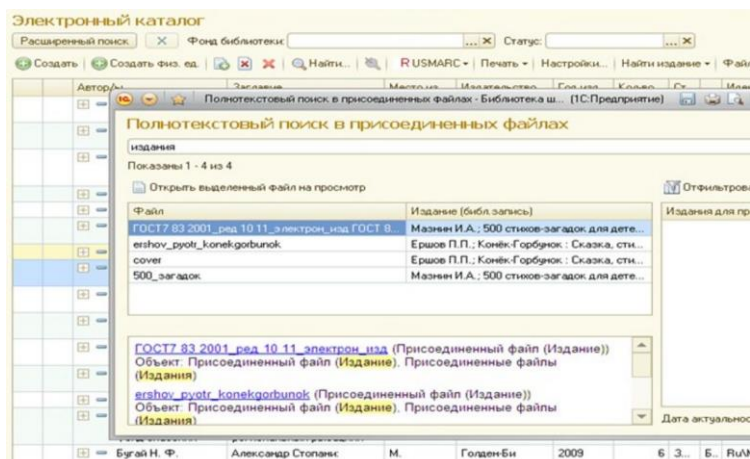


Рисунок 5 1С:Библиотека окно работы

1.2.2. OPAC-Global

Автоматизированная 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.

The screenshot shows the OPAC-Global search interface. At the top, there are two main sections: "Выбор поисковой формы" (Search form selection) and "Выбор базы (книги, журналы...)" (Database selection). Under "Выбор поисковой формы", there are three tabs: "Базовый" (Basic), "Расширенный" (Advanced), and "Профессиональный" (Professional). The "Базовый" tab is selected. Below the tabs, there are several input fields: "База данных" (Database), "Автор" (Author), "Заглавие" (Title), "Предмет" (Subject), "Все поля" (All fields), "Язык публикации" (Publication language), and "Год публикации" (Publication year). The "База данных" field is set to "Единый каталог". There are also "Словарь" (Dictionary) buttons next to the "Автор" and "Заглавие" fields. At the bottom of the search form, there are "Искать" (Search) and "Очистить" (Clear) buttons. On the left side, there are links: "Искать", "Очистить", and "История поисков". At the bottom of the page, there are links: "Помощь" (Help), "Объем КД" (Volume of CD), "Ссылка выдачи" (Link to issue), and "Выход" (Exit). Red arrows point to various elements: "Выбор поисковой формы", "Выбор базы (книги, журналы...)", "Поисковые поля" (Search fields), "Нажимаем после заполнения полей" (Click after filling the fields), "Развернутая инструкция" (Expanded instruction), "Информация о количестве записей в каталоге" (Information about the number of records in the catalog), and "Просмотр отобранных записей для печати требования или создания списка книг (после действия «внести в список выдачи»)» (View selected records for printing requirements or creating a list of books (after the action «add to the list of issues»)).

Рисунок 6 OPAC-Global окно работы

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.

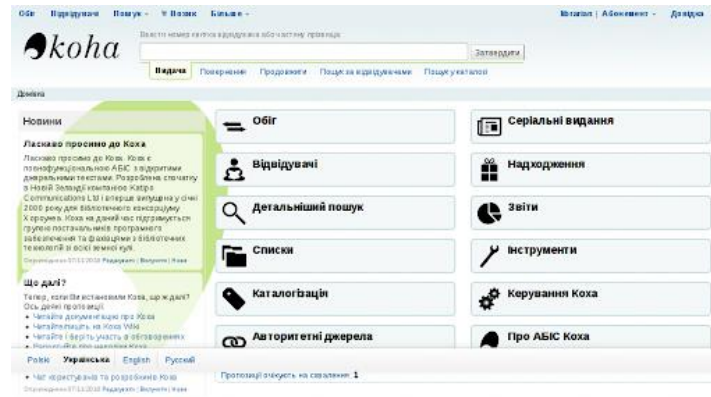


Рисунок 7 Koha окно работы

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

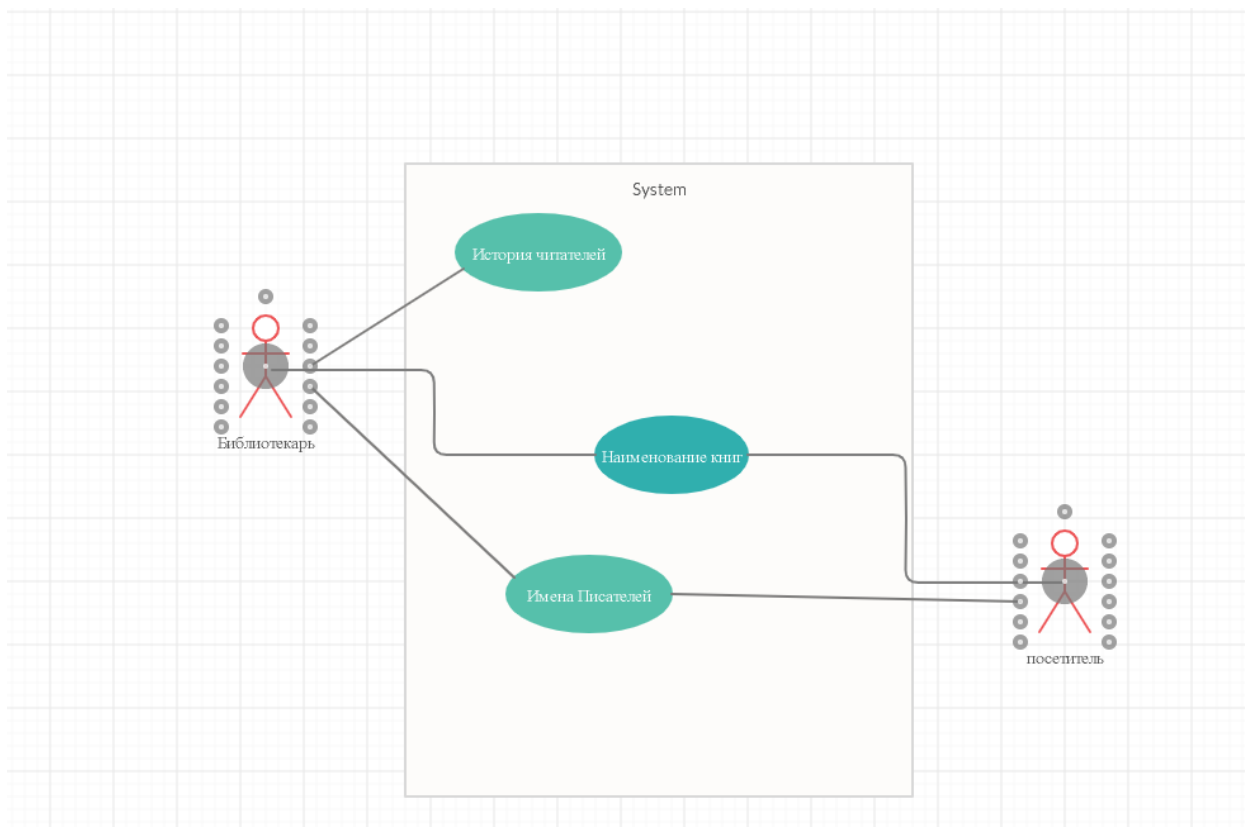


Рисунок 8 Диаграмма прецедентов

Диаграмма прецедентов. 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 просмотру книг, которые есть в наличии и просмотра данных о писателе.

2.2. 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:

Критерий/Язык программирования	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.

2.3. Проектирование сценария

Данная программа ориентирована на такой сценарий (рис. 9).

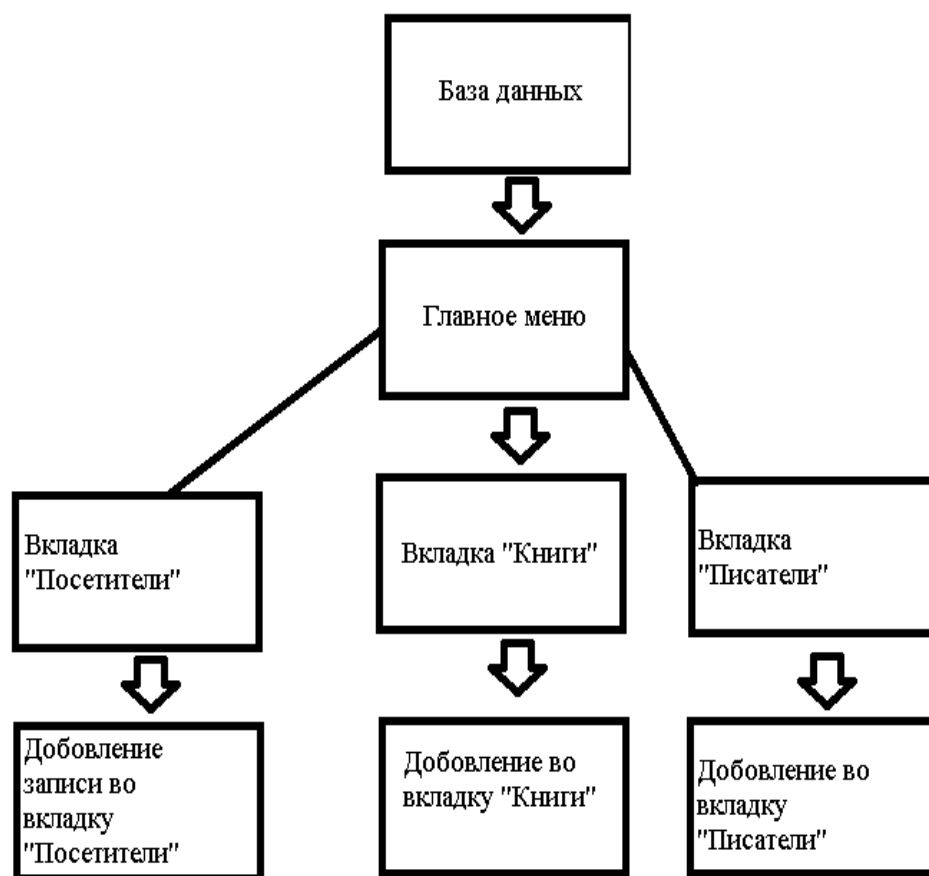


Рисунок 9.Сценарий программы

После запуска программы пользователь имеет несколько вариантов действий

1. Зайти во вкладку «Читатели»
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.

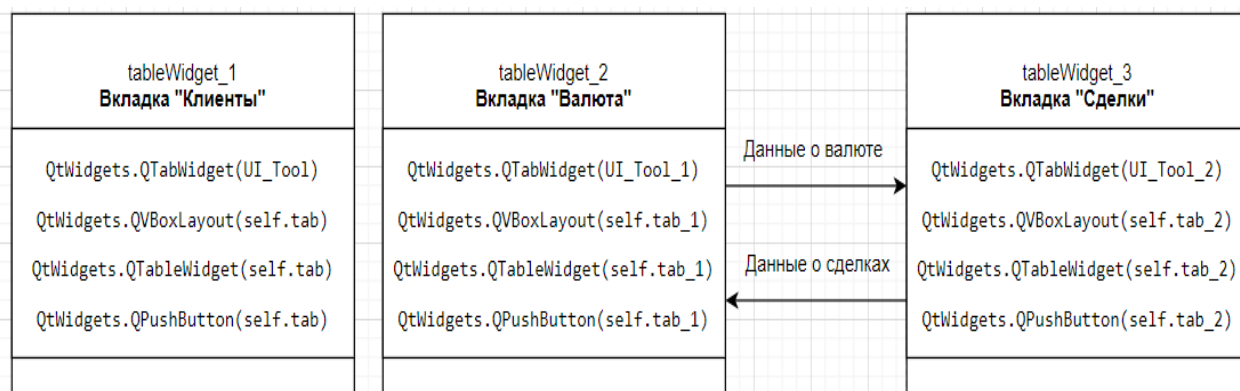


Рисунок 10 Диаграмма классов.

Класс `tableWidget_1` оперирует данными из вкладки «Клиенты». Классы `tableWidget_2` и `tableWidget_3` позволяют взаимодействовать с вкладками программы «Валюта» и «Сделки» соответственно. Между ними существует связь, позволяющая производить вычисления, нажатием на кнопки внутри 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. Функция «retranslateUI».

```
def retranslateUi(self, TPayne_MySQL_Tool):
    #построчное прочтение файла для занесение данных в таблицу Посетители
    for line_clients in db_file_clients.readlines():
        #метод split разделяет строку на элементы
        splited = line_clients.split()
        #занесение данных из текстовых файлов в таблицу
        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)
    #Аналогичная работа. Данные заносятся в таблицу Валюты
    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)
    #Аналогичная работа. Данные заносятся в таблицу Сделки
    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)

            temp = self.tableWidget_1.item(0, i).text()
            db_file_books.write(temp)
            db_file_books.write(' ')
            db_file_books.write('\n')
            db_file_books.close()
            self.tableWidget_1.insertRow(0)

def addrow_2(self):
    db_file_readers = open('database_readers.txt', 'a')
    for i in range(0, 3):
        temp = self.tableWidget_2.item(0, i).text()
        db_file_readers.write(temp)
        db_file_readers.write(' ')
        db_file_readers.write('\n')
        db_file_readers.close()
        self.tableWidget_2.insertRow(0)

def addrow_3(self):
    db_file_extra = open('database_extra.txt', 'a')
```

```

    for i in range(0, 4):
        temp = self.tableWidget_3.item(0, i).text()
        db_file_extra.write(temp)
        db_file_extra.write(' ')
    db_file_extra.write('\n')
    db_file_extra.close()
    self.tableWidget_3.insertRow(0)

#блок выполнения программы
if __name__ == '__main__':
    app = QApplication(sys.argv)
    ex = Library_UI()
    ex.show()

    sys.exit(app.exec_())

```

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.

2.6. Описание модулей

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 соответствующее значения
2. При выборе значения из списка производит изменение скрипта запроса и выводит новую отсортированную таблицу.

Листинг 2. модуль 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)

```

```

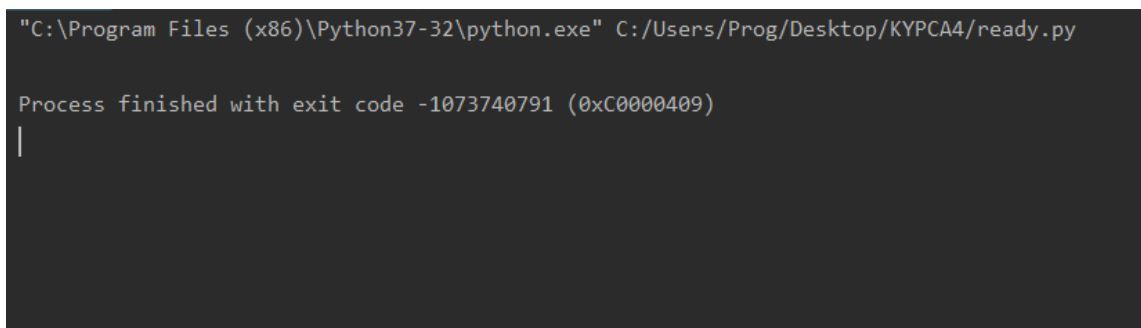
def retranslateUi(self, TPayne_MySQL_Tool):
    _translate = QtCore.QCoreApplication.translate
    TPayne_MySQL_Tool.setWindowTitle(_translate("TPayne_MySQL_Tool", "CEPDB
Manager"))
    item = self.tableWidget_1.verticalHeaderItem(0)
    item.setText(_translate("TPayne_MySQL_Tool", "New Row"))
    item = 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"))

```


2.7. Описание тестовых наборов модулей

В этом разделе показано умение применять средства отладки.

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



```
"C:\Program Files (x86)\Python37-32\python.exe" C:/Users/Prog/Desktop/KYPCA4/ready.py  
  
Process finished with exit code -1073740791 (0xC0000409)  
|
```

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:

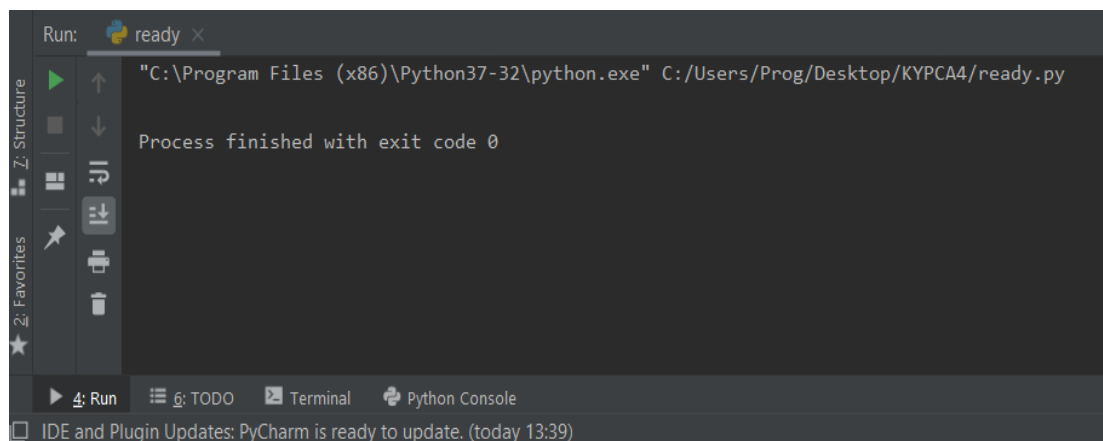


Рисунок 11. После применения средств отладки

Это означает что ошибка была устранена и скрипт запустился.

2.8. Анализ оптимальности использования памяти и быстродействия

Список принятых оптимальных решений:

Подключение некоторых модулей внутри функций/методов.

В данном проекте некоторые модули были подключены не в весь модуль, а только в функции/методы, которые его используют. Сделано это потому что работа с локальными объектами быстрее работы с глобальными объектами, к тому же импортироваться эти модули будут только при срабатывании этих функций что ускорит запуск программы.

3. Эксплуатационная часть

3.1. Руководство оператора

АННОТАЦИЯ

В данном программном документе приведено руководство оператора по применению и эксплуатации программы «Currency Exchange Point», предназначенной для облегчения работы пунктами обменом валюты.

В разделе «Назначение программы» указаны сведения о назначении программы и информация, достаточная для понимания функций программы и ее эксплуатации.

В разделе «Условия выполнения программы» указаны условия, необходимые для выполнения программы (минимальный состав аппаратных и программных средств и т.п.).

В разделе «Выполнение программы» указана последовательность действий оператора, обеспечивающих загрузку, запуск, выполнение и завершение программы, приведено описание функций, формата и возможных вариантов команд, с помощью которых оператор осуществляет загрузку и управляет выполнением программы, а также ответы программы на эти команды.

Оформление программного документа «Руководство оператора» произведено по требованиям ЕСПД (ГОСТ 19.101-77 ¹⁾, ГОСТ 19.103-77 ²⁾, ГОСТ 19.104-78* ³⁾, ГОСТ 19.105-78* ⁴⁾, ГОСТ 19.106-78* ⁵⁾, ГОСТ 19.505-79* ⁶⁾, ГОСТ 19.604-78* ⁷⁾).

1. Назначение программы

1.1. Функциональное назначение программы

¹⁾ ГОСТ 19.101-77 ЕСПД. Виды программ и программных документов

²⁾ ГОСТ 19.103-77 ЕСПД. Обозначение программ и программных документов

³⁾ ГОСТ 19.104-78* ЕСПД. Основные надписи

⁴⁾ ГОСТ 19.105-78* ЕСПД. Общие требования к программным документам

⁵⁾ ГОСТ 19.106-78* ЕСПД. Общие требования к программным документам, выполненным печатным способом

⁶⁾ ГОСТ 19.505-79* ЕСПД. Руководство оператора. Требования к содержанию и оформлению

⁷⁾ ГОСТ 19.604-78* ЕСПД. Правила внесения изменений в программные документы, выполненные печатным способом

1.2. Эксплуатационное назначение программы

Специальное программное обеспечение «Currency Exchange Point» может эксплуатироваться на объектах любого масштаба в сфере финансового бизнеса для облегчения работы персонала.

1.3. Состав функций

1.3.1. Функция смены таблицы

Эта функция позволяет менять таблицу в зависимости от необходимости.

1.3.2. Функция добавления элемента в таблицу.

Эта функция позволяет добавлять нужную информацию в таблицу.

1.3.3. Функция высчитывания наибольшей сделки.

1.3.4. Функция вывода доли по сделкам.

2. Условия выполнения программы

2.1. Минимальный состав аппаратных средств

ОС: Windows 10

Процессор: Как минимум 1 ГГц или SoC.

ОЗУ: 1 ГБ (для 32-разрядных систем) или 2 ГБ (для 64-разрядных систем).

Место на жестком диске: 20 мб.

Видеоадаптер: DirectX версии не ниже 9 с драйвером WDDM 1.0.

Дисплей: 800 x 600.

2.2. Минимальный состав программных средств

Дополнительные программные средства не требуются.

2.3. Требование к персоналу (пользователю)

Конечный пользователь программы должен обладать практическими навыками работы с графическим пользовательским интерфейсом операционной системы.

3.2. Выполнение программы

2.4. Загрузка и запуск программы

При запуске программы пользователь сразу же может наблюдать на экране окно программы. В окне таблицы уже может быть занесена информация, если заранее уже были созданы файлы database_clients.txt, database_curr.txt, database_deals.txt и в них уже была занесена какая-либо информация. В противном случае эти файлы автоматически создаются программой при первом открытии и не имеют никаких записей при открытии. В случае, представленном на рисунке 12, данные уже были занесены в текстовые файлы.

Код читателя	Фамилия	Имя	Отчество
--------------	---------	-----	----------

Добавить ряд для записи

Определить наиболее читаемого автора

Определить выдачу книг по датам

Рисунок 12. Начальный экран программы

2.5. Выполнение программы

Для перемещения по вкладкам базы данных необходимо привести курсор на одну из надписей «Книги», «Читатели», или «Выдача» и нажать левую кнопку мыши. По умолчанию программа открывается на вкладке «Клиенты», поэтому чтобы посмотреть базу данных по клиентам при запуске, на эту вкладку нажимать не требуется.

The screenshot shows a window titled "База данных библиотеки". It has three tabs: "Книги", "Читатели", and "Выдача". The "Книги" tab is active. Below the tabs is a table with five columns: "Код книги", "Название", "Автор", "Жанр", and "Год издания". The table is currently empty. Below the table is a horizontal scrollbar. Under the scrollbar is a button labeled "Добавить ряд для записи". At the bottom of the window are two more buttons: "Определить наиболее читаемого автора" and "Определить выдачу книг по датам".

Код книги	Название	Автор	Жанр	Год издания
-----------	----------	-------	------	-------------

< [Progress Bar] >

Добавить ряд для записи

Определить наиболее читаемого автора

Определить выдачу книг по датам

Рисунок 13.Открыта вкладка «Книги»

База данных библиотеки

Книги Читатели **Выдача**

Код читателя	Код книги	Дата выдачи

Добавить ряд для записи

Определить наиболее читаемого автора

Определить выдачу книг по датам

Рисунок 14.Открыта вкладка «Выдачи»

2.6. Функционал интерфейса.

Интерфейс программы имеет возможность растягиваться до необходимых размеров. При наведении курсора на один из углов окна программы и появления подсказки (курсор превращается в линию со стрелками на каждом краю, направленную на 45 градусов от нижней границы экрана монитора) и зажатия левой кнопки мыши с последующим передвижением курсора в произвольном направлении, можно заметить, что элементы окна (кроме текста) будут динамически расширяться в направлении перемещения курсора.

2.7. Функционал кнопок.

2.7.1. Кнопка «Добавить ряд для записи».

Данная кнопка расположена в каждой из вкладок и имеет один функционал. Для правильной работы необходимо сначала вписать данные (показано на рисунке 16) в верхний ряд таблицы и только после этого произвести нажатие.

База данных библиотеки

Книги Читатели **Выдача**

Код читателя	Код книги	Дата выдачи
111111	2222222	33.33.33

Добавить ряд для записи

Определить наиболее читаемого автора

Определить выдачу книг по датам

Рисунок 15 Добавление записи в “Выдачу”

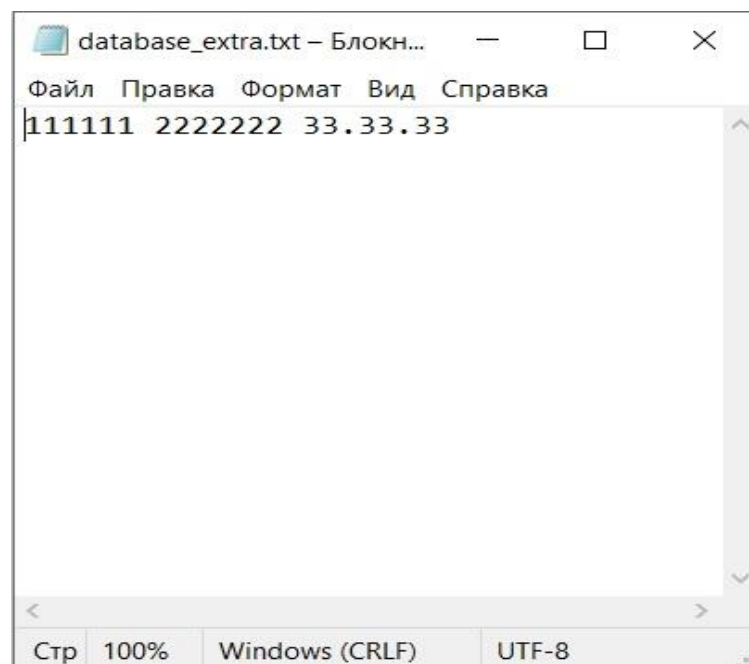


Рисунок 16 Состояние БД

Приведённые выше действия имеют аналогичный результат и для других вкладок.

Поочерёдно на консоли выводится название валюты из базы данных, за ней идёт доля сделок по этой валюте, относительно общей суммы сделок.

2.7.2. Кнопки внешнего интерфейса.



Рисунок 17.Кнопки внешнего интерфейса

Представлены кнопки внешнего интерфейса. Их функционал слева-направо: свернуть окно программы в трей, развернуть окно программы на весь экран, закрыть окно программы (при этом завершается работа программы).

Заключение

В результате выполнения курсового проекта была написана программа «Библиотека» для упрощения ведения статистики и слежения за книгами в библиотеке, и непосредственно выбора того или иного писателя.

В ходе работы были проанализированы предметная область, существующие разработки, посвященные данному направлению, получены практические навыки по созданию UI с помощью библиотеки PyQt5.

Также планируется продолжать работу над данным проектом с целью расширения возможностей и удобства приложения для пользователей. Планы по доработкам представлены ниже.

То-do лист:

1. Доработка всех пунктов главного меню, а именно функции трекинга любой книги и писателя
2. Добавления дат печати той или иной книги
3. Доработка интерфейса с целью упрощения работы с программой.
4. Перенос всех данных в БД.
5. Улучшение дизайна интерфейса.

Список литературы и интернет-источников.

1. Сайт для создания блок-схем.
<https://app.diagrams.net/>
2. Сайт для создания диаграмм прецедентов “Creately”.
<https://creately.com/ru/>
3. Сайт для создания блок-схем “Draw.io”.
<https://app.diagrams.net/>
4. Сайт 1.C Библиотека
<https://solutions.1c.ru/catalog/library>
5. Официальный сайт Koha
<https://koha.org>
6. Сайт OPAC’a
<https://opac-global.ru>

Приложение 1. Код программы

```

class Library_UI(QtWidgets.QWidget):
    """
    PyQt5 - библиотека для создания графического дизайна,
    окно программы, и всё его содержимое, были созданы
    с помощью её функций и методов
    """

# database_ui - основной класс программы, содержащий в себе все функции
class database_ui(QtWidgets.QWidget):

    def __init__(self):
        QtWidgets.QWidget.__init__(self)
        self.setupUi(self)
    """
    setupUI - главная функция программы, в ней идёт
    описание всех элементов графического интерфейса.
    Код setupUI, как и код функции retranslateUi
    были созданы при конвертации файла типа .ui,
    в файл типа .py с помощью команды pyuic5
    """
    def setupUi(self, Lib_Tool):

        """setName задаётся внутри программы QT Designer,
        позволяет дать имена объектам для более удобного использования"""
        #UI_Tool.setObjectName("CEPDB")
        #resize отвечает за открытие окна программы в установленном разрешении
        #UI_Tool.resize(610, 431)
        """ QGridLayout, QVBoxLayout - элементы модуля PyQt5,
        отвечающие за расстановку элементов внутри окна по сетке
        и их динамическое расширение при расширении окна программы """
        Lib_Tool.setObjectName("TPayne_MySQL_Tool")
        Lib_Tool.resize(610, 431)
        self.gridLayout = QtWidgets.QGridLayout(Lib_Tool)
        self.gridLayout.setObjectName("gridLayout")
        self.verticalLayout = QtWidgets.QVBoxLayout()
        self.verticalLayout.setObjectName("verticalLayout")
        self.label = QtWidgets.QLabel(Lib_Tool)
        font = QtGui.QFont()
        #QTabWidget - виджет переключаемых вкладок
        font.setPointSize(11)
        """setFont, setLayoutDirection, setScaledContents, setAlignment
        отвечают за свойства и поведение текста в окне программы"""
        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(Lib_Tool)
        self.tabWidget.setObjectName("tabWidget")

```

#следующие 2 строки устанавливают количество изначальных колонок и рядов в таблице

```

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")
    #QPushButton - кнопка
self.tableWidget_1.setColumnCount(5)
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)
item = QtWidgets.QTableWidgetItem()
self.tableWidget_1.setHorizontalHeaderItem(3, item)
item = QtWidgets.QTableWidgetItem()
self.tableWidget_1.setHorizontalHeaderItem(4, 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(4)
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)
item = QtWidgets.QTableWidgetItem()
self.tableWidget_2.setHorizontalHeaderItem(3, 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(3)
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)
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(Lib_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(Lib_Tool)
self.pushButton_2.setObjectName("pushButton_2")
self.gridLayout.addWidget(self.pushButton_2, 2, 0, 1, 1)

self.retranslateUi(Lib_Tool)
self.tabWidget.setCurrentIndex(1)
QtCore.QMetaObject.connectSlotsByName(Lib_Tool)
#функция retranslateUi устанавливает текст на элементы окна

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", ""))
    item = self.tableWidget_1.verticalHeaderItem(0)
    item.setText(_translate("TPayne_MySQL_Tool", "New Row"))
    #setText устанавливает текст на какой - либо элемент
    item = self.tableWidget_1.horizontalHeaderItem(0)
    item.setText(_translate("TPayne_MySQL_Tool", "Код книги"))
    item = self.tableWidget_1.horizontalHeaderItem(1)
    item.setText(_translate("TPayne_MySQL_Tool", "Название"))
    item = self.tableWidget_1.horizontalHeaderItem(2)

```



```

        item.setText(_translate("TPayne_MySQL_Tool", "Автор"))
        item = self.tableWidget_1.horizontalHeaderItem(3)
        item.setText(_translate("TPayne_MySQL_Tool", "Жанр"))
        item = self.tableWidget_1.horizontalHeaderItem(4)
        item.setText(_translate("TPayne_MySQL_Tool", "Год издания"))
        self.pushButton_3.setText(_translate("TPayne_MySQL_Tool", "Добавить
ряд для записи"))
        self.tabWidget.setTabText(self.tabWidget.indexOf(self.tab),
_translate("TPayne_MySQL_Tool", "Книги"))
        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 читателя"))
        item = self.tableWidget_2.horizontalHeaderItem(1)
        item.setText(_translate("TPayne_MySQL_Tool", "Фамилия"))
        item = self.tableWidget_2.horizontalHeaderItem(2)
        item.setText(_translate("TPayne_MySQL_Tool", "Имя"))
        item = self.tableWidget_2.horizontalHeaderItem(3)
        item.setText(_translate("TPayne_MySQL_Tool", "Отчество"))
        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", "Читатели"))
        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 читателя"))
        item = self.tableWidget_3.horizontalHeaderItem(1)
        item.setText(_translate("TPayne_MySQL_Tool", "Код книги"))
        item = self.tableWidget_3.horizontalHeaderItem(2)
        item.setText(_translate("TPayne_MySQL_Tool", "Дата выдачи"))
        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", "Выдача"))
        self.pushButton.setText(_translate("TPayne_MySQL_Tool", "Определить
наиболее читаемого автора"))
        self.pushButton_2.setText(_translate("TPayne_MySQL_Tool", "Определить
выдачу книг по датам"))

db_file_books = open('database_books.txt', 'r')
db_file_readers = open('database_readers.txt', 'r')
db_file_extra = open('database_extra.txt', 'r')

for line_books in db_file_books.readlines():
    splited = line_books.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_readers in db_file_readers.readlines():
    splited = line_readers.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_extra in db_file_extra.readlines():
        splited = line_extra.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)
#открытие текстовых файлов с данными для считывания из них информации
    db_file_books.close()
    db_file_readers.close()
    db_file_extra.close()
    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_books = open('database_books.txt', 'a')
# занесение данных из текстовых файлов в таблицу Книги
        for i in range(0, 3):
            temp = self.tableWidget_1.item(0, i).text()
            db_file_books.write(temp)
            db_file_books.write(' ')
            db_file_books.write('\n')
        db_file_books.close()
        self.tableWidget_1.insertRow(0)
    def addrow_2(self):
        db_file_readers = open('database_readers.txt', 'a')
        for i in range(0, 3):
            temp = self.tableWidget_2.item(0, i).text()
            db_file_readers.write(temp)
            db_file_readers.write(' ')
            db_file_readers.write('\n')
        db_file_readers.close()
        self.tableWidget_2.insertRow(0)
    def addrow_3(self):
        db_file_extra = open('database_extra.txt', 'a')
        for i in range(0, 4):
            temp = self.tableWidget_3.item(0, i).text()
            db_file_extra.write(temp)
            db_file_extra.write(' ')
            db_file_extra.write('\n')
        db_file_extra.close()
        self.tableWidget_3.insertRow(0)
#блок выполнения программы
    if __name__ == '__main__':
        app = QApplication(sys.argv)
        ex = Library_UI()
        ex.show()
        sys.exit(app.exec_())

```