Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный автомобильно-дорожный университет (СибАДИ)»

Институт Информационные системы, экономика и управление

Кафедра Информационная безопасность

Направление Информационная безопасность

Профиль Безопасность автоматизированных системы

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА**

**к курсовому проекту**

по дисциплине: «Технологии и методы программирования»

Тема работы «Разработка приложения для БД система учета финансовых операций для банковской системы»

Выполнил:

студент группы БИб-21Э1 Чигарев Александр Юрьевич

Курсовой проект сдан на проверку:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Чигарев Александр Юрьевич

(подпись студента)

Курсовой проект допущен к защите:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Руководитель работы

доцент кафедры, канд. пед. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Сапрыкина

доцент кафедры, канд. техн. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Толкачева

Курсовой проект защищен с оценкой:

«\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_2023 г.

Руководитель работы

доцент кафедры, канд. пед. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Сапрыкина

доцент кафедры, канд. техн. наук \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Толкачева

Омск 2023

Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«Сибирский государственный автомобильно-дорожный университет (СибАДИ)»

Кафедра «Информационная безопасность»

УТВЕРЖДАЮ

Зав. кафедрой ИБ, д.п.н., проф.

\_\_\_\_\_\_\_\_\_\_\_\_\_ З.В. Семенова

«\_\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_ 2023 г.

**ЗАДАНИЕ**

**к курсовому проекту**

студенту группы БИб-21Э1 Чигареву Александру Юрьевичу

1. Тема курсовому проекту: «Разработка приложения для БД система учета финансовых операций для банковской системы»
2. Содержание пояснительной записки:

* титульный лист;
* задание к курсовому проекту;
* содержание;
* введение;
* …
* заключение;
* список использованных источников.

1. Срок сдачи работы: «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

Задание выдано «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

Руководитель работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Н.А. Сапрыкина

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Е.В. Толкачева

(подпись преподавателя)

Задание к исполнению принял «\_\_\_» \_\_\_\_\_\_\_\_\_\_ 2023 г.

Студент \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ А. Ю. Чигарев

(подпись студента)

**Содержание**

[**Введение** 4](#_Toc135515462)

[**1. Обоснование метода стека технологии.** 5](#_Toc135515463)

[**1.1 Инструменты для работы с БД.** 5](#_Toc135515464)

[**1.2 Инструменты для разработки пользовательского интерфейса.** 5](#_Toc135515465)

[**2. Реализация приложения.** 5](#_Toc135515466)

[**2.1 Структура БД и реализация запросов.** 5](#_Toc135515467)

[**2.2 Пользовательский интерфейс.** 5](#_Toc135515468)

[**2.3 Результаты тестирования.** 5](#_Toc135515469)

[**Вывод** 6](#_Toc135515470)

[**Библиографический список** 7](#_Toc135515471)

# **Введение**

Разработка приложения для учета финансовых операций в банковской системе является актуальной задачей в информационной технологии. В данном курсовом проекте будет использован Microsoft Access в качестве хранилища базы данных, а Python будет выбран в качестве основной среды для реализации проекта. Для создания графического интерфейса будет использована библиотека PyQt5, которая была рекомендована научным руководителем.

Реализация проекта на основе Access, Python и PyQt5 позволит создать удобный и функциональный пользовательский интерфейс, а также обеспечить надежное хранение и обработку данных.

**Цель работы:** Повышение оперативности обработки информации в БД система учета финансовых операций для банковской системы

Для достижения следующей цели необходимо решить следующие **задачи:**

* + - 1. Обосновать выбор стека технологий разработки.
      2. Определить структуру данных для работы с файловой базой данных.
      3. Сформулировать запросы на выборку и сортировку данных из БД.
      4. Создать приложение для с графическим интерфейсом для работы с данными из БД.

# **1. Обоснование метода стека технологии.**

В данной работе были использованы следующие технологии:

1. Python – язык программирования для создания приложения;

2. Microsoft Access – БД для хранения, обработки и организации данных;

3. PyQt5 – средство для создания визуального интерфейса программы.

Данный выбор обусловлен тем что:

Python - мощный и популярный язык программирования, который обладает широким спектром библиотек и фреймворков для работы с базами данных. Использование Python позволяет разработчикам эффективно обрабатывать данные, выполнять запросы к базе данных и управлять ее структурой.

Microsoft Access - база данных, предоставляемая Microsoft, которая обладает простым в использовании интерфейсом и широкими возможностями для хранения, организации и обработки данных. Access хорошо интегрируется с языком программирования Python, что позволяет легко обращаться к базе данных и выполнять операции чтения и записи данных.

PyQt5 - библиотека для разработки графического интерфейса пользователя на основе Qt. Она предоставляет гибкие инструменты для создания современных и функциональных пользовательских интерфейсов. Использование PyQt5 позволяет разработчикам создавать интуитивно понятные и привлекательные интерфейсы для взаимодействия с базой данных и обеспечивает высокую гибкость в настройке внешнего вида и функциональности. Сочетание Python, Access и PyQt5 обеспечивает разработчикам удобную и эффективную среду для работы с базой данных. Python предоставляет мощные инструменты для обработки данных, Access - удобное хранилище данных, а PyQt5 - возможности создания привлекательного пользовательского интерфейса.

Основываясь на всём этом, было решено остановится на данных вариантах, и такой комбинации программы, языка и библиотеки.

## **1.1 Инструменты для работы с БД.**

В данной работе для работы с БД была использована такая программа как Microsoft Access.

Microsoft Access - это реляционная система управления базами данных (СУБД), разработанная корпорацией Microsoft. Она предоставляет средства для создания и управления базами данных, хранения и обработки данных, а также создания пользовательского интерфейса для работы с данными.

Основные функциональные возможности Microsoft Access включают:

1. Создание таблиц: Access позволяет создавать таблицы для хранения данных. Вы можете определить поля и их типы данных, настроить ограничения целостности и установить связи между таблицами.
2. Формы: С помощью Access вы можете создавать пользовательские формы, которые облегчают ввод, редактирование и просмотр данных. Формы позволяют настраивать внешний вид и поведение элементов интерфейса.
3. Запросы: Access предоставляет возможность создавать запросы для выборки, фильтрации и анализа данных в базе данных. Запросы могут включать условия, сортировку, объединение таблиц и вычисляемые поля.
4. Отчеты: С помощью Access можно создавать отчеты, которые представляют данные в удобном и информативном виде. Отчеты позволяют форматировать данные, добавлять графики и диаграммы, а также применять расчеты и агрегатные функции.
5. Макросы и модули VBA: Access поддерживает создание макросов и программирование на языке VBA (Visual Basic for Applications). Это позволяет автоматизировать операции, создавать пользовательские функции и настраивать поведение базы данных.
6. Импорт и экспорт данных: Access позволяет импортировать данные из различных источников, таких как Excel, текстовые файлы и другие базы данных. Также можно экспортировать данные в различные форматы, включая Excel, текстовые файлы и PDF.
7. Безопасность и управление доступом: Access обеспечивает возможность управления доступом к базе данных, устанавливать права и ограничения для пользователей и групп пользователей.

Для соединения Access с Python в данной работе использовался модуль pyodbc. pyodbc - это открытая библиотека Python, которая предоставляет доступ к базам данных через ODBC (Open Database Connectivity).

Установка параметров соединения в самом коде происходит посредством следующей команды:

conn\_str = f'DRIVER={Microsoft Access Driver (\*.mdb, \*.accdb)};DBQ={«путь к файлу»}'

conn = pyodbc.connect(conn\_str)

Посредством этих команд мы подключаемся к драйверу Microsoft Access Driver (\*.mdb, \*.accdb), и по пути указанном далее открываем БД.

Данные при соединении с БД не хранятся полностью в памяти. Вместо этого ODBC предоставляет стандартный интерфейс для взаимодействия с различными типами БД.

Когда выполняется запрос к БД, данные из базы передаются в Python по мере необходимости. Это позволяет работать с большими объёмами данных, не требуя больших объёмов оперативной памяти.

Полученные данные обычно хранятся в специальном объекте cursor, который представляет собой указатель на текущую позицию в наборе данных. По мере обращения к данным, курсор перемещается по набору данных, загружая только необходимые строки в память.

Данные в курсоре хранятся в том формате, который соответствует типам данных, определённым в БД. Курсор же возвращает данные в виде итерируемого объекта, который можно перебирать или получать доступ к его элементам, обычно это кортеж, где каждый кортеж представляет собой одну строку из результирующего набора. Каждый элемент кортежа соответствует полю или столбцу данных в результирующем наборе.

## **1.2 Инструменты для разработки пользовательского интерфейса.**

Для работы с графическим интерфейсом был использован популярный набор инструментов для разработки графического интерфейса пользователя (GUI) в языке программирования Python – PyQt5.

Он предоставляет обертку над библиотекой Qt, которая является одной из самых мощных и широко используемых библиотек для создания GUI-приложений.

Вот несколько основных характеристик и преимуществ PyQt5:

1. Кроссплатформенность: PyQt5 поддерживает разработку приложений, которые могут работать на различных операционных системах, включая Windows, macOS, Linux и другие.
2. Богатые возможности: PyQt5 обладает широким спектром функций и возможностей для разработки современных и профессионально выглядящих GUI-приложений. Он включает в себя множество виджетов (кнопки, поля ввода, таблицы и другие), графические элементы, возможности работы с мультимедиа, базами данных, сетевым взаимодействием и многое другое.
3. Простота использования: PyQt5 предоставляет простой и интуитивно понятный интерфейс для создания GUI-приложений. Он имеет хорошо документированный API и множество примеров, что делает процесс разработки более эффективным и быстрым.
4. Поддержка Qt Designer: Qt Designer - это инструмент, который позволяет визуально создавать пользовательский интерфейс для PyQt5. С его помощью вы можете создавать и настраивать виджеты и размещать их на формах без необходимости вручную писать код. Qt Designer интегрирован в PyQt5, что облегчает создание интерфейса вашего приложения.
5. Активное сообщество: PyQt5 имеет большое и активное сообщество разработчиков, которое предлагает поддержку, обновления, документацию, руководства и ответы на вопросы. Это означает, что вы можете найти множество ресурсов и помощи, если у вас возникают трудности или вам нужна дополнительная информация.

PyQt5 предоставляет мощные инструменты для создания профессиональных и функциональных GUI-приложений на языке Python. Он является популярным выбором для разработчиков, которые ищут гибкое, мощное и кроссплатформенное решение для создания пользовательского интерфейса своих программ.

В PyQt5 виджеты являются основными элементами, из которых состоит пользовательский интерфейс. Они представляют собой различные элементы управления, такие как кнопки, поля ввода, таблицы, метки и многое другое. Каждый виджет имеет свой уникальный внешний вид и выполняет определенную функцию. Вот некоторые распространенные виджеты, их назначение и основные свойства, методы и сигналы:

1. QPushButton (кнопка):

Назначение: Создание интерактивных кнопок.

Основные свойства: text (текст на кнопке), icon (иконка на кнопке), enabled (доступность кнопки).

Основные методы: setText(), setIcon(), setEnabled().

Основные сигналы: clicked() (сигнализирует о нажатии на кнопку).

2. QLineEdit (поле ввода):

Назначение: Получение текстового ввода от пользователя.

Основные свойства: text (текст в поле ввода), placeholderText (подсказывающий текст), enabled (доступность поля ввода).

Основные методы: setText(), setPlaceholderText(), setEnabled().

Основные сигналы: textChanged() (сигнализирует об изменении текста в поле ввода).

3. QLabel (метка):

Назначение: Отображение текста или изображения.

Основные свойства: text (текст метки), pixmap (изображение метки), alignment (выравнивание текста), enabled (доступность метки).

Основные методы: setText(), setPixmap(), setAlignment(), setEnabled().

4. QTableView (таблица):

Назначение: Отображение и редактирование табличных данных.

Основные свойства: model (модель данных для таблицы), selectionModel (модель выделения элементов), enabled (доступность таблицы).

Основные методы: setModel(), setSelectionModel(), setEnabled().

Основные сигналы: clicked() (сигнализирует о нажатии на элемент таблицы).

5. QComboBox (выпадающий список):

Назначение: Выбор одного элемента из списка.

Основные свойства: currentText (текущий выбранный элемент), currentIndex (индекс текущего элемента), enabled (доступность списка).

Основные методы: setCurrentText(), setCurrentIndex(), setEnabled().

Основные сигналы: currentTextChanged() (сигнализирует об изменении выбранного элемента).

6. QCheckBox (флажок):

Назначение: Позволяет пользователю выбрать или снять флажок.

Основные свойства: text (текст флажка), isChecked (состояние флажка), enabled (доступность флажка).

Основные методы: setText(), setChecked(), setEnabled().

Основные сигналы: stateChanged() (сигнализирует об изменении состояния флажка).

Это лишь некоторые примеры виджетов, доступных в PyQt5. Каждый виджет имеет свои дополнительные свойства, методы и сигналы, которые можно использовать для настройки и управления ими.

В PyQt5 для верстки виджетов используется система компоновки (layout system), которая позволяет располагать виджеты на форме или окне в удобном и гибком способе. Существуют различные типы компоновщиков в PyQt5, такие как QVBoxLayout, QHBoxLayout, QGridLayout и другие, которые позволяют управлять расположением и размерами виджетов.

Например, для создания компоновщика вертикального расположения используются следующие команды (Листинг 1):

Листинг 1 - Создания компоновщика вертикального расположения

vbox = QVBoxLayout()

window.setLayout(vbox)

Например, для добавления компоновщика горизонтального расположения внутрь компоновщика вертикального расположения используются следующие команды (Листинг 2):

Листинг 2 - Создания компоновщика горизонтального расположения

hbox = QHBoxLayout()

vbox.addLayout(hbox)

Это лишь пример, и в PyQt5 есть и другие методы, и свойства, которые позволяют более сложные и гибкие компоновки виджетов.

В итоге в данном разделе был выбран стек технологий, включающий Python, PyQt5 и Access. Python был выбран в качестве основного языка программирования, PyQt5 был выбран для создания графического интерфейса, Access был выбран в качестве файловой базы данных, так как он обладает удобным интерфейсом и хорошей совместимостью с Python.

Для работы с файловой базой данных Access была выбрана структура данных, основанная на таблицах. Таблицы в базе данных могут быть созданы с использованием Access и содержать различные поля. Структура данных также включает связи между таблицами, индексы и ограничения целостности для обеспечения целостности данных.

# **2. Реализация приложения.**

## **2.1 Структура БД и реализация запросов.**

Схема данных представляет собой описание структуры и организации данных в базе данных. Она определяет таблицы, поля, связи между таблицами и ограничения, обеспечивая целостность и согласованность данных в системе. Схема данных играет ключевую роль в обеспечении эффективного хранения, доступа и управления информацией, а также обеспечивает основу для разработки и выполнения SQL-запросов.

В данной курсовой работе используется схема данных которая представлена на рисунке 1.

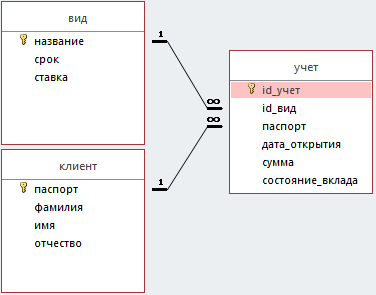


Рисунок 1 – Схема данных БД

SQL (Structured Query Language) является стандартным языком программирования, используемым для работы с базами данных. Он предоставляет набор команд и операторов для создания, изменения и извлечения данных, а также для управления структурой и безопасностью базы данных. SQL играет ключевую роль в обработке и анализе данных, позволяя эффективно выполнять различные операции, такие как поиск, сортировка, фильтрация и агрегация данных, обеспечивая гибкость и мощные возможности для работы с базами данных.

Используя язык SQL для БД были написаны следующие запросы:

1. найти наибольшую сумму вклада: SELECT MAX(сумма) FROM учет;

2. найти наименьшую сумму вклада: SELECT MIN(сумма) FROM учет;

3. найти вклады за определённый период: SELECT \* FROM учет WHERE дата\_открытия Between ? And ?;

4. найти вклады по номеру паспорта человека: SELECT \* FROM учет WHERE паспорт = ?;

5. упорядочить вкладчиков по возрастанию суммы: SELECT \* FROM учет ORDER BY сумма ASC;

6. упорядочить вкладчиков по убыванию суммы: SELECT \* FROM учет ORDER BY сумма DESC;

7. посчитать общую сумму вкладов: SELECT SUM(сумма) FROM учет;

8. посчитать сумму вкладов за определённый период: SELECT SUM(сумма) FROM учет WHERE дата\_открытия Between ? And ?;

9. посчитать сумму вкладов по определённому виду вклада: SELECT SUM(сумма) FROM учет WHERE id\_вид = ?;

10. посчитать средний размер вклада: SELECT AVG(сумма) FROM учет;

## **2.2 Пользовательский интерфейс**

Данное визуальное оформление, которое можно увидеть на рисунке 2, лишь бета версия визуально графического интерфейса, в дальнейшем он будет доработан и улучшен. Но на данный момент в этом графическом интерфейсе находится 10 кнопок для выполнения заранее продуманных SQL запросов. Нумерация кнопок совпадает с нумерацией запросов в данном списке:

1. найти наибольшую сумму вклада

2. найти наименьшую сумму вклада

3. найти вклады за определённый период

4. найти вклады по номеру паспорта человека

5. упорядочить вкладчиков по возрастанию суммы

6. упорядочить вкладчиков по убыванию суммы

7. посчитать общую сумму вкладов

8. посчитать сумму вкладов за определённый период

9. посчитать сумму вкладов по определённому виду вклада

10. посчитать средний размер вклада

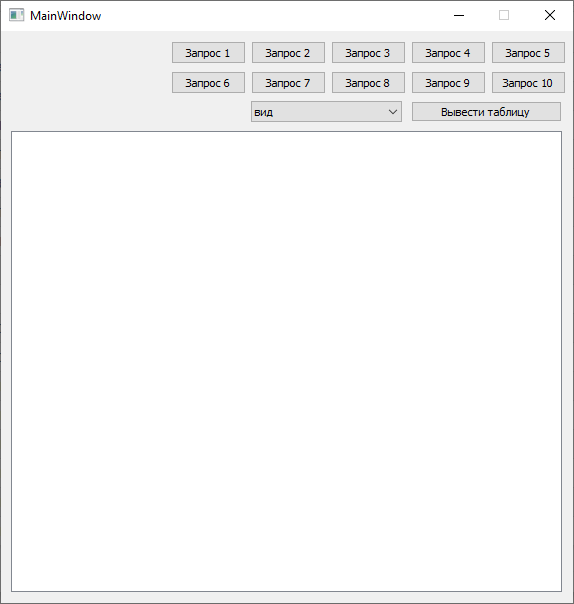


Рисунок 2 – Графический интерфейс

В данном интерфейсе присутствует выпадающее меню, которое содержит в себе название таблиц, которые содержатся в БД Access. Вывод таблиц в окно Table Widget производится по нажатию кнопки «Вывести таблицу».

На рисунках 3, 4, 5 можно увидеть, как выглядит данный вывод таблиц.

|  |  |
| --- | --- |
|  |  |
| а) | б) |
|  | |
| в) | |

Рисунок 3 – Вовод таблиц: а) «Вид»; б) «Клиент»; в) «Учет»

Кнопки в данном интерфейсе создаются следующим котом (листинг 3):

Листинг 3 – Создание кнопок

self.t4 = QtWidgets.QPushButton(self.centralwidget)

self.t4.setGeometry(QtCore.QRect(410, 10, 75, 23))

self.t4.setObjectName("t4")

Виджет таблицы создаётся следующим образом (листинг 4):

Листинг 4 – Создание области вывода таблиц

self.tableWidget.setObjectName("tableWidget")

self.tableWidget.setColumnCount(0)

self.tableWidget.setRowCount(0)

Выпадающее меню со списком таблиц создаётся следующим образом (листинг 5):

Листинг 5 – Создание выпадающего окна

self.comboBox = QtWidgets.QComboBox(self.centralwidget)

self.comboBox.setGeometry(QtCore.QRect(250, 70, 151, 21))

self.comboBox.setObjectName("comboBox")

Все таблицы из базы данных Access выводятся по общему шаблону кода (листинг 6)

Листинг 6 – Вывод таблицы из базы данных Access

cursor.execute(query)

result = cursor.fetchall()

self.ui.tableWidget.setRowCount(0)

self.ui.tableWidget.setColumnCount(0)

self.ui.tableWidget.setColumnCount(len(result[0]))

for row in range(len(result[0])):

self.ui.tableWidget.horizontalHeader().setSectionResizeMode(row, QHeaderView.ResizeToContents)

self.ui.tableWidget.setRowCount(len(result))

for i, row in enumerate(result):

for j, cell in enumerate(row):

item = QTableWidgetItem(str(cell))

self.ui.tableWidget.setItem(i, j, item)

В данном коде переменная query передаёт запрос курсору, после чего курсор возвращает значение в переменную result, после чего происходит сброс размера таблицы, если ранее уже отображалась таблица, после установка количества столбцов и строк, в итоге в последнем for происходит вывод значений.

## **2.3 Результаты тестирования.**

Далее на следующих снимках экрана можно увидеть результат выполнения запросов. Которые были заранее придуманы на языке SQL и добавлены виде временного решения выводящиеся через MessageBox.

|  |  |  |
| --- | --- | --- |
|  |  |  |
| a) | б) | в) |
|  |  |  |
| г) | д) | е) |

Рисунок 4 – Результаты запросов: а) результат запроса 1; б) результат запроса 2; в) результат запроса 7; г) результат запроса 8; д) результат запроса 9; е) результат запроса 10.

Все всплывающие окна с результатами запросов указанные на рисунке 4, выводятся по одному шаблонному коду вывода всплывающих окон (листинг 7)

Листинг 7 – Вывод всплывающего окна результатов

cursor.execute(query)

result = cursor.fetchone()[0]

message\_box = QMessageBox()

message\_box.setText(str(result))

message\_box.exec\_()

В данном коде переменная query передаёт запрос курсору, после чего курсор возвращает значение в переменную result, а уже после в созданное окно через функцию setText выводит нужное значение. В данном коде функция exec\_() нужна для выполнения закрытия всплывающего окна.

|  |  |
| --- | --- |
|  |  |
| а) | б) |
|  |  |
| в) | г) |

Рисунок 5 – Результаты запросов: а) результат запроса 3; б) результат запроса 4; в) результат запроса 5; г) результат запроса 6;

Все таблицы результатов выводятся по общему шаблону кода (листинг 8)

Листинг 8 – Вывод таблицы результатов запроса

cursor.execute(query)

result = cursor.fetchall()

self.ui.tableWidget.setRowCount(0)

self.ui.tableWidget.setColumnCount(0)

self.ui.tableWidget.setColumnCount(len(result[0]))

for row in range(len(result[0])):

self.ui.tableWidget.horizontalHeader().setSectionResizeMode(row, QHeaderView.ResizeToContents)

self.ui.tableWidget.setRowCount(len(result))

for i, row in enumerate(result):

for j, cell in enumerate(row):

item = QTableWidgetItem(str(cell))

self.ui.tableWidget.setItem(i, j, item)

В данном коде переменная query передаёт запрос курсору, после чего курсор возвращает значение в переменную result, после чего происходит сброс размера таблицы, если ранее уже отображалась таблица, после установка количества столбцов и строк, в итоге в последнем for происходит вывод значений.

В итоге с помощью языка SQL в данном проекте были указаны условия фильтрации и сортировки для получения необходимых данных. Python обеспечивает поддержку работы с базами данных через соответствующие библиотеки, что позволило формулировать запросы на выборку и сортировку данных из базы данных Access.

Для создания приложения с графическим интерфейсом для работы с данными из базы данных Access был использован PyQt5, который обеспечил возможность разработки понятного интерфейса.

**Заключение**

В данном проекте был выбран стек технологий, включающий Python, PyQt5 и Access. Python был выбран в качестве основного языка программирования, предоставляющего широкие возможности по работе с базами данных, а также графическими элементами. PyQt5 был выбран для создания графического интерфейса. Access был выбран в качестве файловой базы данных благодаря его удобному интерфейсу и совместимости с Python.

Для работы с файловой базой данных Access была определена структура данных на основе таблиц. SQL-запросы используются для выборки и сортировки данных из базы данных. Создание приложения с графическим интерфейсом для работы с данными из базы данных Access было реализовано с помощью PyQt5, который предоставляет средства для создания удобного и интуитивно понятного интерфейса, а Python обеспечивает взаимодействие с базой данных Access, позволяя получать, обрабатывать и отображать данные.

Таким образом, выбранный стек технологий в сочетании с определенной структурой данных и использованием SQL-запросов позволил эффективно разработать приложение для работы с данными из базы данных Access.

# **Библиографический список**

1. Основные сведения о базах данных [Электронный ресурс] – URL: https://support.microsoft.com/ru-ru/office/основные-сведения-о-базах-данных-a849ac16-07c7-4a31-9948-3c8c94a7c204 (Дата обращения)

2. Microsoft Access для чайников – что это за программа и как ей пользоваться [Электронный ресурс] – URL: http://composs.ru/access-chto-eto-za-programma/ (Дата обращения)

3. Python 3.11.3 documentation [Электронный ресурс] – URL: https://docs.python.org/3/ (Дата обращения)

4. Руководство по PyQt5 [Электронный ресурс] – URL: https://python-scripts.com/pyqt5 (Дата обращения)

5. A Quick Start to Qt Designer [Электронный ресурс] – URL: https://doc.qt.io/qt-5/designer-quick-start.html (Дата обращения)

6. Общие сведения о программировании в Access [Электронный ресурс] – URL: https://support.microsoft.com/ru-ru/office/общие-сведения-о-программировании-в-access-92eb616b-3204-4121-9277-70649e33be4f (Дата обращения)

7. QMainWindow Class [Электронный ресурс] – URL: https://doc.qt.io/qt-6/qmainwindow.html (Дата обращения)

8. QTableWidgetItem Class [Электронный ресурс] – URL: https://doc.qt.io/qt-6/qtablewidgetitem.html (Дата обращения)

9. QHeaderView Class [Электронный ресурс] – URL: https://doc.qt.io/qt-6/qheaderview.html (Дата обращения)

10. QMessageBox Class [Электронный ресурс] – URL: https://doc.qt.io/qt-6/qmessagebox.html (Дата обращения)

# **Приложение**

import pyodbc

import sys

from PyQt5.QtWidgets import QApplication, QMainWindow, QTableWidgetItem, QHeaderView, QMessageBox

from test\_wind import Ui\_MainWindow

try:

conn = pyodbc.connect(r'DRIVER={Microsoft Access Driver (\*.mdb, \*.accdb)};DBQ=D:\prog\project\pycharm\pp\kursach\kursach\_timp\_bd.accdb')

cursor = conn.cursor()

except pyodbc.Error as e:

print("Error: ", e)

class MyWindow(QMainWindow):

def \_\_init\_\_(self):

super(MyWindow, self).\_\_init\_\_()

self.ui = Ui\_MainWindow()

self.ui.setupUi(self)

t = cursor.tables(tableType="TABLE")

table\_names = [table.table\_name for table in t]

self.ui.comboBox.addItems(table\_names)

self.ui.returnTable.clicked.connect(self.table\_def)

self.ui.t1.clicked.connect(self.task1)

self.ui.t2.clicked.connect(self.task2)

self.ui.t3.clicked.connect(self.task3)

self.ui.t4.clicked.connect(self.task4)

self.ui.t5.clicked.connect(self.task5)

self.ui.t6.clicked.connect(self.task6)

self.ui.t7.clicked.connect(self.task7)

self.ui.t8.clicked.connect(self.task8)

self.ui.t9.clicked.connect(self.task9)

self.ui.t10.clicked.connect(self.task10)

def table\_def(self):

self.ui.tableWidget.setRowCount(0)

self.ui.tableWidget.setColumnCount(0)

val = self.ui.comboBox.currentText()

tables = cursor.tables(tableType="TABLE")

for table in tables:

if val == table.table\_name:

table\_cursor = conn.cursor()

table\_cursor.execute(f"SELECT \* FROM {table.table\_name}")

data = table\_cursor.fetchall()

self.ui.tableWidget.setColumnCount(len(data[0]))

self.ui.tableWidget.setHorizontalHeaderLabels(["название", "срок", "ставка"])

for row in range(len(data[0])):

self.ui.tableWidget.horizontalHeader().setSectionResizeMode(row, QHeaderView.ResizeToContents)

self.ui.tableWidget.setRowCount(len(data))

for i, row in enumerate(data):

for j, cell in enumerate(row):

item = QTableWidgetItem(str(cell))

self.ui.tableWidget.setItem(i, j, item)

def task1(self):

query = "SELECT MAX(сумма) FROM учет;"

cursor.execute(query)

result = cursor.fetchone()[0]

message\_box = QMessageBox()

message\_box.setText(str(result))

message\_box.exec\_()

# print(result)

def task2(self):

query = "SELECT MIN(сумма) FROM учет;"

cursor.execute(query)

result = cursor.fetchone()[0]

message\_box = QMessageBox()

message\_box.setText(str(result))

message\_box.exec\_()

# print(result)

def task3(self):

start\_date = '01.01.2023'

end\_date = '01.03.2023'

query = "SELECT \* FROM учет WHERE дата\_открытия Between ? And ?;"

cursor.execute(query, (start\_date, end\_date))

result = cursor.fetchall()

self.ui.tableWidget.setRowCount(0)

self.ui.tableWidget.setColumnCount(0)

self.ui.tableWidget.setColumnCount(len(result[0]))

for row in range(len(result[0])):

self.ui.tableWidget.horizontalHeader().setSectionResizeMode(row, QHeaderView.ResizeToContents)

self.ui.tableWidget.setRowCount(len(result))

for i, row in enumerate(result):

for j, cell in enumerate(row):

item = QTableWidgetItem(str(cell))

self.ui.tableWidget.setItem(i, j, item)

# print(result)

def task4(self):

passport\_number = '9245546743'

query = "SELECT \* FROM учет WHERE паспорт = ?;"

cursor.execute(query, (passport\_number,))

result = cursor.fetchall()

self.ui.tableWidget.setRowCount(0)

self.ui.tableWidget.setColumnCount(0)

self.ui.tableWidget.setColumnCount(len(result[0]))

for row in range(len(result[0])):

self.ui.tableWidget.horizontalHeader().setSectionResizeMode(row, QHeaderView.ResizeToContents)

self.ui.tableWidget.setRowCount(len(result))

for i, row in enumerate(result):

for j, cell in enumerate(row):

item = QTableWidgetItem(str(cell))

self.ui.tableWidget.setItem(i, j, item)

# print(result)

def task5(self):

query = "SELECT \* FROM учет ORDER BY сумма ASC;"

cursor.execute(query)

result = cursor.fetchall()

self.ui.tableWidget.setRowCount(0)

self.ui.tableWidget.setColumnCount(0)

self.ui.tableWidget.setColumnCount(len(result[0]))

for row in range(len(result[0])):

self.ui.tableWidget.horizontalHeader().setSectionResizeMode(row, QHeaderView.ResizeToContents)

self.ui.tableWidget.setRowCount(len(result))

for i, row in enumerate(result):

for j, cell in enumerate(row):

item = QTableWidgetItem(str(cell))

self.ui.tableWidget.setItem(i, j, item)

# print(result)

def task6(self):

query = "SELECT \* FROM учет ORDER BY сумма DESC;"

cursor.execute(query)

result = cursor.fetchall()

self.ui.tableWidget.setRowCount(0)

self.ui.tableWidget.setColumnCount(0)

self.ui.tableWidget.setColumnCount(len(result[0]))

for row in range(len(result[0])):

self.ui.tableWidget.horizontalHeader().setSectionResizeMode(row, QHeaderView.ResizeToContents)

self.ui.tableWidget.setRowCount(len(result))

for i, row in enumerate(result):

for j, cell in enumerate(row):

item = QTableWidgetItem(str(cell))

self.ui.tableWidget.setItem(i, j, item)

# print(result)

def task7(self):

query = "SELECT SUM(сумма) FROM учет;"

cursor.execute(query)

result = cursor.fetchone()[0]

message\_box = QMessageBox()

message\_box.setText(str(result))

message\_box.exec\_()

# print(result)

def task8(self):

start\_date = '01.01.2023'

end\_date = '01.03.2023'

query = "SELECT SUM(сумма) FROM учет WHERE дата\_открытия Between ? And ?;"

cursor.execute(query, (start\_date, end\_date))

result = cursor.fetchone()[0]

message\_box = QMessageBox()

message\_box.setText(str(result))

message\_box.exec\_()

# print(result)

def task9(self):

id\_wid = 'первый'

query = "SELECT SUM(сумма) FROM учет WHERE id\_вид = ?;"

cursor.execute(query, (id\_wid,))

result = cursor.fetchone()[0]

message\_box = QMessageBox()

message\_box.setText(str(result))

message\_box.exec\_()

# print(result)

def task10(self):

query = "SELECT AVG(сумма) FROM учет;"

cursor.execute(query)

result = cursor.fetchone()[0]

message\_box = QMessageBox()

message\_box.setText(str(result))

message\_box.exec\_()

# print(result)

if \_\_name\_\_ == '\_\_main\_\_':

app = QApplication(sys.argv)

window = MyWindow()

window.show()

sys.exit(app.exec\_())