Министерство науки и высшего образования Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ ИТМО
ITMO University

Отчет

По дисциплине Программирование

Тема работы Разработка программы для контроля собственных денежных средств с графическим интерфейсом

Обучающийся Новиков Николай Викторович

Факультет факультет инфокоммуникационных технологий

Группа К3123

Направление подготовки 11.03.02 Инфокоммуникационные технологии и системы связи

Образовательная программа Программирование в инфокоммуникационных системах

Обучающийся _____ (дата) _____ (подпись) Новиков Н. В. (Ф.И.О.)

Руководитель _____ (дата) _____ (подпись) Казанова П.П. (Ф.И.О.)

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Целью проекта было создание программы для контроля собственных денежных средств с графическим интерфейсом. Программа должна была содержать в себе такие функции как: Добавление трат, просмотр общего баланса, фильтрация по категории и цене.

Предварительно был проведен анализ предметной области и требований. Выяснилось, что наиболее схожим ПО является корзина в заметках, которой пользуются, в большинстве своем, люди, не понимающие программирования. Поэтому приоритетной целью всего проекта было создание простого и понятного приложения для корзины.

1. Создание проекта

1.1 Файловая структура проекта и используемые модули

Файловая структура проекта (Рисунок 1) представляет из себя несколько исполняемых файлов питона, главным из который является main.py, вместе с папкой для иконок, базой данных и файлами дизайна интерфейса

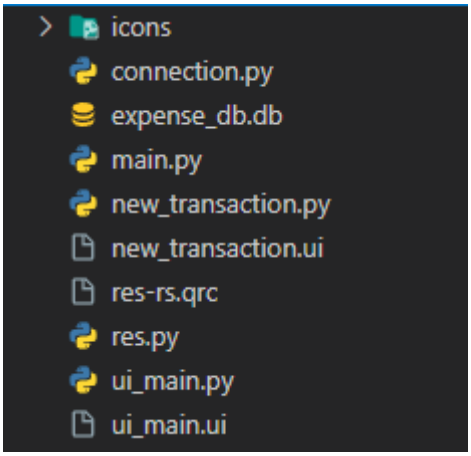

Рисунок 1 - Файловая структура проекта

1.2 Подключение модулей и файлов

Так как по сравнению с консольным приложением, в графическом приложении зависимостей и библиотек будет куда больше. В данной программе вся графическая часть была построена на основе библиотеки PySide6, которая имеет крайне большой спектр возможностей. В main.py, помимо PySide6, были подключены также и другие библиотеки (Рисунок 2), такие как sys для работы с оболочкой приложения. В качестве модулей выступил файл подключения к базе данных.

```
1  import sys
2
3  from PySide6 import QtWidgets
4  from PySide6.QtWidgets import QApplication, QMainWindow
5  from PySide6.QtSql import QSqlTableModel
6
7  from ui_main import Ui_MainWindow
8  from new_transaction import Ui_Dialog
9  from connection import Data
```

Рисунок 2 - Модули основного файла

1.3 Создание дизайна системы

Перед началом разработки было необходимо создать статичный дизайн программы для дальнейшей работы. В качестве графического редактора PySide6 имеет инструмент QtDesigner. В нем был создан, как главный экран приложения (Рисунок 3), так и модальное окно, в котором будет происходить добавление новых покупок (Рисунок 4).

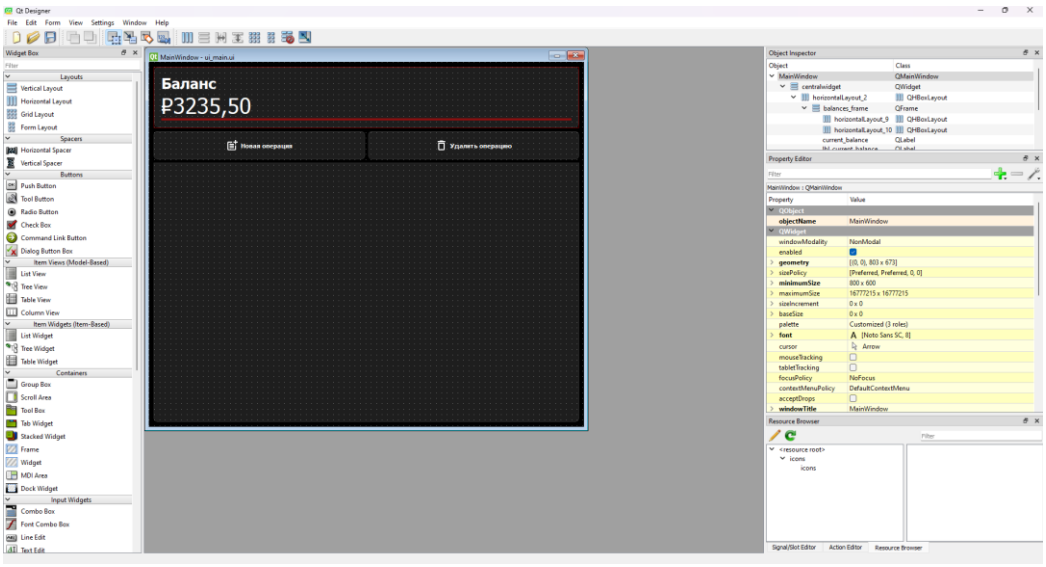

Рисунок 3 - Основной экран

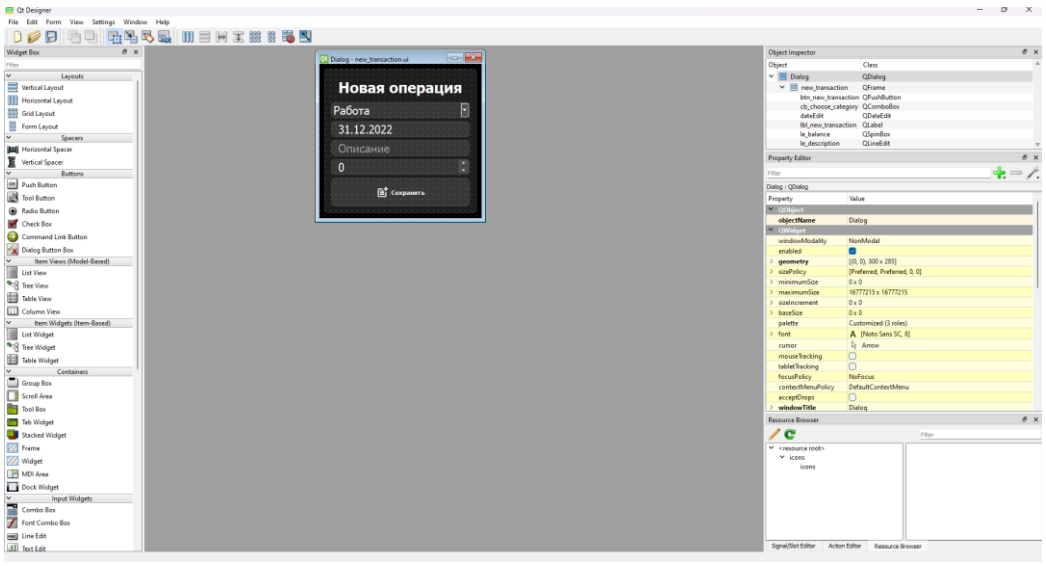

Рисунок 4 - модальное окно

В дальнейшем эти файлы, с помощью PySide6, были преобразованы в файлы с расширением .py (new_transaction.py и ui_main.py).

2. Функционал программы

2.1 Добавление товара в корзину.

Одной из основных функций программы является добавление продукта в корзину. Оно реализовано следующим образом (Рисунок 5).

```
34    def open_new_transaction_window(self):
35        self.new_window = QtWidgets.QDialog()
36        self.ui_window = Ui_Dialog()
37        self.ui_window.setupUi(self.new_window)
38        self.new_window.show()
39        sender = self.sender()
40        if sender.text() == "Новая операция":
41            self.ui_window.btn_new_transaction.clicked.connect(self.add_new_transaction)
42
43    def add_new_transaction(self):
44        date = self.ui_window.dateEdit.text()
45        category = self.ui_window.cb_choose_category.currentText()
46        description = self.ui_window.le_description.text()
47        balance = self.ui_window.le_balance.text()
48
49        self.conn.add_new_transaction_query(date, category, description, balance)
50        self.view_data()
51        self.reload_data()
52        self.new_window.close()
```

Рисунок 5 - Добавление нового товара

При нажатии на кнопку “Новая операция” вызывается открытие модального окна, в котором пользователь заполняет такие поля как: Категория товара, Дата приобретения, Описание товара и его стоимость. В стоимость пользователь не сможет ввести ничего, кроме числа, так как в QtDesigner для чисел есть специальное поле, которое ограничивает вводимые данные на уровне моделирования системы. После чего следуют стандартные функции обновления экрана и показа таблицы (Рисунок 6). Баланс также обновляется автоматически после каждого действия пользователя.

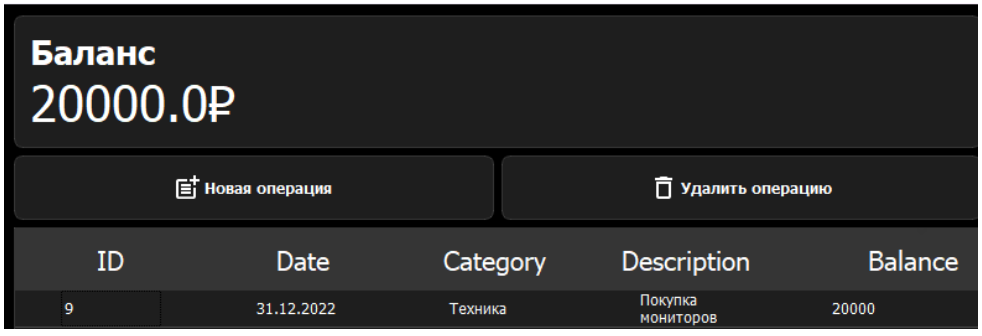

Рисунок 6 - Экран с введенными данными

2.2 Удаление продукта

Удаление продукта происходит при клике пользователя по нужной кнопке. Функция ищет в строке id данного продукта и вызывает функцию удаления товара на уровне базы данных.

```
54      def delete_current_transaction(self):
55          index = self.ui.tableView.selectedIndexes()[0]
56          id = str(self.ui.tableView.model().data(index))
57
58          self.conn.delete_transaction_query(id)
59          self.view_data()
60          self.reload_data()
```

Рисунок 7 - Функция удаления продукта

2.3 Фильтрация

В программе было необходимо также реализовать фильтрацию товара по категории и цене. Благодаря PySide6 данная задача стала крайне простой в реализации. При выводе полей к ним присоединялась функция сортировки по нужному полю. Условно нажав на кнопку “Balance”, можно отсортировать продукты по балансу в обе стороны, так же работает и для всех кнопок.

3. Работа с базой данных

3.1 Создание и подключение к базе данных

Все товары, которые были добавлены пользователем, должны где-то храниться. PySide6 предоставляет свой модуль для работы с базами данных. Алгоритм создания и подключения к базе данных представлен на Рисунке 8

```python
def create_connection(self):
    db = QtSql.QSqlDatabase.addDatabase('QSQLITE')
    db.setDatabaseName('expense_db.db')

    if not db.open():
        QtWidgets.QMessageBox.critical(None, "Cannot open database",
                                       "Click Cancel to exit.", QtWidgets.QMessageBox.Cancel)
        return False

    query = QtSql.QSqlQuery()
    query.exec("CREATE TABLE IF NOT EXISTS expenses (ID integer primary key AUTOINCREMENT, Date VARCHAR(20), "
               "Category VARCHAR(20), Description VARCHAR(20), Balance REAL)")
    return True
```

Рисунок 8 - Создание и подключение к БД

Здесь создается база данных с именем (expence_db.db), а после всевозможных проверок создается таблица с названием expenses со всеми полями из модального окна.

3.2 Добавление и удаление продукта на уровне БД

Как было выше сказано, то функция удаления и добавления хоть и вызывается в главном файле, но все вычисления и запросы выполняются непосредственно в файле с БД (Рисунок 9).

```python
35      def add_new_transaction_query(self, date, category, description, balance):
36          sql_query = "INSERT INTO expenses (Date, Category, Description, Balance) VALUES (?, ?, ?, ?)"
37          self.execute_query_with_params(sql_query, [date, category, description, balance])
38
39      def delete_transaction_query(self, id):
40          sql_query = "DELETE FROM expenses WHERE ID=?"
41          self.execute_query_with_params(sql_query, [id])
```

Рисунок 9 - Удаление и добавление товара

В функции передаются необходимые данные удаленного/добавленного товара и после на их основе составляется SQL запрос к базе данных.

4. Диаграммы и схемы

4.1 Диаграмма классов

В программе существует два класса – ExpenceTracker (основной класс всей программы) и Data (отвечает за взаимодействие с БД). Они являются независимыми друг от друга. Диаграмма классов представлена на Рисунке 10

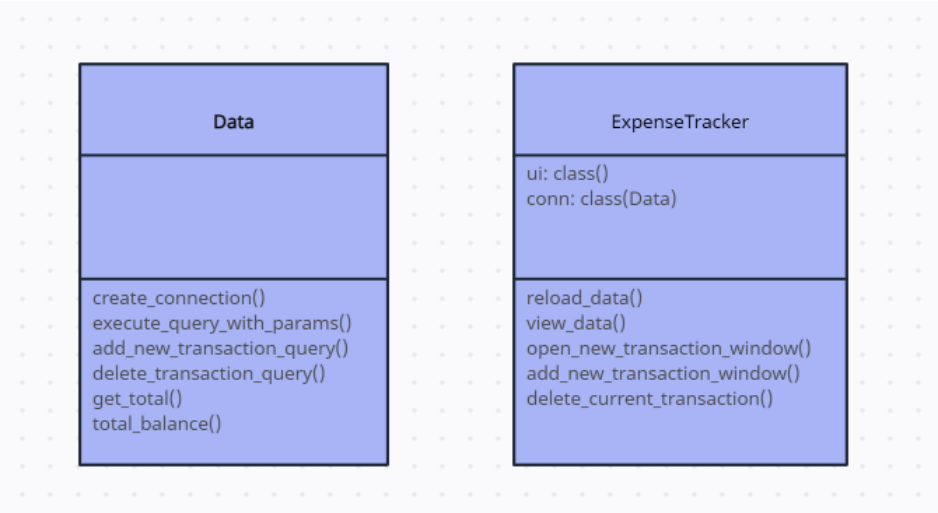

Рисунок 10 - диаграмма классов

4.2 Схема базы данных

База данных, используемая проекта, имеет в себе лишь одну таблицу, хранящую в себе данные о товаре, так как в данном приложении больше ничего и не надо. Схема базы данных представлена на Рисунке 11

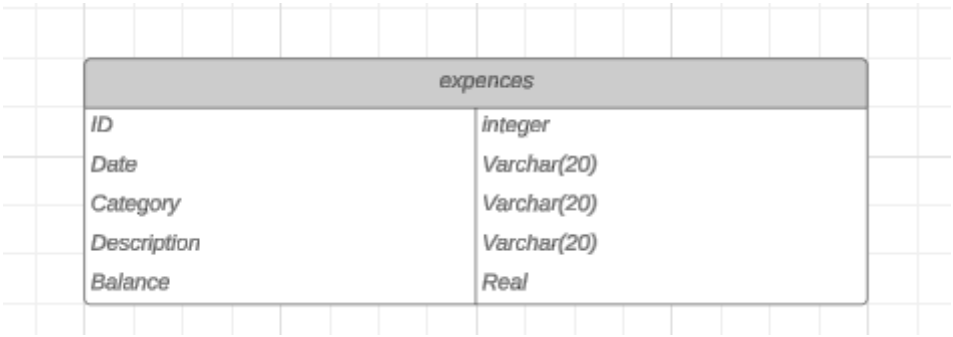

Рисунок 11 - Схема базы данных

ЗАКЛЮЧЕНИЕ

С помощью нескольких модулей python`a удалось создать программу для контроля собственных денежных средств с графическим интерфейсом. Были реализованы такие функции как: добавление продукта в корзину, просмотр всего записанного, просмотр по категории, распределение по стоимости, удаление добавленных записей.