Проект PyQT – «Доходы и расходы»

Основная информация

**Автор проекта:** Даниил Сюрмаченко

Главная задача проекта – это помочь людям начать контролировать свои доходы и расходы. Программа даёт возможность создать счёт, в котором пользователь может задать такие настройки: название счёта, валюта, цвет-маркер, которым будет помечаться счёт, а также начальную сумму. Добавлять транзакции (доход или расход) к определённому счёту. В транзакции можно дополнительно указать категорию транзакции (например, расход за транспорт) и дату. Счета и транзакции можно редактировать и удалять. Списки категорий для транзакций и валют для счетов поставляются по умолчанию и их изменить нельзя.

Реализация проекта

Проект был разделён на основных файла:

1. **/main.py** – главный файл, в котором реализована основная функциональность программы

* **Main** – Основной класс программы, наследованный от QMainWindow. Здесь происходит подключение к базе данных, создание меню, а также отображение главной страницы
* **MainPage** – Класс реализующий главную страницу программы, наследован от QWidget и MainPageUi (из main\_page.py). Главная страница отображает основную информацию о счетах, такую как: количество денег на счету, список последних доходов и расходов. Также можно создать транзакцию к выбранному счёту.
* **AddTransactionDialog** – Класс реализующий окно с формой создания транзакции, наследован от QDialog и AddTransactionDialogUi (из add\_transaction\_dialog.py). Диалоговое окно, в котором пользователь заполняет необходимую информацию для транзакции: количество денег (отрицательное число, если пользователь хочет записать расход или положительное число, если пользователь хочет записать доход), счёт, категория и дата.
* **EditTransactionDialog** – Класс реализующий окно с формой редактирования транзакции, наследован от AddTransactionDialog. Диалоговое окно, в котором можно изменить или удалить транзакцию.
* **ListAccountDialog** – Класс реализующий окно со списком счетов у пользователя, наследован от QDialog. Элемент списка показывает название и цвет-маркер счёта, а также кнопку для редактирования счёта.
* **CreateAccountDialog** – Класс реализующий окно с формой создания счёта, наследован от QDialog и CreateAccountDialogUi (из create\_account\_widget.py). Диалоговое окно, в котором пользователь заполняет необходимую информацию для счёта: название, валюта, цвет-маркер и начальная сумма на счету.
* **EditAccountDialog** – Класс реализующий окно с формой редактирования счёта, наследован от CreateAccountDialog. Диалоговое окно, в котором можно изменить или удалить (если у пользователя больше одного счёта) счёт.

1. **/core/database.py** – файл, в котором храниться классы, ориентированные на удобное использование данных из базы данных.
   * **Currency –** для валюты счёта.
   * **Account –** для счёта.
   * **Category –** для категории транзакции.
   * **Transaction –** для транзакции.
2. **/core/widgets.py** – файл хранит в себе модифицированные виджеты QT
   * **Tag** – класс реализующий тег, наследован от QLabel. Видоизменённый QLabel, в котором можно поставить фон в качестве одного цвета и в зависимости от контрастности фона – текст будет либо белым, либо чёрным. Используется для отображения дат, категорий и счетов.
   * **AccountInfo** – класс реализующий виджет с основной информацией о счёте, наследован от QWidget и ItemOfListWidgetUi (из item\_of\_list\_widget.py). Информация, которую показывает виджет: количество денег, валюта, название и цвет-маркер.
   * **TransactionInfo** – класс реализующий виджет с основной информацией о транзакции, наследован от QWidget и ItemOfListWidgetUi. Информация, которую показывает виджет: количество денег (расход помечается красным, доход - зелёным), дата, категория.
   * **TransactionInfoButton** – класс реализующий кнопку-виджет, которая вмешает в себя информацию о транзакции, наследован от QPushButton и TransactionInfo. Главная особенность данного виджета – при нажатии на этот виджет, пользователь откроет диалог с редактированием нажатой транзакции.
   * **List** – класс реализующий виджет-список, в котором можно вмещать другие виджеты, наследован от QScrollArea. Особенность виджета в том, что все элементы выравниваются в верх списка, а не заполняют всё пространство, а при переполнении виджетами списка будет показан ползунок.
   * **AccountItem** – класс реализующий виджет в качестве элемента списка счетов, наследован от QWidget. Виджет показывает название счёта и его цвет-марекер, а также кнопку для редактирования этого счёта.
   * **AccountList** – класс реализующий виджет-список, в котором вмещаются виджеты AccountItem, наследован от List. Список показывает текущие счета, которые есть у пользователя.
   * **TransactionList** – класс реализующий виджет-список, в котором вмещаются виджеты TransactionInfoButton, наследован от List. Список показывает транзакции, совершённые на счету.

В проекте есть дополнительные файлы:

1. В папке **/ui** хранятся ui файлы QtDesigner и результат их компиляции с помощью pyuic в виде файлов Python.
2. **/icon.ico** – иконка, которая стоит на каждом окне в программе
3. **/requerements.txt** – файл с необходимыми библиотеками
4. **/db.sqlite** – база данных, в котором хранится вся информация о кошельках, транзакциях, валютах и категориях.

Использованные технологии

В проекте были использованы:

* стандартные библиотеки Python (random, functools, copy)
* PyQT (библиотека для работы с GUI-интерфейсами)
* SQLite3 (база данных)

Необходимые библиотеки, чтобы запустить проект:

1. PyQt5 (5.15.1)
2. PyQt5-sip (12.8.1)
3. PyQt5-stubs (5.14.2.2)

Скриншоты проекта

