**РАЗРАБОТКА КРОССПЛАТФОРМЕННОГО ФИНАНСОВОГО ПРИЛОЖЕНИЯ ДЛЯ КОНТРОЛЯ ЛИЧНЫХ РАСХОДОВ**

В настоящее время остро стоит проблема распределения финансов. Многие люди часто сталкиваются с нехваткой денег, эмоциональными покупками, неспособностью распределить бюджет на месяц.

В связи с этим актуальной задачей является помочь отследить движение финансовых потоков каждому человеку. Поэтому идея разработки финансового приложения, позволяющего контролировать расходы, является актуальной.

Необходимо выявить – какие функции в приложении являются приоритетными для пользователей. Для этого были опрошены студенты ГУАПа трех направлений. На основании опроса были выделены приоритетные задачи: простой и понятный функционал, индивидуальный отчет по расходам в заданный период.

Особенность приложения – отслеживание всех видов трат и возможность группировки их по сферам, которые пользователь может определять самостоятельно.

Финансовое приложение по структуре состоит из трех слоев, представленных на рис. 1.

Презентационный слой

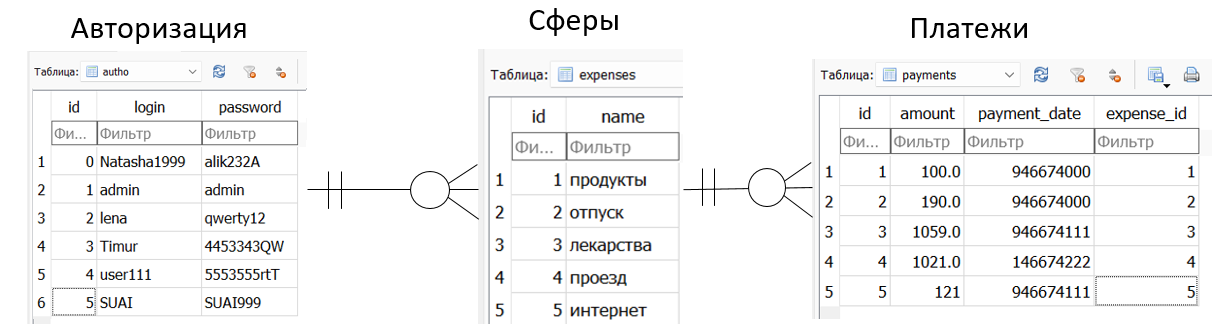
Бизнес-логика

Хранилище данных

Рис. 1. Структура приложения

Слой данный реализован через систему управления базами данных (СУБД) SQLite [1]. Это реляционная база данных (БД), совместимая с SQL.

База данных состоит из трех таблиц. На рис. 2 представлены таблицы базы данных авторизации - с логинами и паролями. Для каждого пользователя создается своя таблица со сферами расходов, которые вносит сам пользователь. Каждой сфере расходов соотносится таблица платежей по этой сфере, куда пользователь вносит самостоятельно сумму платежа и дату. Дата сохраняется как целое число – это число секунд с 1970-01-01 00:00:00 UTC.

  
Рис. 2. Схема данных БД

Презентационный слой реализован через PyQt [2] — это мультиплатформенная библиотека, которая работает на всех основных операционных системах, в том числе Unix, Windows и Mac OS.

Когда пользователь открывает приложение, ему необходимо войти в существующий аккаунт или создать новый. При регистрации отправляется SQL-запрос и в таблицу БД авторизации добавляется новый логин и пароль. При наличии такого логина в БД – пользователю выводится диалоговое окно с сообщением о невозможности использования логина. При попытке войти в существующий аккаунт производится проверка поля с паролем по совпадающему логину. После авторизации открывается главное окно приложения (рис. 3).

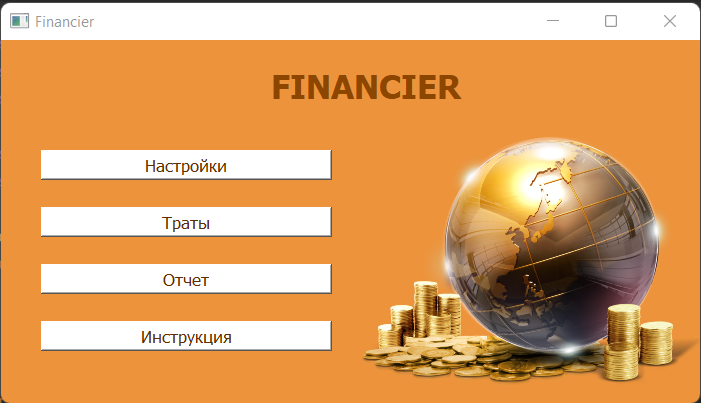
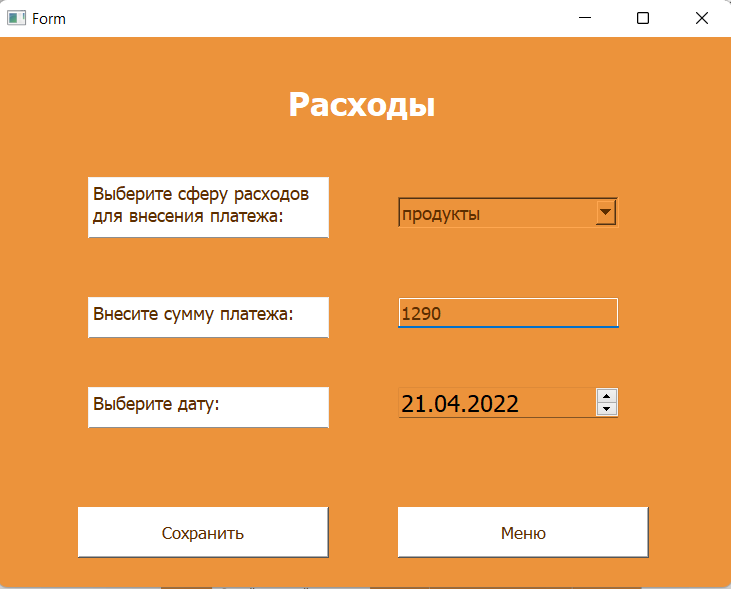
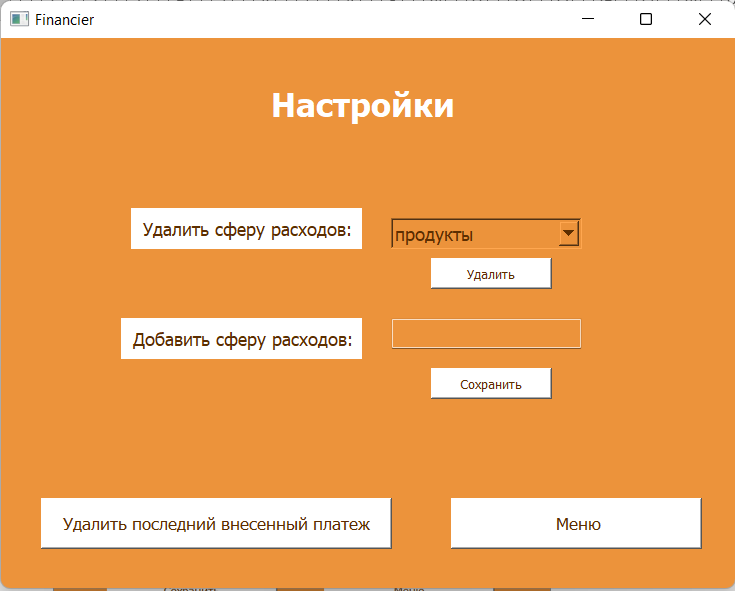


Рис. 3. Главное окно приложения

Слой бизнес-логики реализован на языке Python версии 3.8 с использованием множественных запросов в базу данных: добавление данных в таблицы пользователей, сфер и расходов; удаление данных из таблиц; фильтрация по датам и сферам расходов; и другая обработка данных из БД программными средствами.

В окне приложения «Расходы» необходимо внести сумму платежа, дату и выбрать сферу расходов (рис. 4). В окне «Настройки» возможно добавить индивидуальную сферу расходов, удалить ранее созданную сферу или платеж, записанный по ошибке (рис. 5).

|  |  |
| --- | --- |
| Рис. 4. Страница «Расходы» | Рис. 5. Страница «Настройки» |

В окне «Отчет» изначально в полях находятся данные по самому затратному дню и месяцу. Чтобы увидеть остальные данные пользователю необходимо нажать кнопку «Фильтровать» (рис. 6).



Рис. 6. Окно приложения «Отчет» после фильтрации

Пользователь вводит дату начала и конца периода для получения отчета. Дата по умолчанию помогает предотвратить ошибки. Период может быть задан как один день и как промежуток дат, но при введении даты конца периода меньше, чем начала – пользователю выводится диалоговое окно, представленное на рис. 7.

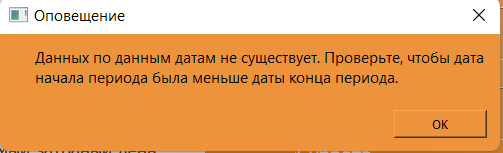


Рис. 7. Оповещение об ошибке

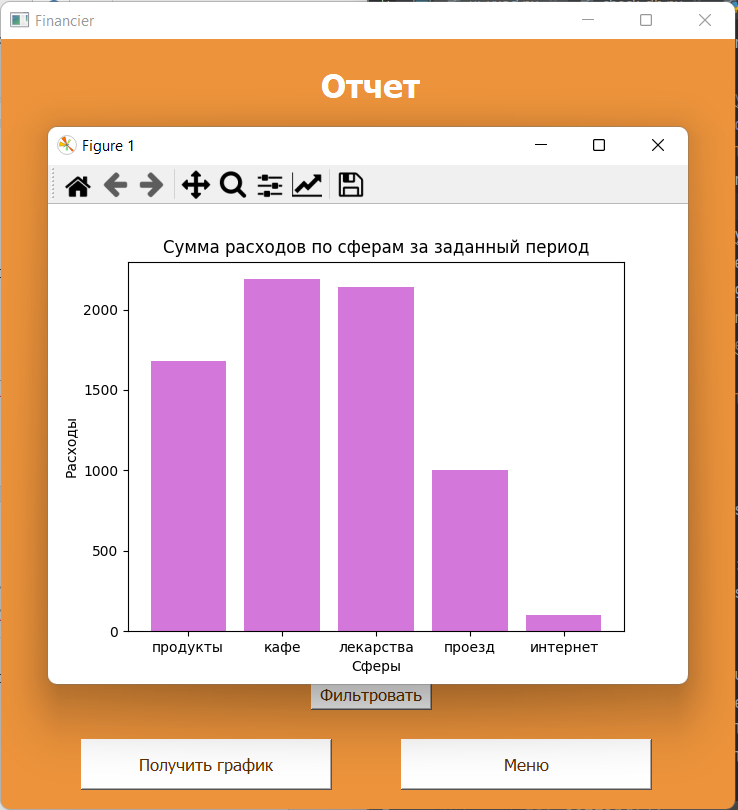
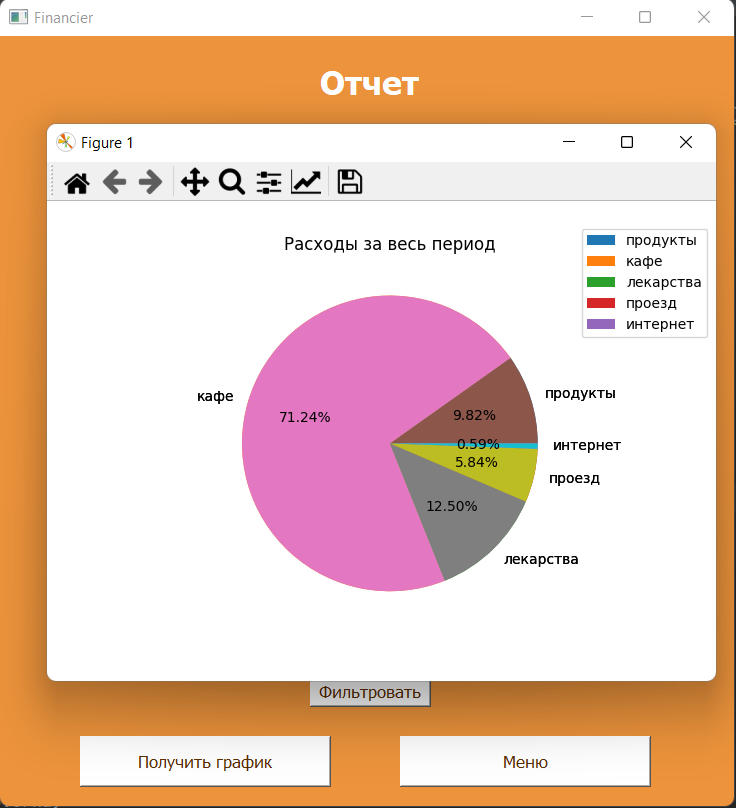
Изначально пользователю выводится информация о самом затратном дне и месяце за весь период. Для этого используется функция запроса к базе данных**, по которой мы подсчитываем сумму платежей по каждой дате. Далее находится самая большая сумма и ее дата. Остается конвертировать дату, записанную в БД в секундах, переводя ее в привычный формат и сравнить со списком всех дней недели. После этого происходит проверка корректности значения и только после – вывод информации пользователю, так как в период по умолчанию данных может не оказаться.** Функция для вывода наиболее затратного месяца за весь период работает аналогично, но вместо списка дней недели используется список месяцев.

После нажатия кнопки фильтрации выводятся данные о самой затратной сфере за период и общей сумме расходов по этой сфере. Для вывода этих данных сначала через SQL-запрос, в котором мы получаем все данные из таблицы платежей **и сфер по id, н**еобходимо получить все данные, находящиеся в промежутке дат между введенными. Далее по данным из прошлой функции подсчитывается сумма расходов по каждой сфере, сопоставленной по id.

По этим же данным SQL-запроса находится самая часто встречающаяся сфера. Подсчитывается количество платежей по каждой сфере в заданный период, пользователю выводится сфера с наибольшим значением.

Для нахождения средней суммы расходов подсчитывается количество сфер и общая сумма расходов по ним, сумма делится на количество сфер, производится проверка полученного значения и вывод информации пользователю.

Для вывода графика, пользователю необходимо выбрать вид графика и значения. Например, пользователь может выбрать столбчатую диаграмму за определенный период (рис. 8) или круговую диаграмму за весь период (рис. 9). Для этого используется запрос на получение всех данных по сферам с просуммированными платежами по ним, где дата платежа находится в диапазоне введенных пользователем дат. Если таких значений нет, то график построен не будет. Далее платеж сопоставляется со сферой по id сферы, формируется словарь из сфер платежей и расходов, где ключом является сфера расходов, а значением – сумма расходов (листинг 1). По данному словарю отображается график с помощью библиотеки «matplotlib» [3].

|  |  |
| --- | --- |
| Рис. 8. Столбчатая диаграмма расходов по сферам за выбранным пользователем период | Рис. 9. Круговая диаграмма расходов по сферам за весь период |

Таким образом, было разработано полноценное кроссплатформенное финансовое приложение для контроля личных расходов.

Листинг 1

def krug():  
 with sqlite3.connect("database.db") as db:  
 db.row\_factory = sqlite3.Row  
 paymetsCursor = db.cursor()  
 querty = """SELECT expense\_id, SUM(amount) as amount from payments GROUP BY expense\_id"""  
 paymetsCursor.execute(querty)  
 all\_data = paymetsCursor  
 if paymetsCursor.rowcount == 0:  
 return "НЕТ ДАННЫХ"  
 amount = {}  
 for i in all\_data:  
 amount[i["expense\_id"]] = i["amount"]  
 cursor = db.cursor()  
 querry = """SELECT id, name from expenses"""  
 cursor.execute(querry)  
 result = {}  
 for item in cursor:  
 if item["id"] in amount.keys():  
 result[item["name"]] = amount[item["id"]]  
 else:  
 result[item["name"]] = 0  
 db.close()  
 fig1, ax1 = plt.subplots() # вывод круговой диаграммы  
 wedges, texts, autotexts = ax1.pie(result.values(), labels=result.keys(), autopct='%1.2f%%') # вывод значений в процентах  
 ax1.pie(result.values(), labels=result.keys()) # значения для диаграммы  
 ax1.legend(result.keys(), loc='upper left', bbox\_to\_anchor=(1.0, 1.1)) # легенда - какие данные и их расположение  
 plt.title("Расходы за весь период")  
 plt.show()

**Библиографический список:**

1. DB-API 2.0 interface for SQLite databases 2022. URL: https://docs.python.org/3/library/sqlite3.html (дата обращения: 09.03.22)
2. PyQT5 Reference Guide 2021. URL: https://www.riverbankcomputing.com/static/Docs/PyQt5/ (дата обращения: 17.03.21)
3. Matplotlib: Visualization with Python 2021. URL: https://matplotlib.org/ (дата обращения: 21.02.22)