МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ

ННІ Навчально-науковий інститут економіки та бізнес-освіти

Кафедра Економіки та цифрового бізнесу

Спеціальність 122 «Комп'ютерні науки»

Форма навчання Заочна

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

Сердюк Олександри Олегівни
(прізвище, ім'я, по батькові здобувача)

на тему Розробка застосунку для класифікації та моніторингу персональних витрат
(повна назва теми)

за матеріалами

(повна назва бази дослідження)

науковий керівник к.т.н. _____ Селезньов М.Є.
(наук. ступінь, вчене звання) *(підпис)* *(прізвище, ініціали)*

Робота допущена до захисту в ЕК
Протокол засідання кафедри
від 12 червня.2026р. № 15
Завідувач кафедри _____
(підпис)
к.е.н., доцент Радько В.М.
наук. ступень, вчене звання прізвище, ініціали

Кривий Ріг – 2026

ЗАТВЕРДЖЕНО
Наказ Міністерства освіти і науки, молоді та спорту України
29 березня 2012 року № 384

Форма № Н-9.01

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ ЕКОНОМІКИ І ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ЕКОНОМІКИ ТА БІЗНЕС-ОСВІТИ
(повне найменування вищого навчального закладу)

Кафедра економіки та цифрового бізнесу
Освітній ступінь бакалавр
Спеціальність 122 «Комп'ютерні науки»

ЗАТВЕРДЖУЮ
Завідувач кафедри _____В.М. Радько

"30" березня 2026 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА ЗДОБУВАЧУ

Сердюк Олександрі Олегівні

1. Тема роботи Розробка застосунку для класифікації та моніторингу персональних витрат

науковий керівник роботи Селезньов Максим Євгенович, затверджені наказом вищого навчального закладу від «23» березня 2026 р. № 194-ст

2. Строк подання здобувачем роботи 29.05.2026р.

3. Зміст кваліфікаційної роботи бакалавра, об'єкт, предмет та мета дослідження:
Розділ 1 Теоретичний аналіз особливостей системи класифікації та моніторингу персональних витрат
Розділ 2 Проєктування застосунку для класифікації та моніторингу персональних витрат
Розділ 3 Реалізація застосунку для класифікації та моніторингу персональних витрат

Об'єкт дослідження – процес управління персональними фінансами

Предмет дослідження – адаптивна система класифікації та моніторингу персональних витрат

Мета кваліфікаційної роботи бакалавра – розробити систему, яка дозволить користувачу зберігати та аналізувати інформацію про свої персональні транзакції

4. Дата видачі завдання 03.04.2026р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної бакалаврської роботи	Строк виконання етапів роботи	Відмітка керівника про виконання етапів (дата, підпис)
1	Підготовка розділу 1	до 17.04.2026р.	16.04.2026р.
2	Підготовка розділу 2	до 08.05.2026р.	07.05.2026р.
3	Підготовка розділу 3	до 25.05.2026р.	24.05.2026р.
4	Реєстрація завершеної кваліфікаційної роботи	до 29.05.2026р.	28.05.2026р.
5	Отримання відгуку від наукового керівника	04.06.2026р.	04.06.2026р.
6	Отримання зовнішньої рецензії	05.06.2026р.	05.06.2026р.
7	Попередній захист кваліфікаційної роботи на кафедрі	08.06.2026р.	08.06.2026р.
8	Перевірка кваліфікаційної роботи на плагіат	18.06.2026р.	18.06.2026р.
9	Допуск кафедрою кваліфікаційної роботи до захисту	18.06.2026р.	18.06.2026р.
10	Підготовка студента до захисту в ЕК	до 23.06.2026р	22.06.2026р

Завдання підготував науковий керівник _____ **Селезньов М.Є.**

(підпис)

Завдання одержав здобувач _____ **Сердюк О.О.**

(підпис) (прізвище та ініціали)

РРЕФЕРАТ

Робота містить 56 сторінок, 56 рисунків, 22 джерела, 5 додатків.

Об’єкт дослідження: процес управління персональними фінансами.

Мета: розробити систему, яка дозволить користувачу зберігати та аналізувати інформацію про свої персональні транзакції.

В результаті дослідження було вивчено методи обліку персональних фінансів. Були сформовані вимоги до функціоналу системи, спроєктована структура БД та макет користувацького інтерфейсу. В результаті був розроблений застосунок для класифікації та моніторингу персональних витрат

Область застосування: результати цієї роботи можуть бути використані індивідуальними користувачами, сім’ями, фінансовими консультантами та планувальниками.

БАЗА ДАНИХ, ДІАГРАМА, КЛАС, МОНІТОРИНГ, ПЕРСОНАЛЬНІ ВИТРАТИ, ПРОГРАМНИЙ ЗАСІБ, СИСТЕМА, ФОРМАОБЛІК

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ ……..……..... ... 6
ВСТУП ……..……..... ... 7
РОЗДІЛ 1 ТЕОРЕТИЧНИЙ АНАЛІЗ ОСОБЛИВОСТЕЙ СИСТЕМИ КЛАСИФІКАЦІЇ ТА МОНІТОРИНГУ ПЕРСОНАЛЬНИХ ВИТРАТ 9
1.1 Аналіз існуючих методів і підходів до класифікації та моніторингу персональних витрат 9
1.2 Аналіз існуючих систем і додатків для управління персональними фінансами 10
Висновки до розділу 1 ……..……..... 12
РОЗДІЛ 2 ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ КЛАСИФІКАЦІЇ ТА МОНІТОРИНГУ ПЕРСОНАЛЬНИХ ВИТРАТ …… 13
2.1 Розробка функціональної схеми застосунку ……..... ... 13
2.2 Розробка структури бази даних застосунку ……...……..... ... 15
2.3 Розробка структури меню та концепту інтерфейсу застосунку ……..... 21
Висновки до розділу 2 ...……..…...……..... ... 25
РОЗДІЛ 3 РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ КЛАСИФІКАЦІЇ ТА МОНІТОРИНГУ ПЕРСОНАЛЬНИХ ВИТРАТ ……..…… 26
3.1 Вибір мови програмування та фреймворку для розробки застосунку …… 26
3.2 Реалізація дизайну інтерфейсу застосунку ……..... 30
3.3 Реалізація основних функцій застосунку ……..……..... 36
3.4 Сценарії використання застосунку ……...……..……..... 42
Висновки до розділу 3 ……..…… … 53
ВИСНОВКИ ……..……..... 54
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ ……...……..... 55
ДОДАТКИ ……...……..... 57

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

БД – База даних

UI – User Interface (укр. Користувацький інтерфейс)

GUI – Graphical User Interface (укр. Графічний користувацький інтерфейс)

ООП – Об'єктно-орієнтоване програмування

СУБД – Система управління базами даних

ВСТУП

Управління особистими фінансами – важливе та актуальне завдання в сучасному світі. Швидкі зміни на фінансових ринках, економічна нестабільність та збільшення фінансового тягаря на індивідуальному рівні наголошують на необхідності ефективного управління фінансовими ресурсами.

Актуальність цього питання обумовлена тим, що багато людей стикаються з труднощами в управлінні своїми фінансами. Відсутність належного планування та організації може призвести до неправильних фінансових рішень, перевитрати коштів та невиправданих боргів. Багатьом людям також важко ефективно використовувати свої гроші через відсутність знань про фінансові продукти та стратегії.

Важливість особистих фінансів у тому, що правильне управління ними значно впливає на якість життя людей. Люди, які ефективно управляють своїми фінансами, здатні досягати своїх фінансових цілей, забезпечувати безпеку собі та своїм сім'ям, інвестувати у свій майбутній та особистісний розвиток.

Це робить управління особистими фінансами актуальним і важливим питанням, оскільки це зачіпає кожного і може вплинути на якість життя та економічний розвиток. Вирішення цієї проблеми є важливим кроком на шляху до досягнення економічної безпеки, економічного щастя та самореалізації.

Отже, метою дипломної роботи є розробка системи, яка дозволить користувачу зберігати та аналізувати інформацію про свої персональні транзакції.

Об’єктом дослідження є процес управління адаптивною системою класифікації та моніторингу персональних витрат.

Предметом дослідження є адаптивна система класифікації та моніторингу персональних витрат.

Інформаційну базу складають джерела такі, як: книги, посібники, наукові статті та документація програмного забезпечення.

Для досягнення поставленої мети треба виконати наступні завдання:

- дослідити методи і підходи класифікації моніторингу персональних витрат;
- проаналізувати адаптивні системи і додатки для управлінная персональними фінансами;
- проаналізувати вимоги системи для реалізації системи управління персональними витратами;
- розробити функціональну схему системи моніторингу персональних витрат, структуру БД;
- розробити систему варіантів персональних витрат та інтерфейсу;
- реалізувати системи класифікації і моніторингу персональних витрат.

РОЗДІЛ 1
ТЕОРЕТИЧНИЙ АНАЛІЗ ОСОБЛИВОСТЕЙ СИСТЕМИ КЛАСИФІКАЦІЇ ТА МОНІТОРИНГУ ПЕРСОНАЛЬНИХ ВИТРАТ

1.1 Аналіз існуючих методів і підходів до класифікації та моніторингу персональних витрат

Управління особистими фінансами стосується доходів, витрат, активів і пасивів є важливою частиною економічного життя кожної людини. Цей процес включає знання ваших фінансів, встановлення фінансових цілей, встановлення бюджету та моніторинг його виконання.

Одним із перших кроків у управлінні особистими фінансами є ретельна оцінка вашого фінансового становища. Це аналіз ваших доходів, витрат, активів і зобов'язань, необхідний щоб отримати чітке уявлення про ваше фінансове становище. Наступним кроком є встановлення фінансових цілей. Цілі можуть бути як короткостроковими (наприклад, погашення боргу), так і довгостроковими (наприклад, створення пенсійного фонду).

Бюджетування – метод організації фінансових ресурсів. Цей інструмент допоможе вам планувати та керувати своїми витратами, оптимізувати свої доходи та вирішувати, що робити зі своїми коштами. Бюджети дозволяють ефективно розподіляти ресурси, уникати перевитрат і планувати майбутню фінансову діяльність.

Важливою частиною управління особистими фінансами є моніторинг та оцінка ефективності. Регулярний моніторинг вашого фінансового стану та оцінка досягнення ваших фінансових цілей допоможе вам внести відповідні корективи та покращити свою стратегію управління фінансами[1-6].

Уміння ефективно управляти особистими фінансами допомагає забезпечити фінансову стабільність, зменшити фінансовий стрес та вдосконалювати стратегію управління фінансами. Забезпечити фінансову стабільність, зменшити економічний стрес і досягти фінансової незалежності.

Уміння раціонально планувати, управляти та організовувати використання фінансових ресурсів є ключовим фактором прийняття обґрунтованих фінансових рішень та ефективного використання коштів.

1.2 Аналіз існуючих систем і додатків для управління персональними фінансами

Для більш якісного оцінювання потреб користувача перед початком проєктування системи потрібно проаналізувати існуючі аналоги. Це дозволить нам визначитися із базовим функціоналом та архітектурою системи.

Book Account від розробника Wakeeson Inc (див. рис. 1.1) - це мобільний додаток, який має версію для персонального комп'ютера, створений для управління персональними фінансами і відстеженнями витрат.

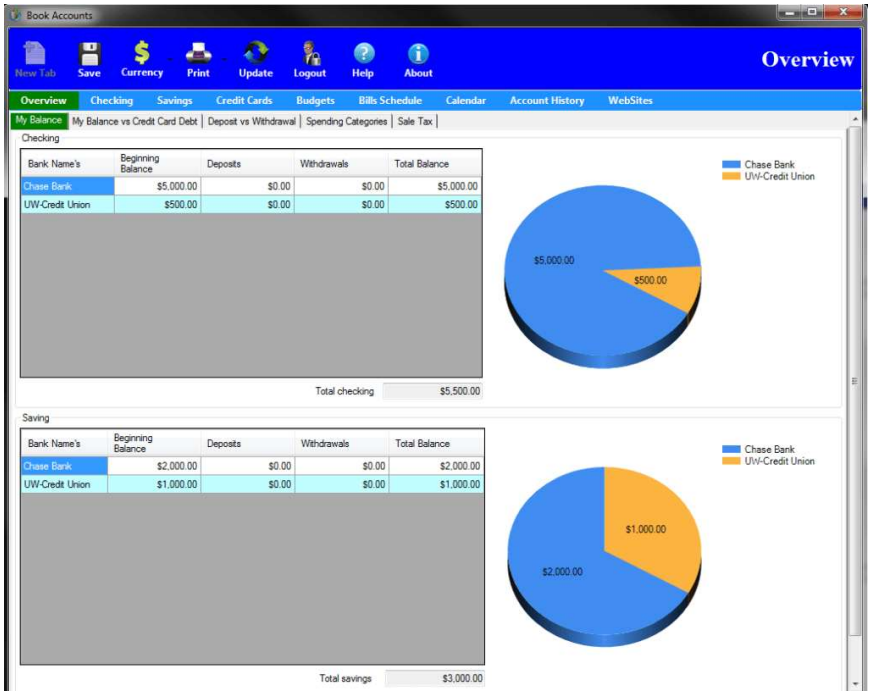

Рис. 1.1. Головне меню Book Account

Book Accounts - це програмне забезпечення для управління особистими фінансами, яке відстежує всі ваші чеки, заощадження та кредити. За допомогою його ви можете організувати свої рахунки в одному місці [17].

Додаток пропонує функцію створення бюджету, яка допомагає встановити ліміти витрат для різних категорій. Ви можете стежити за своїми витратами й отримувати повідомлення, коли ви наближаєтеся до граничної суми. Це допомагає вам планувати свої фінанси й уникати необхідності запозичення грошей.

Додаток надає докладні аналітичні дані та звіти про ваші витрати і доходи. Ви можете легко переглянути графіки та діаграми, які допоможуть вам візуалізувати свої фінансові патерни та звички. Це корисно для ухвалення обґрунтованих рішень про свої фінанси та планування майбутніх витрат.

При цьому, наразі Book Accounts не пропонує синхронізацію даних між кількома пристроями або резервне копіювання у хмарі. Додаток надає базові звіти та візуалізації для відстеження витрат і прибутків у часі. Однак йому бракує більш детальної аналітики та настроюваних функцій звітності, які могли б бути корисними для користувачів, що потребують поглибленого фінансового аналізу.

Додаток Moneyble Personal Finance (див. рис. 1.2), для відстежування персональних витрат із тісною сумістністю MS Excel, розроблений для тих, хто сьогодні використовує Excel для відстеження особистих фінансів.

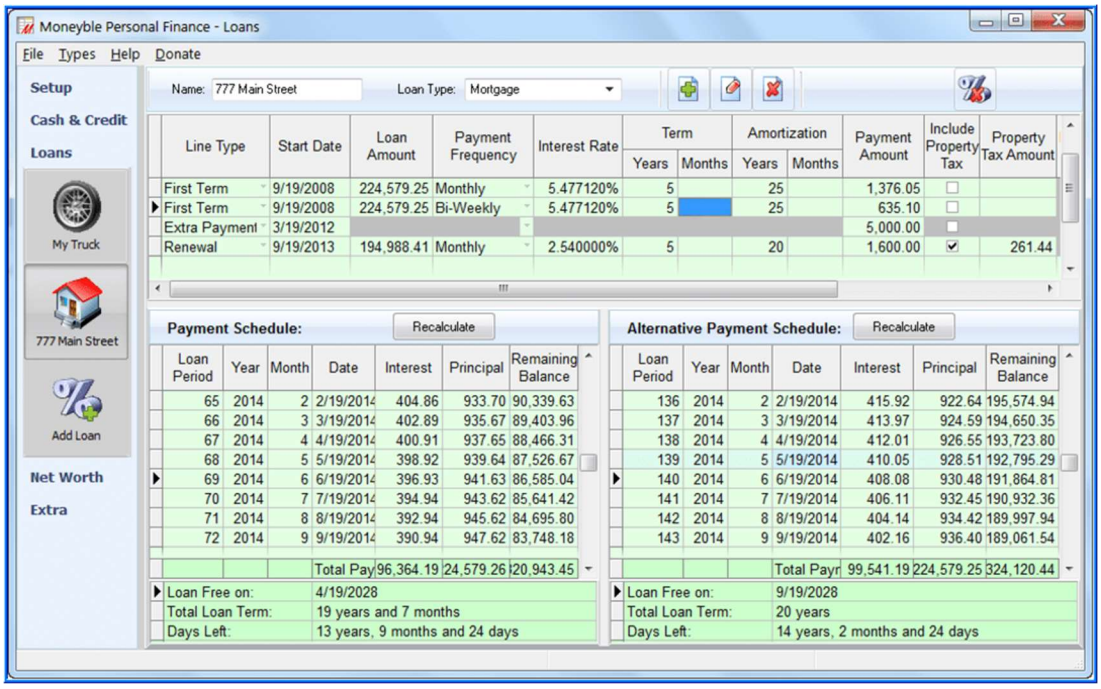

Рис. 1.2. Головне меню Moneyble Personal Finance

Всі екрани засновані на електронних таблицях, що дозволяє швидко вводити та редагувати дані. Ви можете налаштувати, які стовпці ви хочете бачити і як їх називати. Ви можете налаштувати таблицю транзакцій Moneyble так, щоб вона виглядала як ваш існуючий файл Excel [18].

Перевагами є чудова інтеграція з Excel таблицями. Є можливість імпортування даних з вашого Excel файлу для миттєвого початку роботи з додатком. Додаток надає багато можливостей для кастомізації. Такі як редагування видимих стовпців таблиці, типів даних тощо.

Недоліками є те, що додаток наврядчи підійде середньостатистичному користувачу внаслідок не зовсім доброзичливого інтерфейсу. Деякі функції та налаштування можуть бути неочевидними або вимагати часу для освоєння.

Висновки до розділу 1

Після аналізу методів і підходів до класифікації та моніторингу персональних витрат та декількох існуючих додатків орієнтованих на управління фінансами можна скласти перелік основних вимог для реалізації в системі.

Система повинна мати інтуїтивно зрозумілий і простий у використанні інтерфейс, який дає змогу користувачам легко вводити і відстежувати свої витрати. Також, система має надавати можливість створення та налаштування категорій витрат, які відповідають індивідуальним потребам користувача. Категорії можуть включати такі сфери, як харчування, транспорт, розваги, здоров'я тощо.

Крім того, система повинна надавати можливість аналізу та генерації звітів про витрати. Це дає змогу користувачам отримувати оглядові дані про свої витрати, виявляти тренди, аналізувати патерни витрат і ухвалювати поінформовані рішення про свої фінанси.

РОЗДІЛ 2
ПРОЄКТУВАННЯ ЗАСТОСУНКУ ДЛЯ КЛАСИФІКАЦІЇ ТА МОНІТОРИНГУ ПЕРСОНАЛЬНИХ ВИТРАТ

2.1 Розробка функціональної схеми застосунку

Проектування системи полягає в розробленні діаграм, що базуються на попередньо проаналізованих даних та будуть всебічно описувати структуру майбутнього додатка.

На рис. 2.1 – 2.5 зображені контекстні діаграми, що відображають всі взаємодії в системі, та її подальшу декомпозицію.

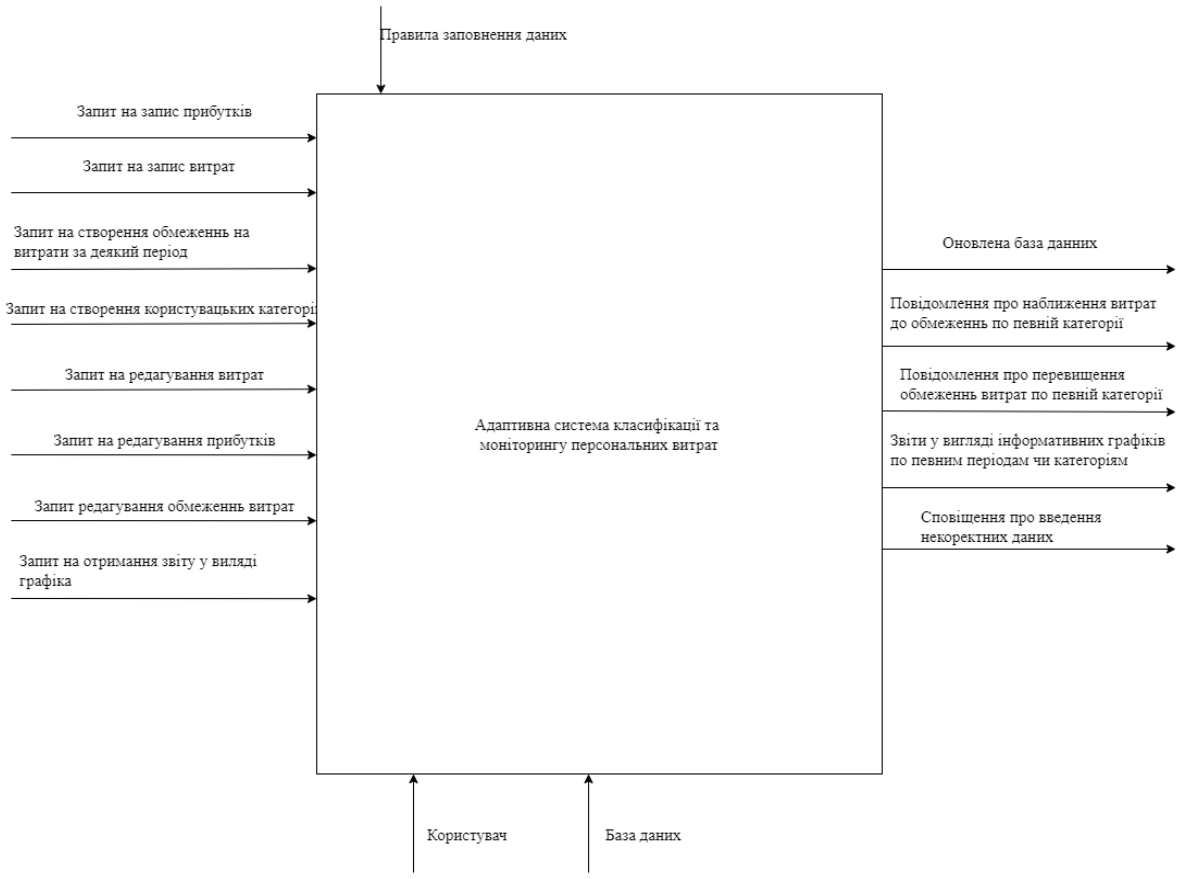

Рис. 2.1. Контекстна діаграма

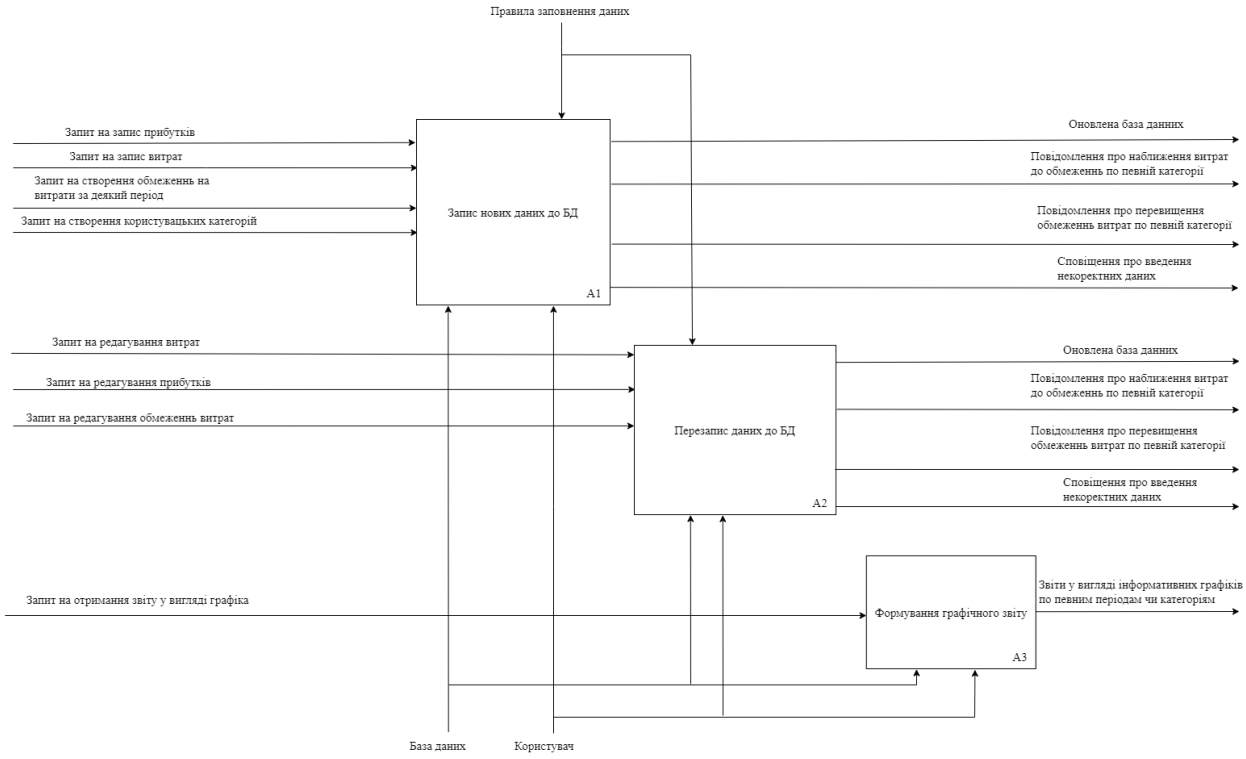

Рис. 2.2. Діаграма A0

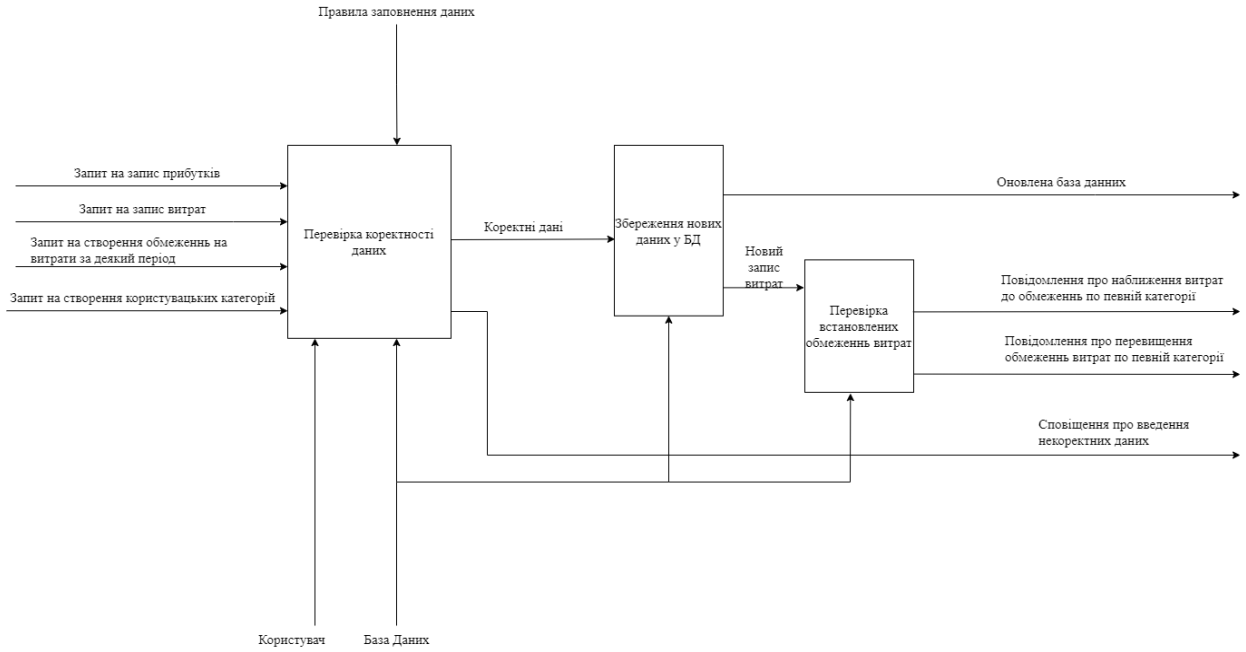

Рис. 2.3. Діаграма A1

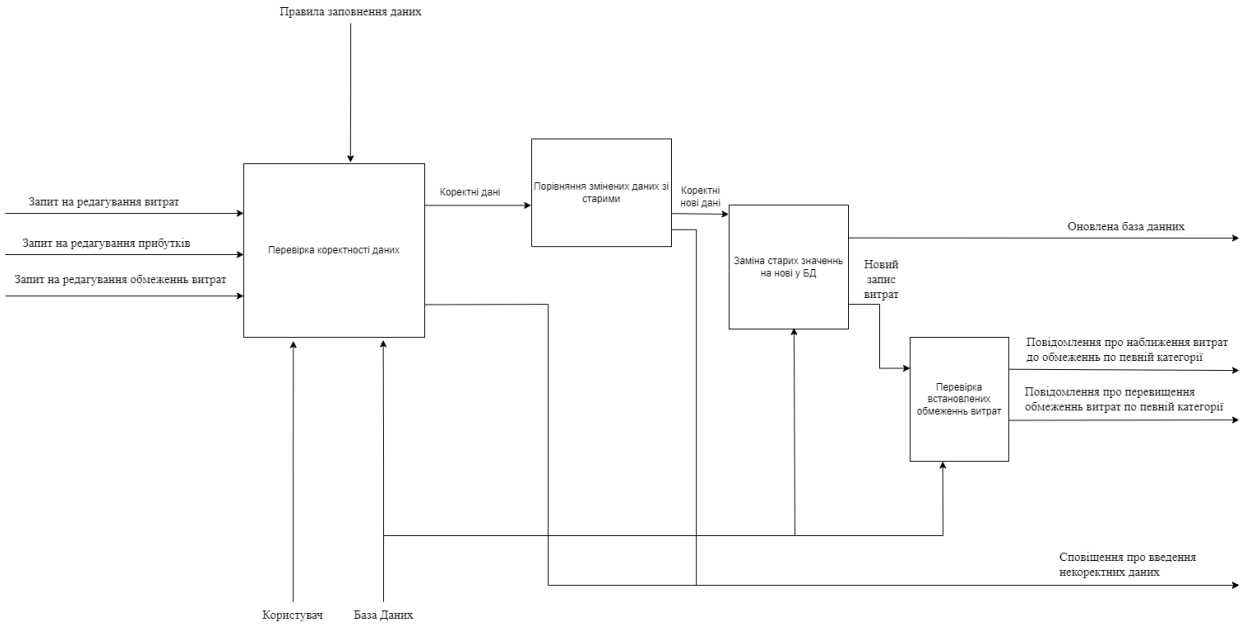

Рис. 2.4. Діаграма А2

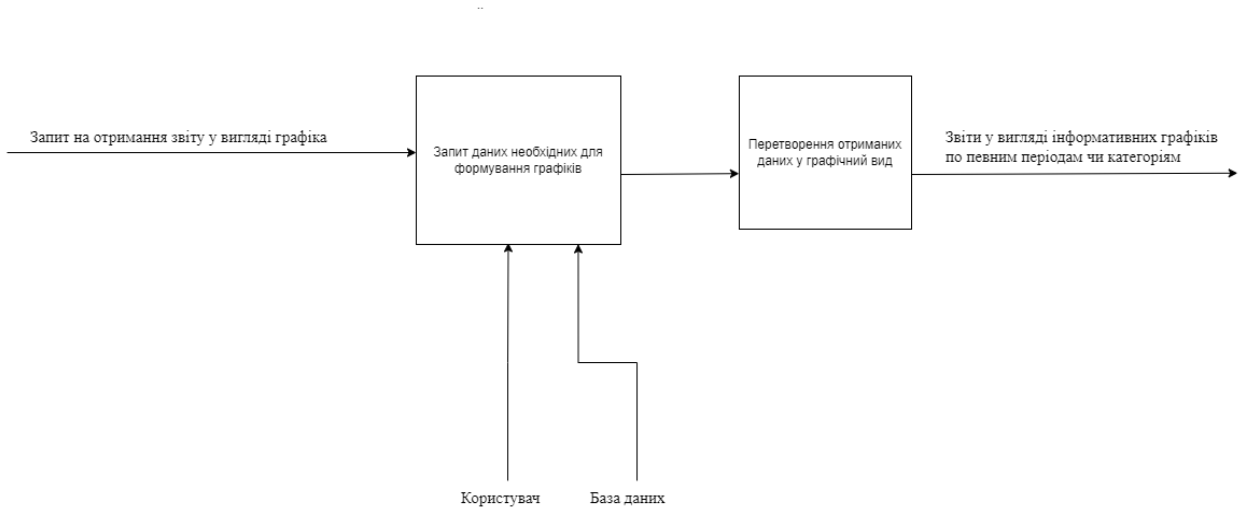

Рис. 2.5. Діаграма А3

2.2 Розробка структури бази даних застосунку

Згідно вищеописаним діаграмам розроблюється структура бази даних. Ця стадія є концептуальним проєктуванням системи. Її результатом є концептуальна модель предметної області.

До концептуальних моделей відносять компоненти, які різними способами відображають предметну область. Окрім найбільш відомого опису об’єктів і зв’язків між ними – сутність-зв’язок, до концептуального опису предметної області можна віднести наступні компоненти:

- систему атрибутів і засобів опису предметної області;
- обмеження цілістності, яке визначає коректність значення окремих полів і взаємозв’язків як на рівні семантики вмісту БД, так і її фізичної структури;
- опис інформаційних потреб користувачів, наприклад, у виді типових запитів, відображаючих процедурні особливості звертання до даних.

Для обраної предметної області була розроблена концептуальна модель, у якій представлено 7 таблиць: Витрати, Прибуток, Рахунок, СписокКатегорійВитрат, СписокКатегорійПрибутку, СписокПеріодів, СписокОбмеженнь. Таблиці пов’язані між собою зв’язками, які потрібні для зберігання цілістності даних та видалення записів.

Таблиці «Витрати» і «Прибуток» матимуть ідентичну структуру (див. рис. 2.6) і зберігатимуть інформацію про витрати зроблені користувачем, або його прибутки відповідно. Ці таблиці матимуть поля: Код, Категорія, Сума, Дата, НазваРахунку. Далі доцільно навести детальний опис полів.

Код – ключове поле, яке дає можливіть однозначно ідентифікувати запис. Тип даних – int, довжина поля – довге ціле число.

Категорія – несе інформацію про категорію транзакції. Тип даних – char, довжина поля – 30 символів, приклад – «Продукти».

Сума – величина транзакції. Тип даних – float, число десятичних знаків – 2 символа, приклад – «38,50».

Дата – несе інформацію про дату та час здійснення транзакції. Тип даних – date, довжина поля – 8 символів, приклад – «01.05.2023».

НазваРахунку – зберігає інформацію про рахунок з якого була здійснена транзакція. Тип даних – char, довжина поля – 20 символів, приклад – «Основний».

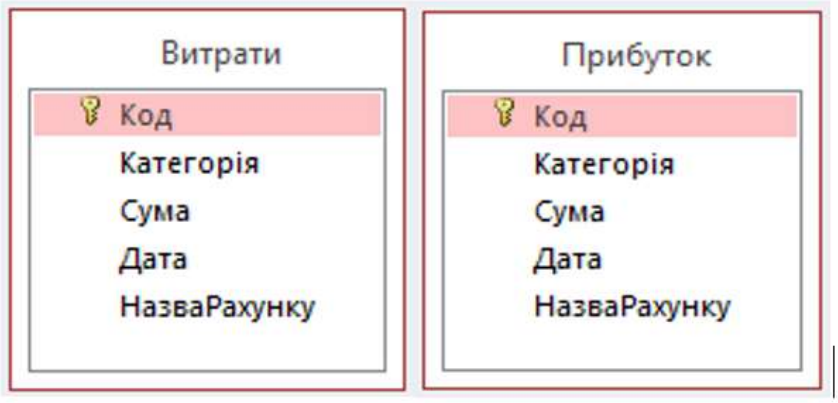

Рис. 2.6. Структура таблиць «Витрати» і «Прибуток»

Таблиця «Рахунок» (див. рис. 2.7) матиме інформацію про рахунки користувача та суму, яка зберігається на цих рахунках. Поля таблиці – НазваРахунку, Сума. Далі наведений детальний опис полів.

НазваРахунку – ключове поле, зберігає інформацію про рахунок користувача. Тип даних – char, довжина поля – 20 символів, приклад – «Основний».

Сума – грошова сума наявна на рахунку. Тип даних – float, число десятичних знаків – 2 символа, приклад – «10362,53».

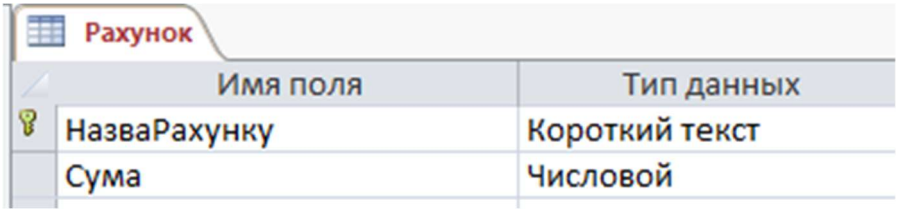

Рис. 2.7. Структура таблиці «Рахунок»

Таблиці «СписокКатегорійВитрат», «СписокКатегорійПрибутку», «СписокПеріодів» також матимуть ідентичну структуру (див. рис. 2.8), але зберігатимуть дані відповідні до своєї суті. Далі наведений детальний опис полів.

Назва – ключове поле, зберігає назви до відповідної категорії об’єктів. Тип даних – char, довжина поля – 20 символів, приклад – «Основний».

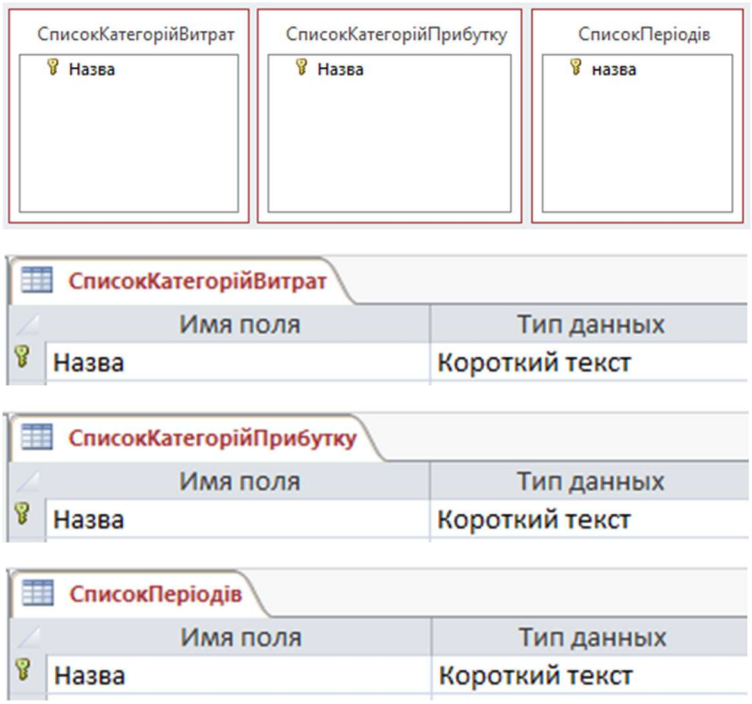

Рис. 2.8. Структура таблиць «СписокКатегорійВитрат», «СписокКатегорійПрибутку», «СписокПеріодів»

Таблиця «СписокОбмежень» (див. рис. 2.9) зберігає дані про обмеження витрат за обраний період часу та категорії. Поля таблиці – Код, НазваОбмеження, Період, Категорія, Сума. Далі наведений детальний опис полів.

Код – ключове поле, яке дає можливіть однозначно ідентифікувати запис. Тип даних – int, довжина поля – довге ціле число.

НазваОбмеження – поле зберігає назву обмеження для зручності у сповіщенні користувача про наближення витрат до порогу, або його перевищення. Тип даних – char, довжина поля – 30 символів, приклад – «ОсвітаРік».

Період – поле зберігає дані про період за який період потрібно розрахувати витрати і звіряти їх з пороговим значенням. Тип даних – char, довжина поля – 20 символів, приклад – «Рік».

Категорія – несе інформацію про категорію за якою розраховуються витрати для їх звіряння з пороговим значенням. Тип даних – char, дожина поля – 30 символів, приклад – «Продукти».

Сума – дані порогового значення витрат. Тип даних – float, число десятичних знаків – 2 символа, приклад – «8000,00».

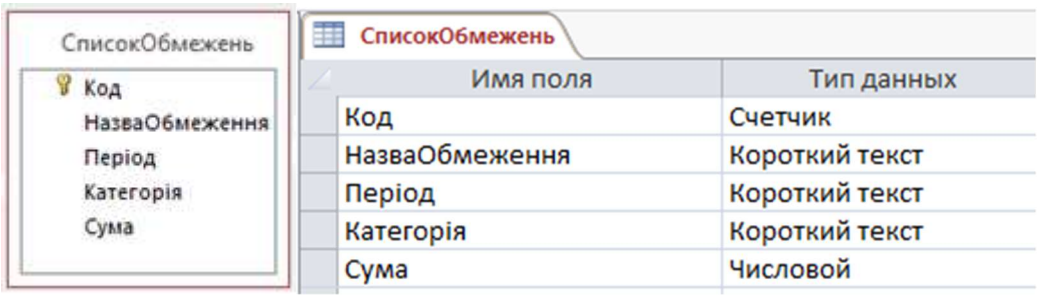

Рис. 2.9. Структура таблиці «СписокОбмежень»

Крім того була розроблена логічна структурна схема БД, яка представлена на рисунку 2.10.

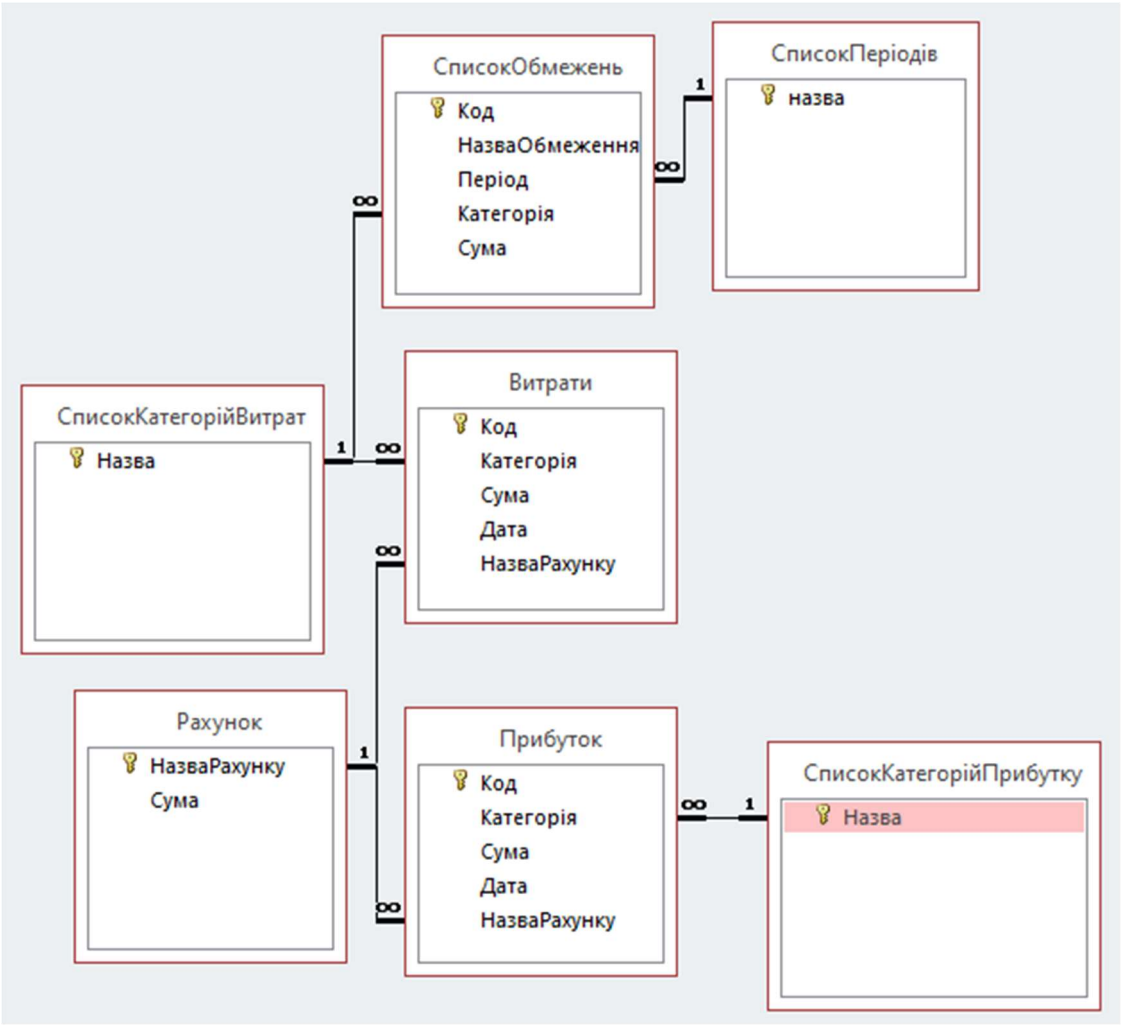

Рис. 2.10. Схема зв’язків БД

Діаграма варіантів використання (див. рис. 2.11) є графом з об’єктами акторів та прецедентів. Вона потрібна для відображення залежностей акторів з прецедентами, що формує наглядну модель взаємодій в системі.

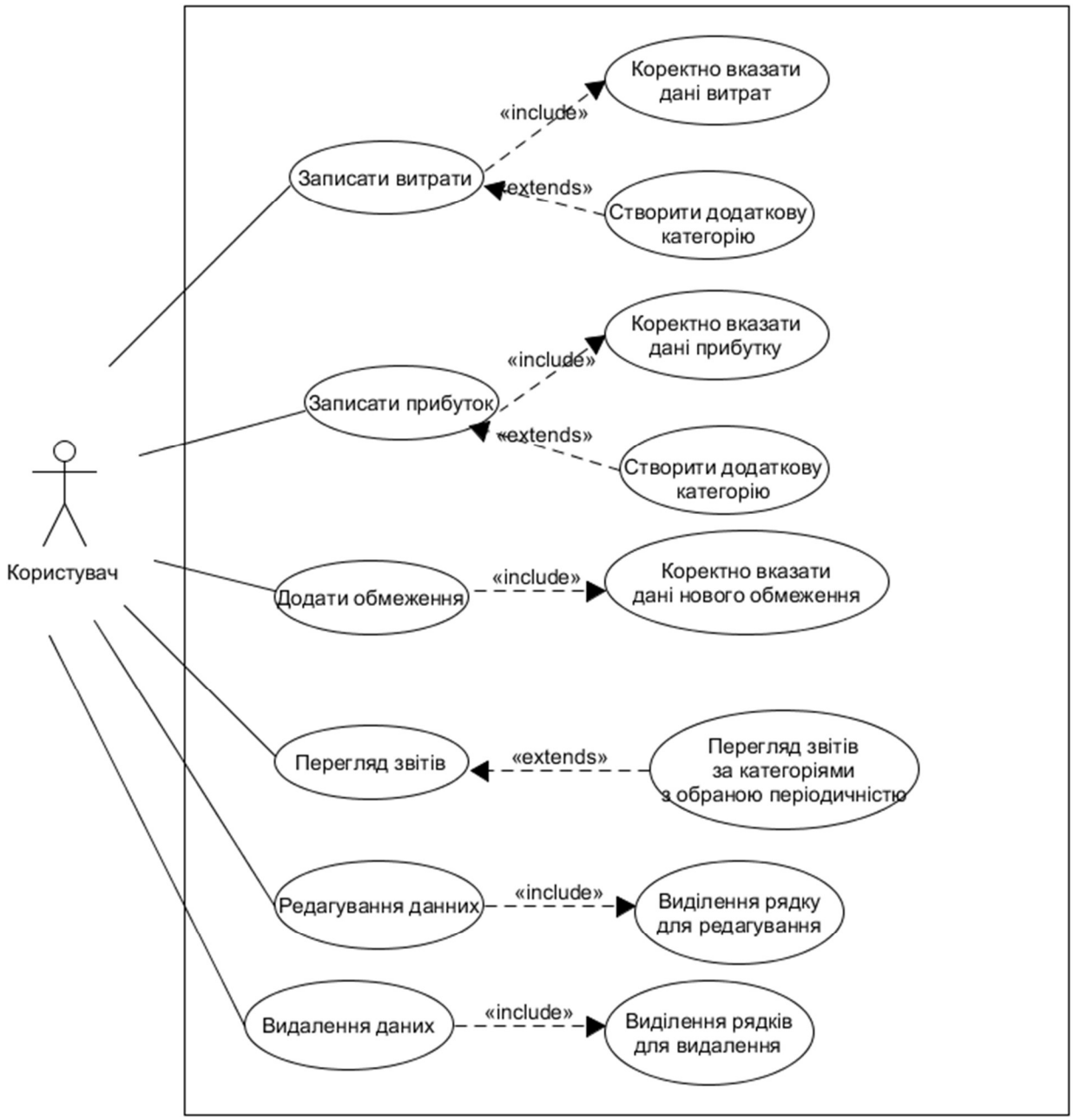

Рис. 2.11. Діаграма варіантів використання

2.3 Розробка структури меню та концепту інтерфейсу застосунку

Користувацький інтерфейс (UI) – це простір, у якому відбувається взаємодія між людиною і програмним засобом. UI складається з усіх елементів за допомогою яких користувач може взаємодіяти з системою і включає в себе такі елементи, як графічний інтерфейс, форми для заповнення даних, текстові поля, кнопки тощо.

Адаптивна система обліку з спроєктованим вище функціоналом повинна мати такі аспекти користувацького інтерфейсу:

- графічний інтерфейс(GUI): використання графічних елементів, таких як кнопки, меню, форми, таблиці і графіки, для візуального представлення фінансової інформації та дій користувача. Графічний інтерфейс повинен бути зрозумілим, зручним у використанні і естетично привабливим;
- меню та навігація: система повинна мати чітку структуру меню та навігації, щоб користувач міг швидко знаходити потрібні функції та переходити між різними розділами системи;
- форми введення даних: система повинна мати зручні форми для введення фінансових даних, таких як суми транзакцій, категорії витрат, дати тощо. Форми повинні бути простими в заповненні і забезпечувати правильну валідацію та перевірку даних;
- звіти та аналітика: система повинна надавати можливість генерувати звіти, графіки та статистику щодо фінансових даних. Це допомагає користувачам аналізувати свої витрати, доходи, збереження тощо;
- повідомлення та сповіщення: система може надсилати повідомлення та сповіщення користувачам щодо важливих фінансових подій, наприклад, наближення до межі бюджету, платежів, сповіщень про важливі дати тощо;

За встановленими критеріями користувацького інтерфейсу була спроєктована структура меню, яка представлена на рис. 2.12.

Рис. 2.12. Структура меню додатку

На основі попередньо спроектованих елементів системи обліку можна представити початковий концепт графічного користувацького інтерфейсу, який представлений на рис. 2.13.

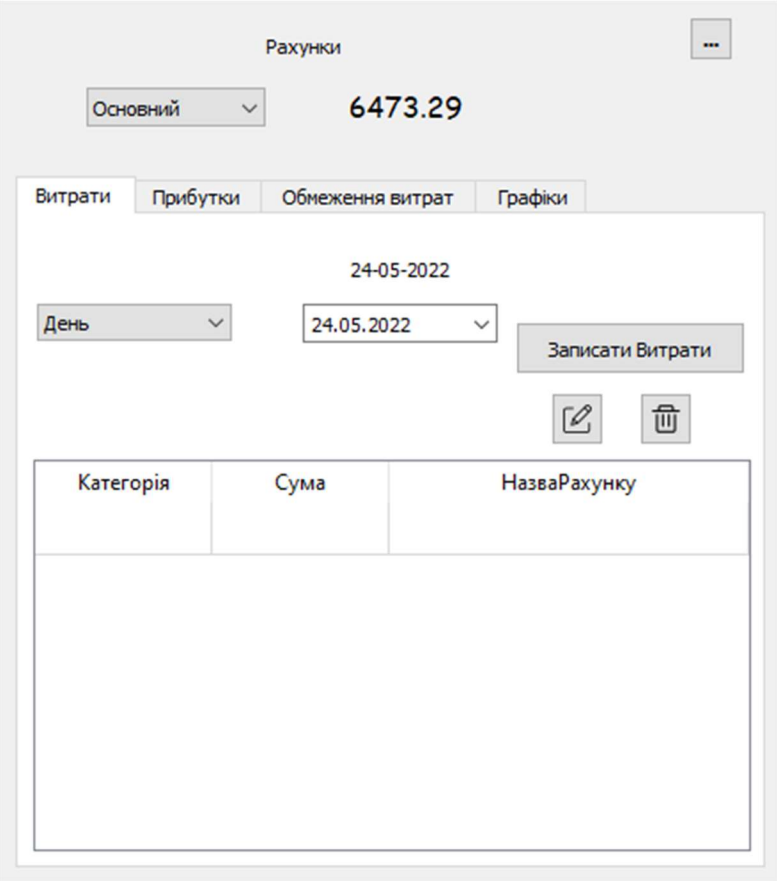

Рис. 2.13. Концепт головного меню

У верхній частині GUI (див. рис. 2.14) будуть розташовуватися інформація про рахунки користувача та невелика іконка зі сповіщеннями про статус обмеженнь витрат встановлених користувачем.

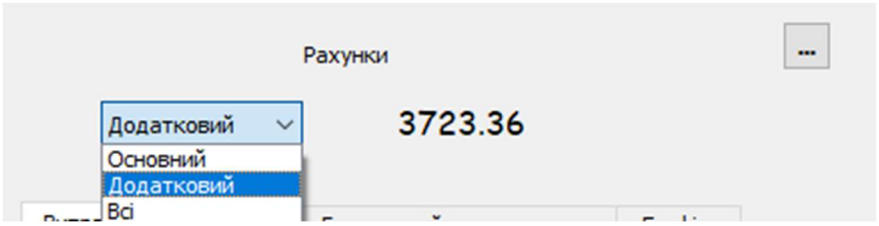

Рис. 2.14. Концепт панелі рахунків та кнопки сповіщень

Нижні 80% графічного інтерфейсу будуть займати форми для перегляду даних по таблицям витрат, прибутків, обмеженнь витрат, а також – графічних звітів.

На формах «Витрати» і «Прибутки» (див. рис. 2.15–2.16) будуть надані елементи, які дозволять обирати період за який були проведені транзакції, а також таблиця для відображення обраних даних та кнопки для створення нових записів, редагування і видалення існуючих записів.

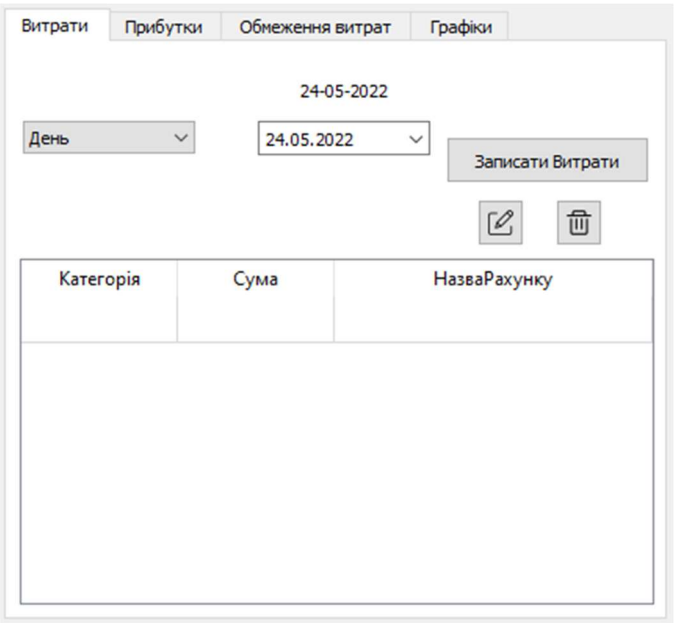

Рис. 2.15. Концепт вкладки **«Витрати»**

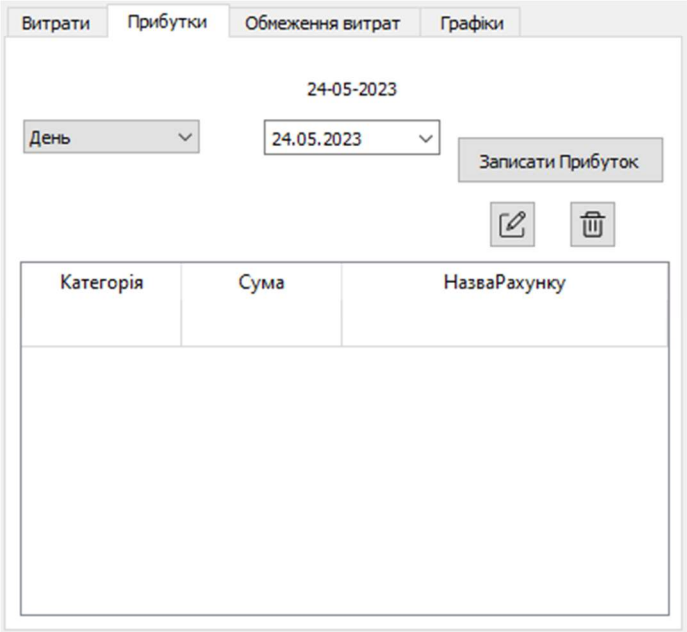

Рис. 2.16. Концепт вкладки «Прибутки»

На вкладці «Обмеження витрат» (див. рис. 2.17) надається форма для створення нових даних і буде присутня одразу так, як не буде потреби у фільтрах для відображення даних і всі створені обмеження будуть відображатися у таблиці. Також наявна кнопка для підтвердження створення запису і кнопки для редагування і видалення існуючих записів.

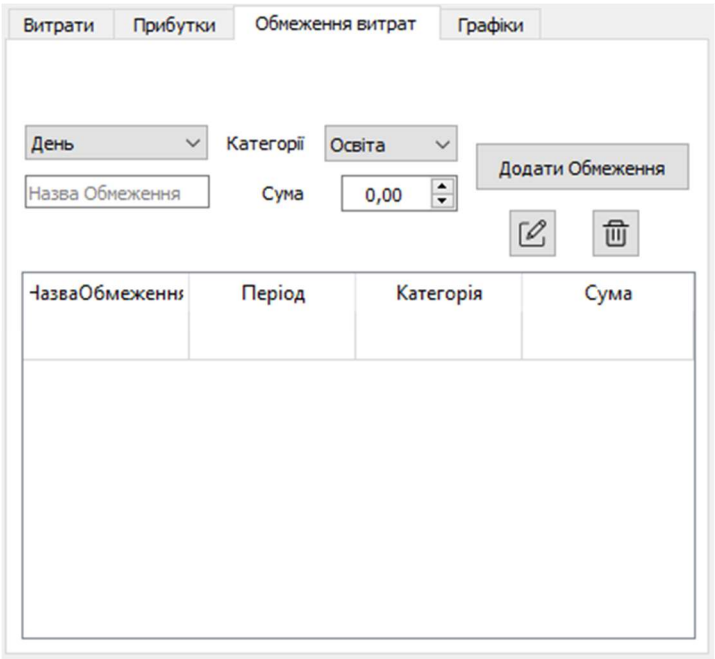

Рис. 2.17. Концепт вкладки «Обмеження витрат»

Остання форма «Графіки» (див. рис. 2.18) буде мати фільтри для даних і відображати звіти по даним з БД у вигляді вертикальної діаграми згідно обраним параметрам фільтрів.

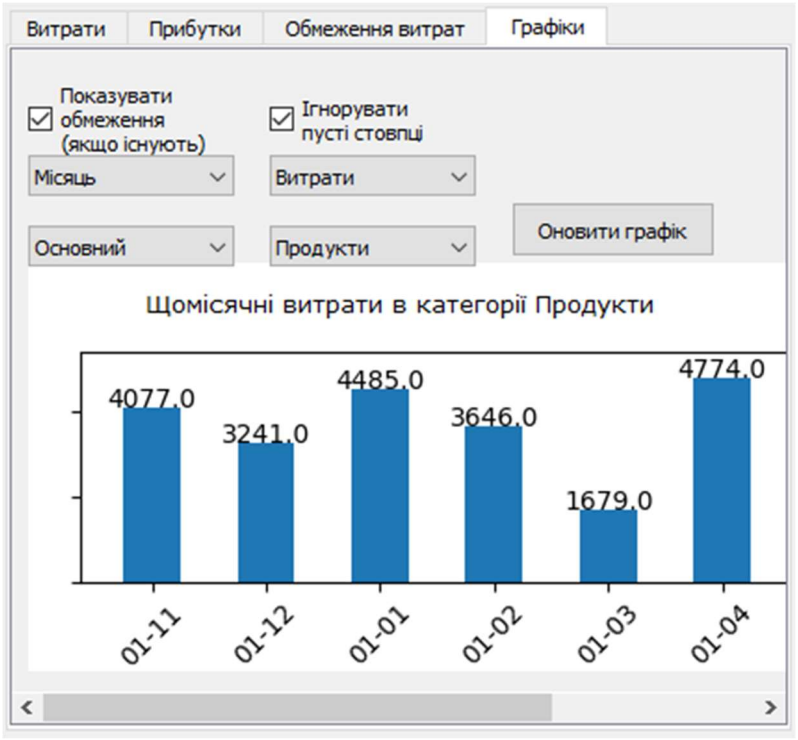

Рис. 2.18. Концепт вкладки «Графіки»

Висновки до розділу 2

На стадії моделювання було спроєктовано структуру БД, яка містить таблиці Витрати, Прибуток, Рахунок, СписокКатегорійВитрат, СписокКатегорійПрибутку, СписокПеріодів, СписокОбмеженнь, та ряд інформативних діаграм, які на різних рівнях описують взаємодії як між об'єктами всередині системі, так і між системою та користувачем, а саме ER-діаграма, функціональна даграма та її розгортки, діаграма варіантів використання. Також спроєктовано концепт користувацького інтерфейсу, який складається з структури меню та концептів основних об’єктів.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ ЗАСТОСУНКУ ДЛЯ КЛАСИФІКАЦІЇ ТА МОНІТОРИНГУ ПЕРСОНАЛЬНИХ ВИТРАТ

3.1 Вибір мови програмування та фреймворку для розробки застосунку

Перед початком розробки застосунку для класифікації та моніторингу персональних витрат треба обрати найбільш підходящу мову програмування. Ту мову, яка дасть можливість розробити зручний користувацький інтерфейс, з підтримкою графічного відображення великих масивів даних.

Далі представлено короткий огляд мов програмування, які мають функціонал для створення подібних додатків.

Java - це мова програмування, яка використовується під час розроблення додатків . Це об'єктно-орієнтована мова програмування, на синтаксис якої вплинув C++. Основні цілі Java - бути простою, об'єктно-орієнтованою, надійною, безпечною та високорівневою мовою [19].

Основні призначення Java є наступні:

- вона є однією з основних мов програмування для розробки мобільних додатків на платформі Android. Вона використовується разом з Android SDK (Software Development Kit) для створення додатків для Android-пристроїв;
- розробка настільних додатків: Java має фреймворки, такі як JavaFX та Swing, які дозволяють створювати графічні інтерфейси для настільних додатків. Це дозволяє розробляти кросплатформові додатки, які можуть працювати на різних операційних системах, таких як Windows, macOS і Linux;
- розробка веб-додатків: Java має широке застосування у веб-розробці. За допомогою Java-фреймворків, таких як JavaServer Faces (JSF), Spring MVC або Apache Struts, можна створювати веб-додатки та веб-сервіси.

Переваги:

- кросплатформенність: додатки Java можуть запускатися на різних операційних системах, таких як Windows, macOS і Linux;
- велика спільнота: Java має велику спільноту розробників, що означає наявність багатьох ресурсів, бібліотек та підтримки;
- широкі можливості: Java має багатий набір бібліотек та інструментів для розробки додатків.

Недоліки:

- порівняно великий обсяг коду: розробка додатків на Java може потребувати більшої кількості коду порівняно з іншими мовами;
- швидкість: деякі розробники вказують на те, що Java може бути менш швидкою порівняно з іншими мовами.

C# - це мова програмування, розроблена компанією Microsoft. Вона є частиною платформи .NET і має широкий спектр застосувань [20]. Основні призначення мови C# включають:

- розробку додатків для Windows: C# використовується для розробки настільних додатків для операційної системи Windows. За допомогою фреймворків, таких як Windows Forms або Windows Presentation Foundation (WPF), можна створювати графічний інтерфейс та функціональні додатки для Windows;
- розробку веб-додатків: за допомогою фреймворків ASP.NET та ASP.NET Core, C# можна використовувати для розробки веб-додатків, веб-сервісів та динамічних веб-сторінок. C# дозволяє створювати потужні та масштабовані веб-додатки з використанням багатого набору бібліотек та інструментів;
- розробку мобільних додатків: C# використовується для розробки мобільних додатків для платформи Xamarin, яка дозволяє створювати додатки для Android та iOS з використанням спільного коду. Це дозволяє розробникам ефективно використовувати C# для створення кросплатформових мобільних додатків.

Переваги:

- строга типізація та безпека: C# має строгу типізацію, що допомагає виявляти помилки на ранніх етапах розробки. Вона також має вбудовані механізми безпеки, що дозволяють запобігати певним видам атак;
- велика екосистема: C# має велику спільноту розробників, а також багатий набір бібліотек та інструментів, які полегшують розробку програмного забезпечення;
- інтеграція з платформою .NET: C# використовується разом з платформою .NET, що надає багато готових компонентів та функціональності для розробки додатків.

Недоліки:

- залежність від платформи Windows: Розробка на C# обмежена платформою Windows, тому додатки, розроблені на C#, можуть працювати лише на операційній системі Windows. Хоча є певні можливості для розробки кросплатформових додатків з використанням C# (наприклад, за допомогою фреймворку Xamarin), але вона не має такої широкої підтримки кросплатформових рішень, як деякі інші мови програмування.

Python - це високорівнева, інтерпретована мова програмування, яка здобула значну популярність завдяки своїй простоті, ефективності та широкому спектру застосувань [21-22]. Основні призначення мови Python включають:

- розробка веб-додатків: Python має ряд фреймворків, таких як Django та Flask, які дозволяють легко створювати вебдодатки та вебсервіси. Вони надають гнучкість, швидкість розробки та багато функціональних можливостей для створення сучасних вебдодатків;
- аналіз даних та наукові обчислення: Python є однією з найпопулярніших мов програмування для аналізу даних та наукових обчислень. Завдяки бібліотекам, таким як NumPy, Pandas, SciPy та Matplotlib. Python надає потужні інструменти для обробки, візуалізації та аналізу даних;
- машинне навчання та штучний інтелект: Python використовується для розробки моделей машинного навчання та алгоритмів штучного інтелекту. Бібліотеки, такі як TensorFlow, PyTorch та Scikit-learn, надають широкі

можливості для реалізації складних моделей машинного навчання та глибинного навчання.

Переваги:

- простота та читабельність коду: Python має простий та зрозумілий синтаксис, що дозволяє розробникам швидко розуміти та писати код.
- багата екосистема: Python має велику спільноту розробників та широкий вибір сторонніх бібліотек, що розширюють його функціональність і полегшують розробку.
- переносимість: Python підтримується на різних платформах, включаючи Windows, macOS та різні дистрибутиви Linux.

Недоліки:

- швидкодія: Python може бути менш продуктивним у виконанні обчислювально важких завдань порівняно з деякими іншими мовами програмування;
- менша підтримка низькорівневого програмування: Python зазвичай використовується на високорівневому програмуванні, і має обмежену підтримку низькорівневих операцій, таких як маніпулювання пам'яттю.

Для розробки було обрано мову програмування Python через наступне.

Об'єктно-орієнтоване програмування. Основна перевага використання Python для розробки програмного забезпечення полягає в тому, що він забезпечує концепції ООП (об'єктно-орієнтованого програмування) і є розширюваним, масштабованим і адаптованим. До нього додається багата бібліотека стандартних шаблонів проєктування та інших найкращих практик. Він краще адаптується до зростання додатків, оскільки є відкритим вихідним кодом. Це дає змогу створювати модульні проекти та багаторазово використовуваний код.

Мова програмування з відкритим вихідним кодом. Ця дивовижна мова програмування також надає велику колекцію бібліотек з відкритим вихідним кодом, які суттєво прискорюють процедуру створення застосунків.

Python має сторонню бібліотеку графічного користувацького інтерфейсу PyQt. PyQt – є зв’язком між python та фреймворком Qt, розробленим Qt Company. Це дозволяє розробникам створювати настільні програми з багатим графічним інтерфейсом користувача (GUI) за допомогою бібліотеки Qt і мови програмування Python. Також PyQt включає в себе додаток Qt Designer, який є інструментом візуального проектування і дозоляє підключати елементи інтерфейсу до коду Python. Що дозволяє прискорити процес реалізування інтерфейсу, а також відокремити частини коду з інтерфейсом від логіки програми.

Python має сторонню бібліотеку графічних зображеннь matplotlib, яка дозволяє створювати широкий спектр статичних, анімованих та інтерактивних візуалізацій. Matplotlib пропонує широкі можливості налаштування для адаптації зовнішнього вигляду графіків.

Бібліотека pyodbc дозволяє програмам Python підключатися до широкого кола баз даних, у тому числі таких популярних, як Microsoft SQL Server, MySQL, Microsoft Access, Oracle, SQLite та багатьох інших, які підтримують драйвери ODBC. Він забезпечує послідовний і уніфікований інтерфейс для роботи з базами даних, незалежно від базової СУБД.

3.2 Реалізація дизайну інтерфейсу застосунку

Розробку застосунку доцільно почати з реалізації графічного інтерфейсу, щоб при розробці логічної частини можна було б опиратися на вже створені віджети інтерфейсу.

Для цього знадобиться інструмент візуального дизайну під назвою Qt Designer. Qt Designer сумісний з багатьма мовами програмування за допомогою спеціальних сторонніх бібліотек, для мови програмування Python такою бібліотекою є PyQt.

Структура додатку складається з основного вікна, в якому можна переглянути таблиці і звіти з даними, а також трьох допоміжних вікон, які слугують для введення або підтвердження даних.

Головне вікно у верхній частині має такі віджети, як ComboBox з вибором доступних користувачу рахунків, Label для виводу суми на рахунку, та кнопку ToolButton відкриваючу сповіщення про перевищення порогів витрат (при їх наявності). Сповіщення відображаються віджетом TextEdit. Більшу частину вікна займає TabWidget, що дає можливість розбити частини додатку на категорії і лаконічно вкласти їх у вкладки відповідні своїй ролі до цього віджету. Що буде корисно, як при розробці логічної частини так і користувачу при використанні програмного засобу у дії. На рисунку 3.1 показане створене за допомогою Qt Designer головне вікно застосунку для класифікації та моніторингу персональних витрат.

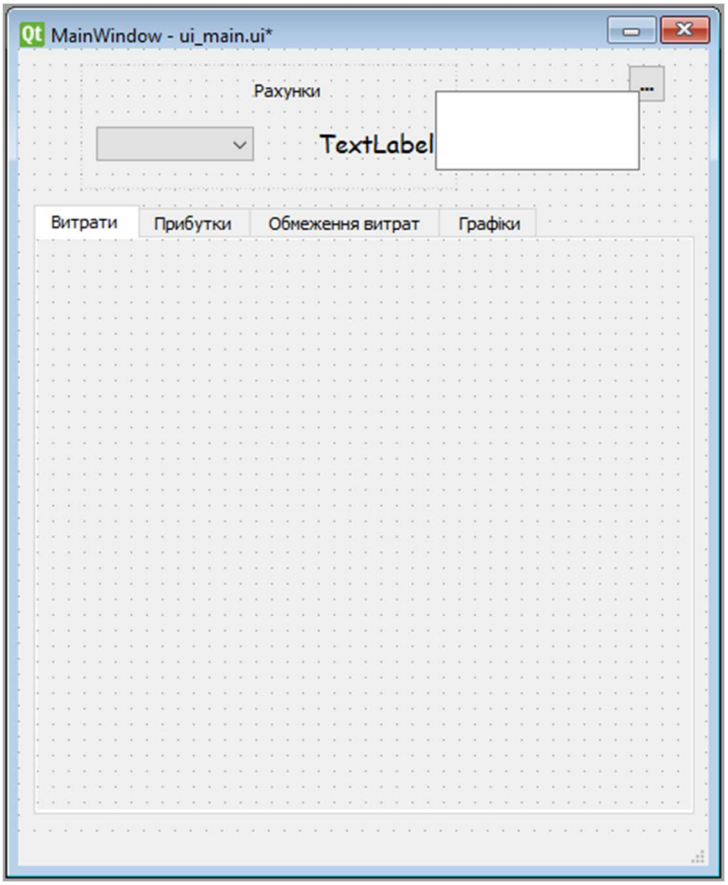

Рис. 3.1. Головне вікно застосунку для класифікації та моніторингу персональних витрат

Наступним кроком буде розташування об’єктів у відповідних вкладках. Вкладки «Витрати» та «Прибутки» мають ідентичну структуру і включають: ComboBox для перегляду даних за обраний період, Label для запису в нього обраної дати або проміжку дат, два віджети DateEdit для вибору дат по яким відображаються дати в таблиці, кнопка PressButton, що відкриває вікно додавання нових записів, дві кнопки ToolButton для редагування та видалення даних з таблиці, віджет TableView для виведення даних витрат за обраним періодом. Відповідні вкладки були створені за допомогою Qt Designer (див. рис. 3.2та 3.3).

Вкладка «Обмеження витрат» має такі об’єкти: два ComboBox для вибора періода та категорії обмеження, LineEdit для вводу назви обмеження, DoubleSpinBox для вводу суми обмеження, кнопка PressButton для запису нового обмеження за введеними даними, дві кнопки ToolButton для редагування та видалення даних з таблиці, віджет TableView для виведення даних обмеженнь витрат.

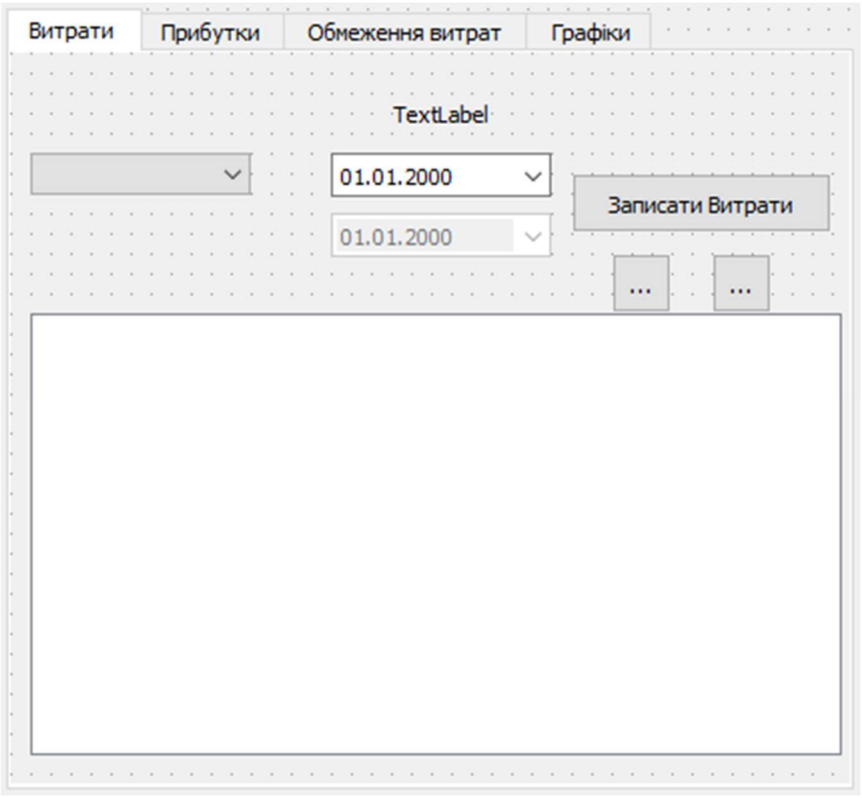

Рис. 3.2. Вкладка «Витрати» у середовищі Qt Designer

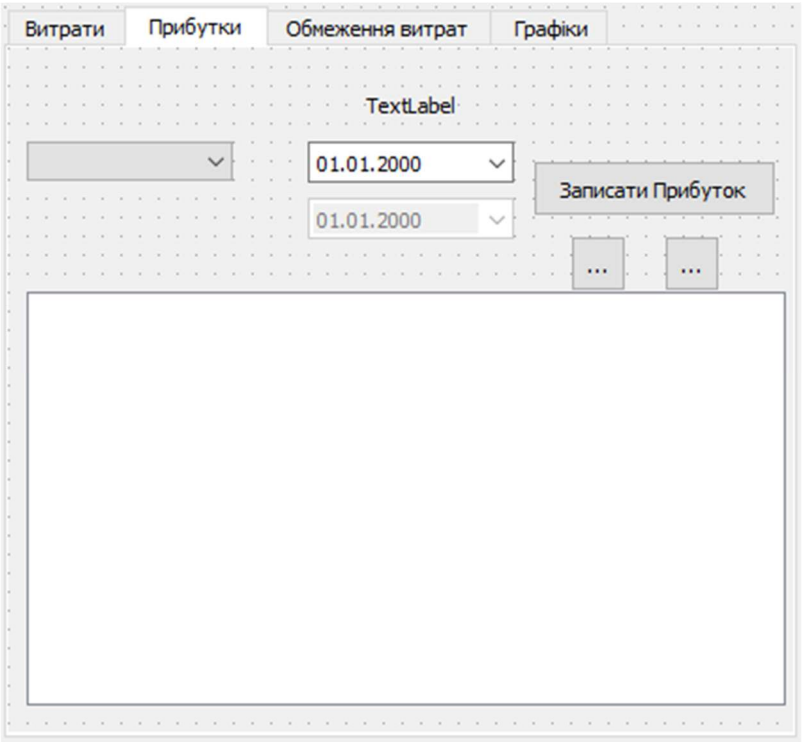

Рис. 3.3. Вкладка «Прибутки» у середовищі Qt Designer

За допомогою Qt Designer також було створено вкладку «Обмеження витрат»(див. рис. 3.4).

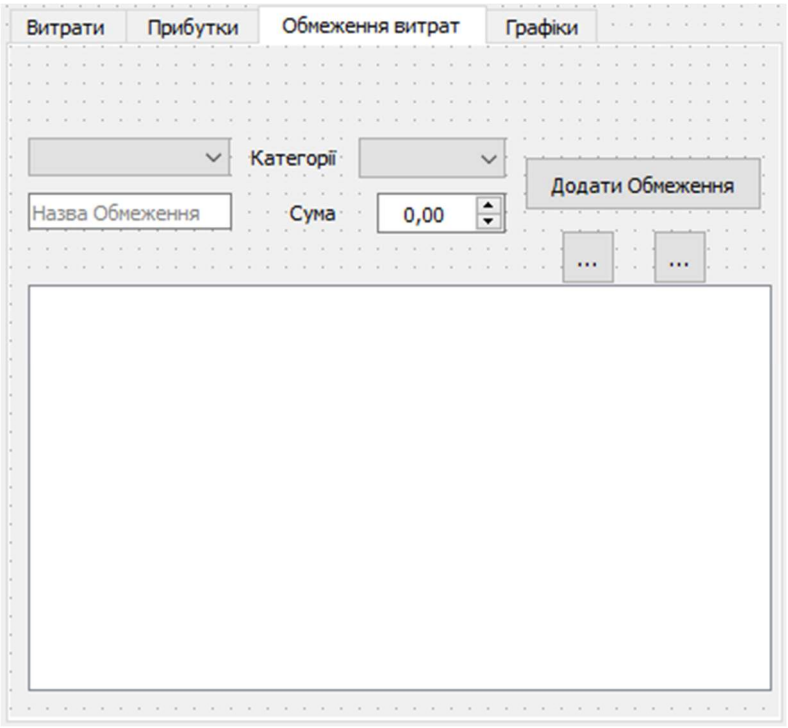

Рис. 3.4. Вкладка «Обмеження витрат» у середовищі Qt Designer

Вкладка «Графіки» включає в себе такі об’єкти: два CheckBox для відображення на діаграмі лінії обмеження витрат та для приховування пустих стовпців, чотири ComboBox для вибору періоду, рахунку, категорії та типу транзакції, PushButton для відображення діаграми за обраними параметрами, контейнер Widget для відображення в ньому діаграм, Label для відображення опису діаграми. Відповідну вкладку було також створено за допомогою Qt Designer (див. рис. 3.5).

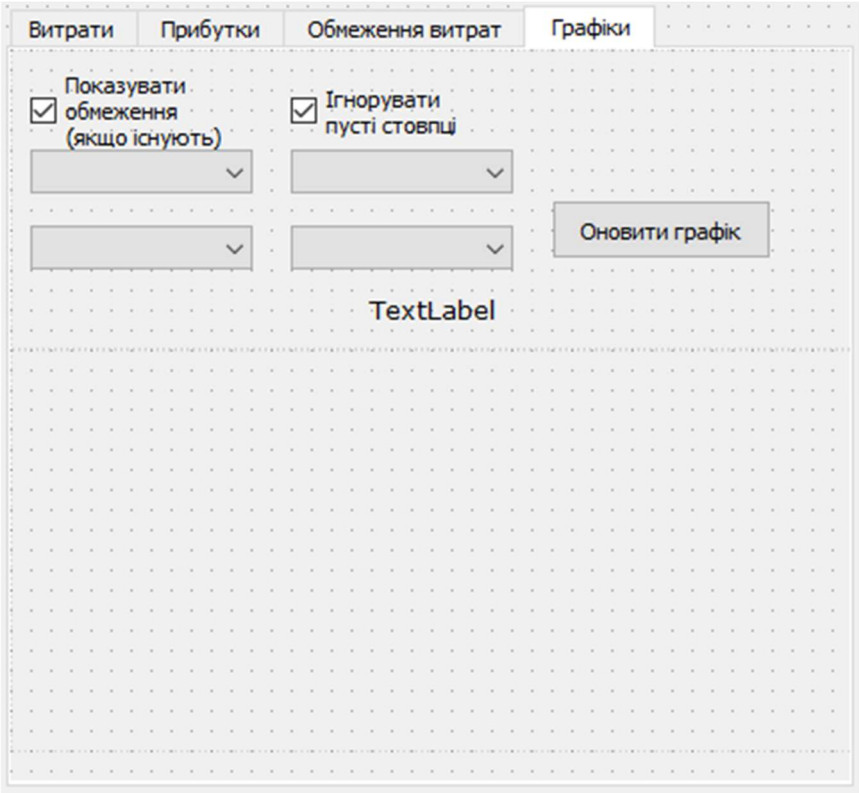

Рис. 3.5. Вкладка «Графіки» у середовищі Qt Designer

Вікно для запису нових значень витрат та прибутків має: п’ять Label для підпису віджетів, три ComboBox для вибору таких параметрів, як категорії та рахунок для якого здійснилась транзакція, DateEdit для вибору дати транзакції, DoubleSpinBox для вводу суми транзакції, LineEdit для вводу назви нової категорії, два ToolButton один для переключення видимості строки вводу інший для створення нової категорії із заданою назвою, DialogButtonBox для кнопок

діалогового вікна на створення нового запису та закриття вікна. Створене за допомогою Qt Designer вікно нового запису показане на рис. 3.6.

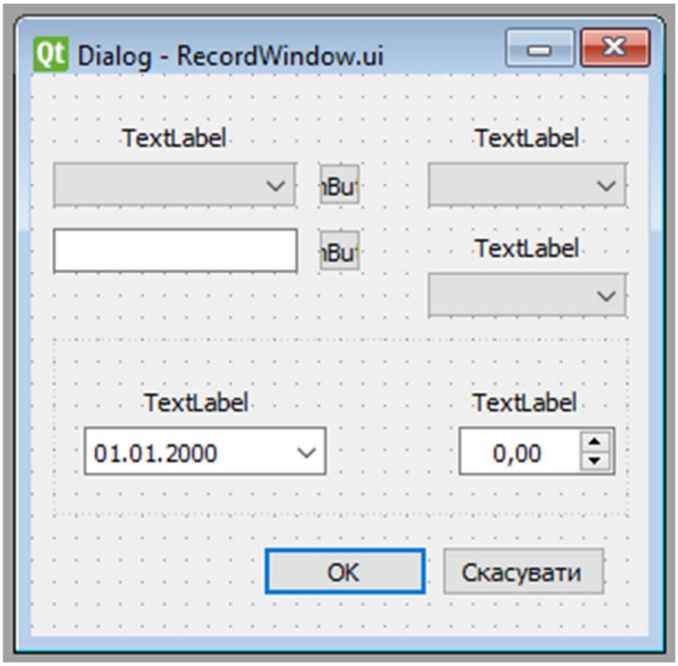

Рис. 3.6. Вікно нових записів у середовищі Qt Designer

Вікно для редагування має: вісім Label перші чотири з яких для назв параметрів, а інші для відображення значеннь параметрів до зміни, два ComboBox для вибору значеннь параметрів зі списку, DoubleSpinBox для вводу суми транзакції, LineEdit для вводу назви обмеження, DialogButtonBox для кнопок діалогового вікна на редагування запису та закриття вікна. Створене за допомогою Qt Designer вікно редагування запису показане на рис. 3.7.

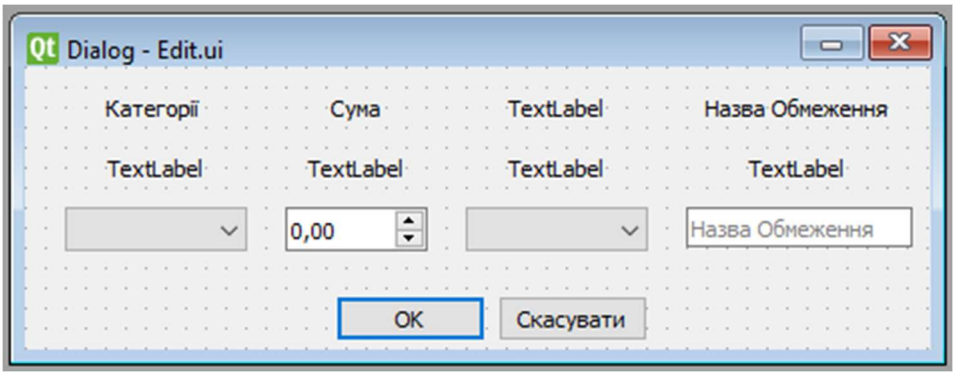

Рис. 3.7. Вікно редагування у середовищі Qt Designer

Вікно підтвердження відправки запиту до бази даних: TextEdit для виведення уточнюючої інформації про запит, DialogButtonBox для кнопок діалогового вікна на відправлення запиту запису та закриття вікна. Створене за допомогою Qt Designer вікно підтвердження запиту показане на рис. 3.8.

3.3 Реалізація основних функцій застосунку

Спочатку ініціалізуємо стандартні значення для деяких таблиць бази даних. В таблиці «СписокКатегорійВитрат» стандартними значеннями будуть: Переказ коштів, Освіта, Кафе, Продукти, Розваги, Сім’я. В таблиці «СписокКатегоріПрибутку» стандартними значеннями будуть: Переказ Коштів, Зарплатня, Стипендія, Подарунок. В таблиці «СписокПеріодів» стандартними значеннями будуть: День, Тиждень, Місяць, Рік, Період.

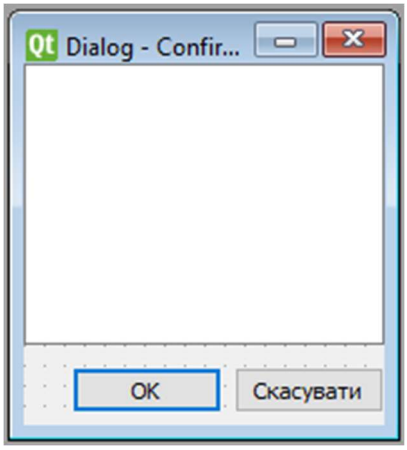

Рис. 3.8. Вікно підвердження у середовищі Qt Designer

Так, як Qt Designer створює окремі файли із класом створених об’єктів інтерфейсу, клас головного вікна створюємо окремо і вже в нього імпортуємо графічні об’єкти. Клас головного вікна є унаслідуваним від стандартного класу QMainWindow бібліотеки PyQt. В конструкторі класу створюється атрибут основних елементів інтерфейсу та додаткових вікон, ініціалізується зв’язок з базою даних за допомогою бібліотеки pyodbc, викликається функція, яка

запитами до БД отримує дані та заповнює об’єкти графічного інтерфейсу необхідними їм початковими даними. Створений клас головного вікна та його конструктор зображено на рисунку 3.9.

```
class ExpenseManager(QtWidgets.QMainWindow):
    def __init__(self):
        super(ExpenseManager, self).__init__()
        # main window
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.setWindowTitle('Window')
        self.setWindowIcon(QIcon('images/main.png'))
        #  db connection
        self.cursor = self.conn_db()
        #  additional windows
        self.addExpenseWindow = RecordWindow(self.conn, self.cursor, (self.update, self.updateRestrictions), 0)
        self.addIncomeWindow = RecordWindow(self.conn, self.cursor, (self.update, self.updateRestrictions), 1)
        self.confirmWindow = ConfirmWindow(None)
        self.editWindow = EditWindow(self.conn, self.cursor, (self.update, self.updateRestrictions))
        self.addExpenseWindow.setStyleSheet(style)
        self.addIncomeWindow.setStyleSheet(style)
        self.confirmWindow.setStyleSheet(style)
        self.editWindow.setStyleSheet(style)

        # msg box
        self.msgBox = QtWidgets.QMessageBox()
        self.msgBox.setIcon(QtWidgets.QMessageBox.Icon.Warning)
        self.msgBox.setWindowTitle('Message')
        self.msgBox.setStandardButtons(QtWidgets.QMessageBox.StandardButton.Ok)
        self.msgBox.setStyleSheet(style)
        self.init_UI()
```

Рис. 3.9. Фрагмент коду ініціалізації класу головного вікна застосунку та його конструктор

На вкладках «Витрати» та «Прибутки» присутній вибір періоду перегляду даних, вибір конкретної дати та кнопка нового запису. Кнопка нового запису відкриває вікно створення нового запису. Вибираючи різні періоди у випадаючому списку вибір дати буде означати вибір обраного періоду, який містить обрану дату. Таким чином:

- при періоді «День» дата визначає конкретний день;
- при періоді «Тиждень» дата визначає початок і кінець тижня який містить в собі обраний день;
- при періоді «Місяць» дата визначає початок і кінець місяця який містить в собі обраний день;
- при періоді «Рік» дата визначає весь рік який містить в собі обраний день;

- при періоді «Період» з’являється додаткове поле вибору дати, що дає можливість обрати потрібний відрізок часу.

Без необхідності підвердження додаток реагує на кожну зміну випадаючого списку чи поля введення дати і оновлює дані, що відображаються у таблиці та текстове поле відображаюче обраний проміжок часу. Функцію, що оновлює таблицю показано на рисунку 3.10.

```
def onCBchange(self):
    tab = self.ui.tabWidget.currentIndex()
    accName = self.ui.accountComboBox.currentText()
    if tab == 0:
        self.ui.dateEdit.setDate(datetime.now())
        self.ui.dateEdit_2.setDate(datetime.now())
        date = self.ui.dateEdit.dateTime().toPyDateTime()
        currCB = self.ui.periodComboBox

        currLabel = self.ui.periodLabel
        dateEditBot = self.ui.dateEdit_2
        tableName = 'Витрати'
        pass
    elif tab == 1:
        self.ui.dateEdit_3.setDate(datetime.now())
        self.ui.dateEdit_4.setDate(datetime.now())
        date = self.ui.dateEdit_3.dateTime().toPyDateTime()
        currCB = self.ui.periodComboBox_2
        currLabel = self.ui.periodLabel_2
        dateEditBot = self.ui.dateEdit_4
        tableName = 'Прибуток'
        pass
    else:
        return None
    if currCB.currentText() == 'День':
        if dateEditBot.isEnabled():
            dateEditBot.setEnabled(False)
            dateEditBot.setVisible(False)
        request = f"SELECT {tableName}.Категорія, {tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE {tableName}.Дата = #{date.strftime('%m-%d-%Y'
        print('table update')
        self.setTable(currTable, request)
        currLabel.setText(getDate())
    elif currCB.currentText() == 'Тиждень':
        if dateEditBot.isEnabled():
            dateEditBot.setEnabled(False)
            dateEditBot.setVisible(False)
        firstweekday = date - timedelta(days=datetime.weekday(date))
        lastweekday = firstweekday + timedelta(days=6)
        request = f"SELECT {tableName}.Категорія, {tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE {tableName}.Дата >= #{firstweekday.strftime('
        print('table update')
        self.setTable(currTable, request)
        currLabel.setText(f'{firstweekday.strftime("%d-%m-%Y")} - {lastweekday.strftime("%d-%m-%Y")}')
    elif currCB.currentText() == 'Місяць':
        dictMonth = {1: 'Січень', 2: 'Лютий', 3: 'Березень', 4: 'Квітень', 5: 'Травень', 6: 'Червень', 7: 'Липень',
                     8: 'Серпень', 9: 'Вересень', 10: 'Жовтень', 11: 'Листопад', 12: 'Грудень', }
        if dateEditBot.isEnabled():
            dateEditBot.setEnabled(False)
            dateEditBot.setVisible(False)
        firstmonthday = date - timedelta(days=date.day - 1)
        next_month = date.replace(day=28) + timedelta(days=4)
        lastmonthday = next_month - timedelta(days=next_month.day)
        request = f"SELECT {tableName}.Категорія, {tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE {tableName}.Дата >= #{firstmonthday.strftime(
        print('table update')
        self.setTable(currTable, request)
        currLabel.setText(f"{dictMonth[date.month]} {date.year}")
    elif currCB.currentText() == 'Рік':
        if dateEditBot.isEnabled():
            dateEditBot.setEnabled(False)
            dateEditBot.setVisible(False)
        request = f"SELECT {tableName}.Категорія, {tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE YEAR({tableName}.Дата) = YEAR(#{getDate()}#)
        print('table update')
        self.setTable(currTable, request)
        currLabel.setText(date.strftime("%Y рік"))
    elif currCB.currentText() == 'Період':
        currLabel.setText('Період')
        if not dateEditBot.isEnabled():
            dateEditBot.setEnabled(True)
            dateEditBot.setVisible(True)
```

Рис. 3.10. Функція оновлення таблиці інтерфейсу

У вікні створення нового запису присутні поля введення даних, кнопка переключення видимості поля та кнопки нової категорії, кнопка збереження нової категорії, кнопки створення нового запису та виходу з вікна.

Створений клас вікна запису нових даних та його конструктор представлено на рисунку 3.11. Функція додавання нової категорії представлена на рисунку 3.12.

```
class RecordWindow(QtWidgets.QDialog):
    def __init__(self, db, dbCurs, mainUpdate, tab):
        super(RecordWindow, self).__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.db = db
        self.dbCursor = dbCurs
        self.mainUpdate = mainUpdate
        self.tab = tab
        # additional windows
        self.confirmWindow = ConfirmWindow(self.close)
        self.msgBox = QtWidgets.QMessageBox()
        self.msgBox.setIcon(QtWidgets.QMessageBox.Icon.Warning)
        self.msgBox.setWindowTitle('Message')
        self.msgBox.setStandardButtons(QtWidgets.QMessageBox.StandardButton.Ok)
        self.confirmWindow.setStyleSheet(style)
        self.msgBox.setStyleSheet(style)
        self.initUI()
```

Рис. 3.11. Код ініціалізації класу вікна створення нового запису та його конструктор

```
def addNewCategory(self):
    name = self.ui.lineEdit.text()
    if name != '':
        tableName = f"СписокКатегорій{self.getTableName(self.tab,1)}"
        request = f"SELECT {tableName}.Назва FROM {tableName} WHERE {tableName}.Назва='{name}';"
        if self.dbCursor.execute(request).fetchone() is None:
            request = f"INSERT INTO {tableName} (Назва) " \
                      f"VALUES ('{name}');"
            print(request)
            label = f"Ви бажаєте додати у \n{tableName}\n\n" \
                    f"Категорію: {name}"
            self.ui.categoryComboBox.addItem(name)
            self.ui.lineEdit.setText('')
            self.confirmWindow.reset(self.simpleRequest, request, label, (*self.mainUpdate, self.reset), parentClose=False)
            self.confirmWindow.show()
            pass
        else:
            self.showMsg('Така категорія вже існує.')
    else:
        self.showMsg('Назву категорії не вказано.')
```

Рис. 3.12. Функція додавання нової категорії

На вкладці «Обмеження витрат» присутні поля для вводу даних, кнопка створення обмеження та таблиця з обмеженнями. Таблиця завжди показує всі наявні обмеження і оновлюється при доданні нових чи видаленні/зміні старих записів. При натисненні кнопки створення нового обмеження дані перевіряються на коректність. Якщо дані коректні, тоді відкривається вікно підтвердження, інакше з’являється вікно з повідомленням про природу некоректності даних.

При запуску додатку або зміненні даних у таблиці з обмеженнями викликається функція, яка перераховує витрачені суми за існуючими обмеженнями. При наближенні витрачених сум до порогів витрат, виникають відповідні сповіщення з кольоровим виділенням в залежності від ступеню наближення витрат до порогу записуються до блоку з повідомленнями. Якщо витрати складають від 75% до 90% порогу, тоді сповіщення виділиться жовтим, якщо від 90% до 100% порогу , тоді – помаранчевим, а при перевищенні порогу – червоним. Також символ на кнопці переключення видимості блоку сповіщенньь змінює колір відповідно до найбільшої наявної витрати до порогу. Наприклад: якщо є декілька витрат у жовтій та помаранчевій зоні, тоді символ на кнопці прийме помаранчевий колір, а якщо витрат, що наближаються до порога, немає, тоді символ стане сірим і блок сповіщенньь буде пустим. Функція що перевіряє відношення витрат до обмеженньь представлена на рисунку 3.13.

```
def checkRestriction(self, name, period, category, summ, date=None, accName=None):
    if date is None:
        date = datetime.now()
    if period == 'День':
        dateStr = f"Витрати.Дата = #{date.strftime('%m-%d-%Y')}#"
    elif period == 'Тиждень':
        firstweekday = date - timedelta(days=datetime.weekday(date))
        lastweekday = firstweekday + timedelta(days=6)
        dateStr = f"Витрати.Дата >= #{firstweekday.strftime('%m-%d-%Y')}# and Витрати.Дата <= #{lastweekday.strftime('%m-%d-%Y')}#"
        pass
    elif period == 'Місяць':
        firstmonthday = date - timedelta(days=date.day - 1)
        next_month = date.replace(day=28) + timedelta(days=4)
        lastmonthday = next_month - timedelta(days=next_month.day)
        dateStr = f"Витрати.Дата >= #{firstmonthday.strftime('%m-%d-%Y')}# and Витрати.Дата <= #{lastmonthday.strftime('%m-%d-%Y')}#"
        pass
    elif period == 'Рік':
        dateStr = f"YEAR(Витрати.Дата) = Year(#{date.strftime('%m-%d-%Y')}#)"
        pass

    if accName is None:
        accStr = ''
    else:
        accStr = f" and Витрати.НазваРахунку = '{accName}'"

    request = f"SELECT Sum(Витрати.Сума) AS [Sum-Сума] FROM Витрати WHERE {dateStr} AND Витрати.Категорія='{category}'{accStr} GROUP BY Витрати.Категорія;"
    periodSum = self.cursor.execute(request).fetchone()
    if periodSum is not None: periodSum = round(periodSum[0], 2)
    print(name, periodSum)
    print(summ)
    periodLabels = {'День': 'Щоденні', 'Тиждень': 'Щотижневі', 'Місяць': 'Щомісячні', 'Рік': 'Щорічні'}
    if periodSum is None:
        return -1,''
    elif periodSum <= summ*0.75:
        return 0,''
    elif periodSum > summ*0.75 and periodSum < summ*0.9:
        return 1, f"{periodLabels[period]} витрати в категорії {category} наближаються до порога (Поріг: {summ}, Витрати: {periodSum})"
    elif periodSum >= summ*0.9 and periodSum < summ:
        return 2, f"{periodLabels[period]} витрати в категорії {category} критично наближаються до порога (Поріг: {summ}, Витрати: {periodSum})"
    else:
        return 3, f"{periodLabels[period]} витрати в категорії {category} перевищують поріг (Поріг: {summ}, Витрати: {periodSum})"
```

Рис. 3.13. Функція перевірки перевищень встановлених обмежень

На попередніх вкладках присутні кнопка редагування та кнопка видалення даних з таблиці наявної на вкладці. При натисненні кнопки видалення

відкривається вікно підтвердження. При натисненні кнопки редагування відкривається вікно для редагування обраного рядку, якщо рядків обрано декілька, тоді на редагування береться верхній у списку. Функцію для видалення рядків з таблиці показано на рис. 3.14.

```
def delTableRows(self):
    def sendReq(requests):
        for item in requests:
            print(item)
            self.cursor.execute(item)
            self.conn.commit()
    rows = sorted(set(index.row() for index in self.ui.tableView.selectedIndexes()))
    cols_num = self.ui.tableView.model().columnCount(0)
    if self.ui.tabWidget.currentIndex() == 0:
        tableName = 'Витрати'
    elif self.ui.tabWidget.currentIndex() == 1:
        tableName = 'Прибуток'
    elif self.ui.tabWidget.currentIndex() == 2:
        tableName = 'СписокОбмежень'
        cols_num = 4
    requestsToDel = []
    label = """<p align="center">Ви дійсно хочете видалити ці об'єкти:</p>"""
    if rows:
        for i in rows:
            arr = [str(self.ui.tableView.model().index(i, j).data()) for j in range(cols_num)] # 0 Категорія 1 Сума 2 Рахунок
            if '' in set(arr):
                self.showMsg('Таблиця порожня.')
                return None
            #rowsToDel.append(arr)
            label = label + ' '.join(arr) + '<br>'
            if len(arr) == 3:
                requestsToDel.append(f"DELETE FROM {tableName} WHERE {tableName}.Категорія = '{arr[0]}' and {tableName}.Сума = {arr[1]} and {tableName}.НазваРах
            else:
                requestsToDel.append(f"DELETE FROM {tableName} WHERE {tableName}.НазваОбмеження = '{arr[0]}' and {tableName}.Період = '{arr[1]}' and {tableName}.

        self.confirmWindow.reset(sendReq, requestsToDel, label, onclose=(self.update, self.updateRestrictions,), parentClose=False)
        self.confirmWindow.show()
    else:
        self.showMsg('Не виділено жодного рядка.')
```

Рис. 3.14. Фрагмент коду функції видалення рядків з таблиці інтерфейсу

У вікні редагування присутні поля введення даних, кнопки підтвердження та закриття вікна. Значення у полях введення даних при відкритті вікна співпадають з даними, що підлягають редагуванню. Створений клас вікна редагування та його конструктор показано на рисунку 3.15.

```
class EditWindow(QtWidgets.QDialog):
    def __init__(self, db, dbCurs, mainUpdate):
        super(EditWindow, self).__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.db = db
        self.dbCursor = dbCurs
        self.mainUpdate = mainUpdate
        self.confirmWindow = ConfirmWindow(self.close)
        self.attr = None
        self.tableName = None
        self.ui.buttonBox.accepted.connect(self.confirmWin)
        self.ui.buttonBox.rejected.connect(self.close)
        self.init_UI()
```

Рис. 3.15. Код ініціалізації класу вікна редагування та його конструктор

На вкладці «Графіки» присутні чекбокси та випадаючі списки для вибору пареметрів діаграми, а також кнопка оновлення графіка за обраними параметрами.

3.4 Сценарії використання застосунку

Запис витрат та прибутків. Алгоритм для запису витрат та прибутків ідентичний, тому далі буде показаний на прикладі запису витрат. Для запису витрат треба перейти на вкладку «Витрати» (див. рис. 3.16).

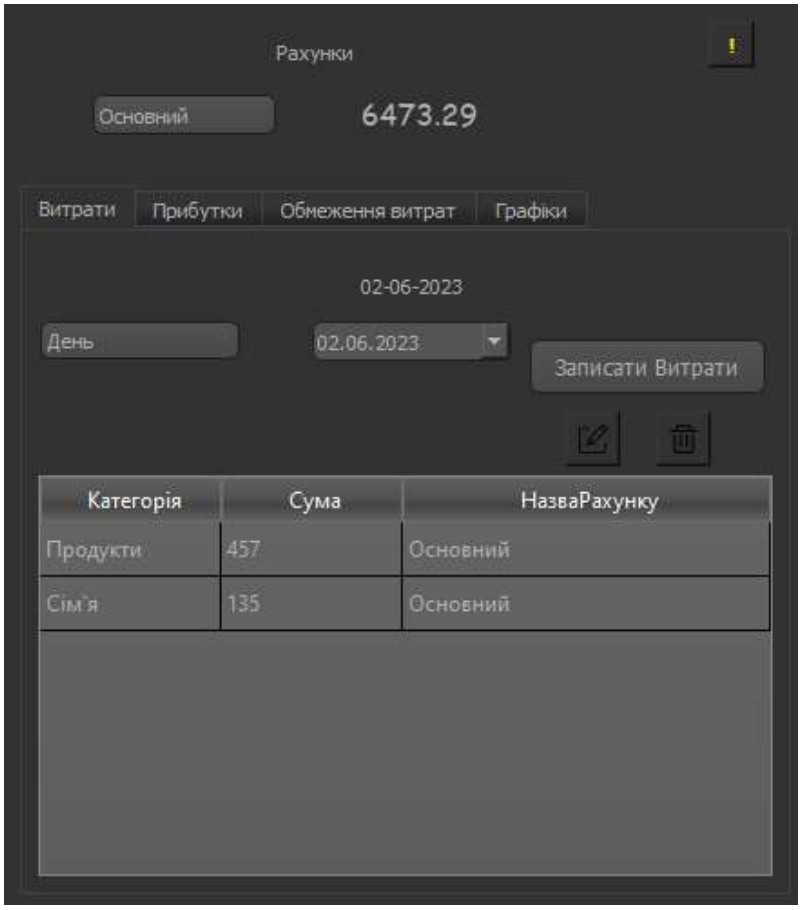

Рис. 3.16. Вкладка «Витрати» головного меню застосунку

Далі потрібно відкрити вікно нової витрати натиснувши кнопку «Записати Витрати» (див. рис. 3.17)

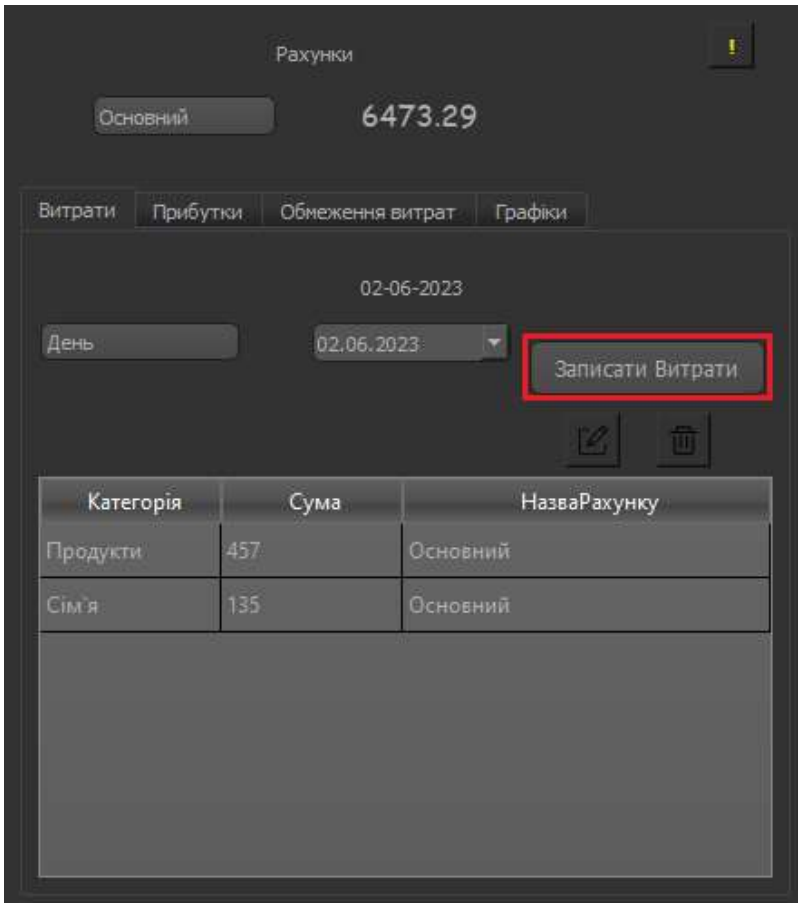

Рис. 3.17. Запис нової витрати

У відкритому вікні заповнюємо дані (див. рис. 3.18) про витрату і натискаємо «Ок» для відкриття вікна з підтвердженням.

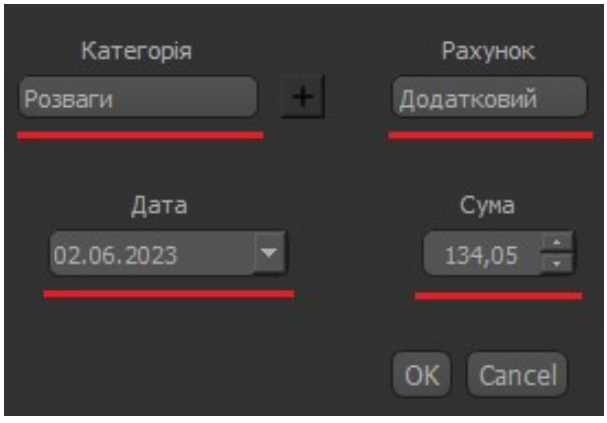

Рис. 3.18. Поля заповнення даних по витратам

Для підтвердження також треба натиснути кнопку «ОК» (див. рис. 3.19)

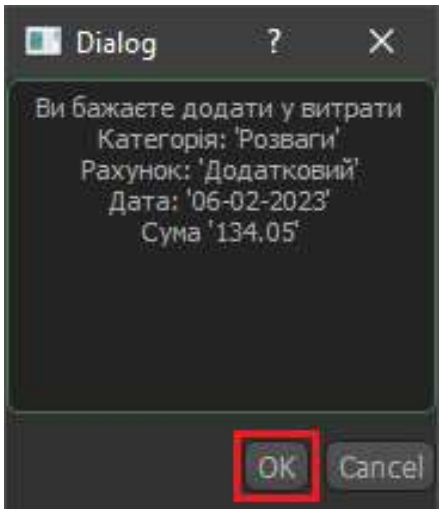

Рис. 3.19. Підтвердження відправки запиту на запис витрат

Далі доцільно розглянути створення нової категорії витрат. Алгоритм для створення нової категорії витрат та прибутків далі буде показаний на прикладі запису категорії витрат. Для цього відкриваємо вікно запису нової витрати див. рис. 3.16. У вікні натискаємо на кнопку із іконкою «+», після чого з'являється поле для запису назви категорії та кнопка запису категорії в БД (див. рис. 3.20).

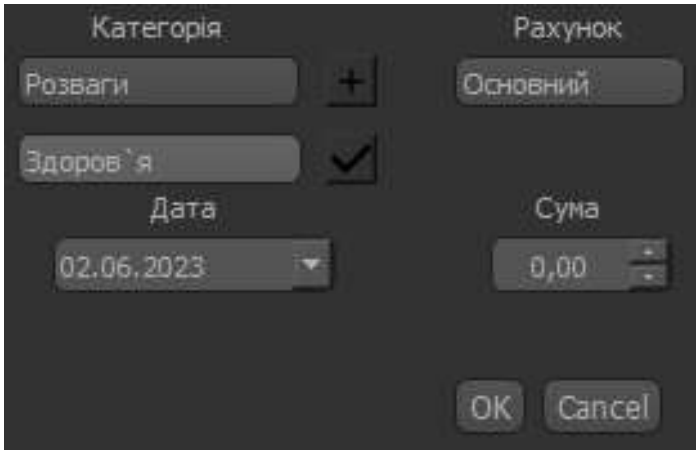

Рис. 3.20. Додавання нової категорії

Далі треба натиснути кнопку «ОК» для створення нового запису, що відкриває вікно підтвердження відправки запиту до БД, яке показане на рисунку 3.21.

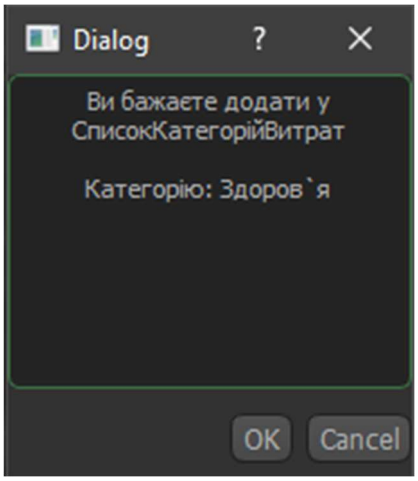

Рис. 3.21. Вікно підтвердження відправлення запиту на створення нового запису

На рисунку 3.22 показано, що категорія дійсно додалася до БД і відповідно до випадаючого списку з категоріями.

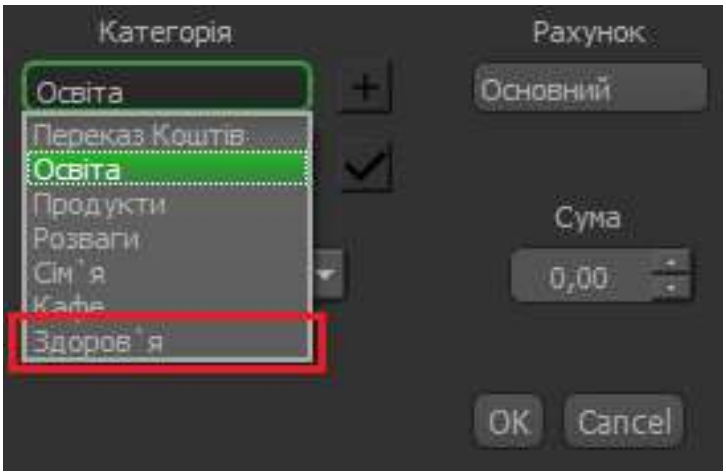

Рис. 3.22. Оновлений список з категоріями витрат

Створення обмеженнь на витрати. Для створення нового обмеженя треба перейти на вкладку «Обмеження витрат», яка показана на рисунку 3.23. Та заповнити виділені поля даними про нове обмеження, як показано на рисунку 3.24, після чого натиснути кнопку «Ок» у вікні підтвердження відправлення запиту до БД.

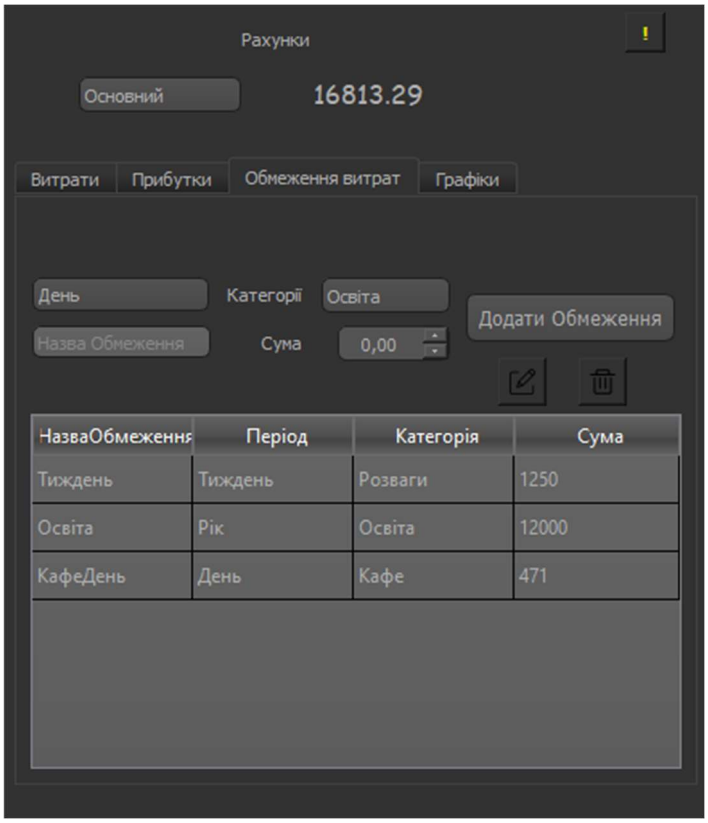

Рис. 3.23. Вкладка «Обмеження витрат»

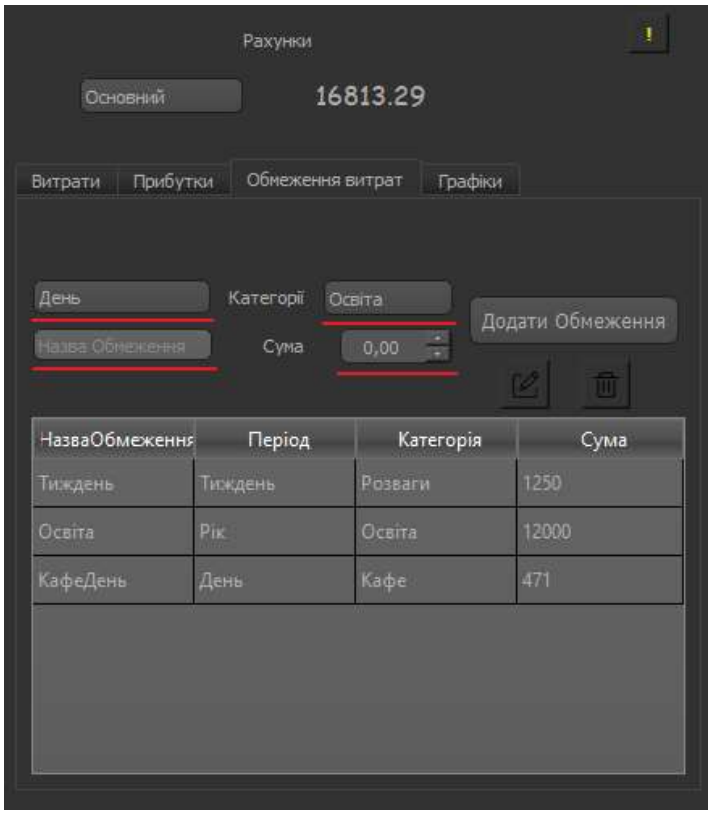

Рис. 3.24. Виділені поля вводу даних нового обмеження

На рисунку 3.25 показано, що обмеження дійсно додалося до бази даних і, відповідно, таблиці користувацького інтерфейсу.

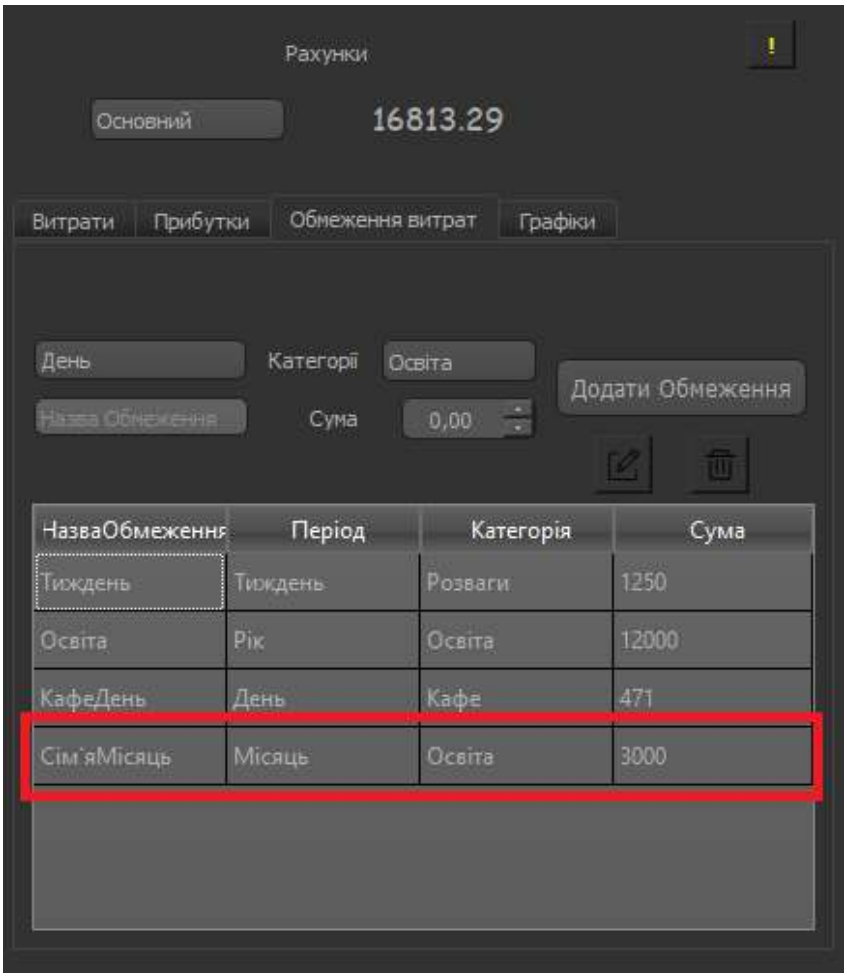

Рис. 3.25. Створений запис обмеження

Перегляд звітів. Для перегляду звітів необхідно перейти на вкладку «Графіки», яка показана на рисунку 3.26. У випадаючих списках треба обрати параметри за вподобаннями, після чого натиснути кнопку «Оновити графік» для відображення діаграм за обраними параментрами. На рисунку 3.27, в якості прикладу, показано діаграму за обраними параметрами. При цьому, чекбокс «Показувати обмеження» відображає лінію, яка відображає граничне значення встановлене користувачем для категорії та періоду у випадку, якщо воно існує. При цьому, чекбокс «Ігнорувати пусті стовпці» регулює відображення тих стовпців діаграм, які мають нульове значення.

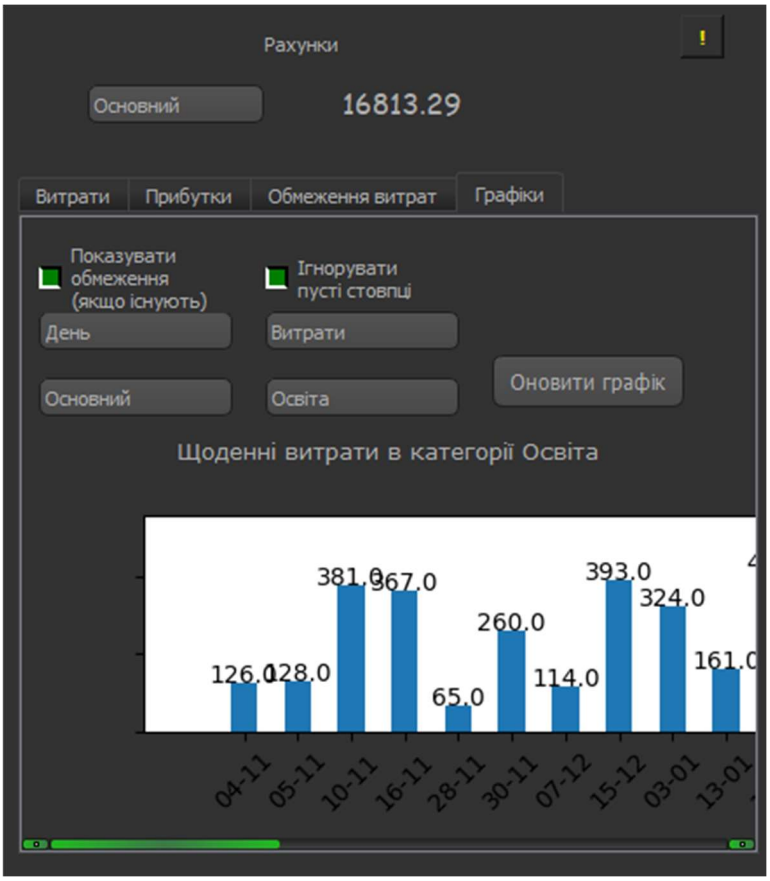

Рис. 3.26. Вкладка «Графіки»

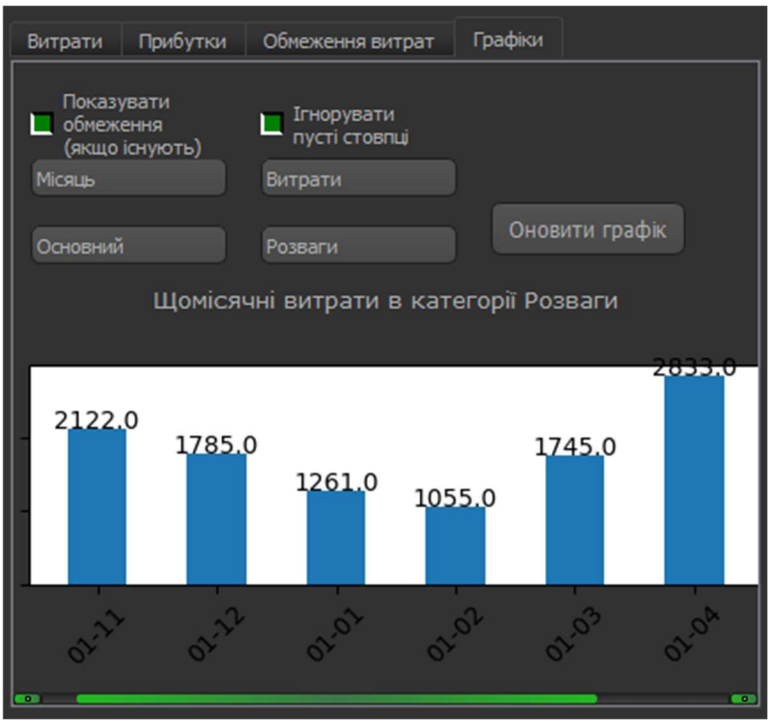

Рис. 3.27. Оновлена діаграма з заданими параметрами

Редагування даних. Для перших трьох вкладок, на яких присутні таблиці з даними є можливість редагування даних в цих таблицях. Для цього, спочатку потрібно виділити хоча б одну комірку з бажаного рядку як зображено на рисунку 3.28. Якщо виділено комірки з різних рядків, тоді з усіх рядків, в яких виділено комірки, для редагування буде обрано верхній.

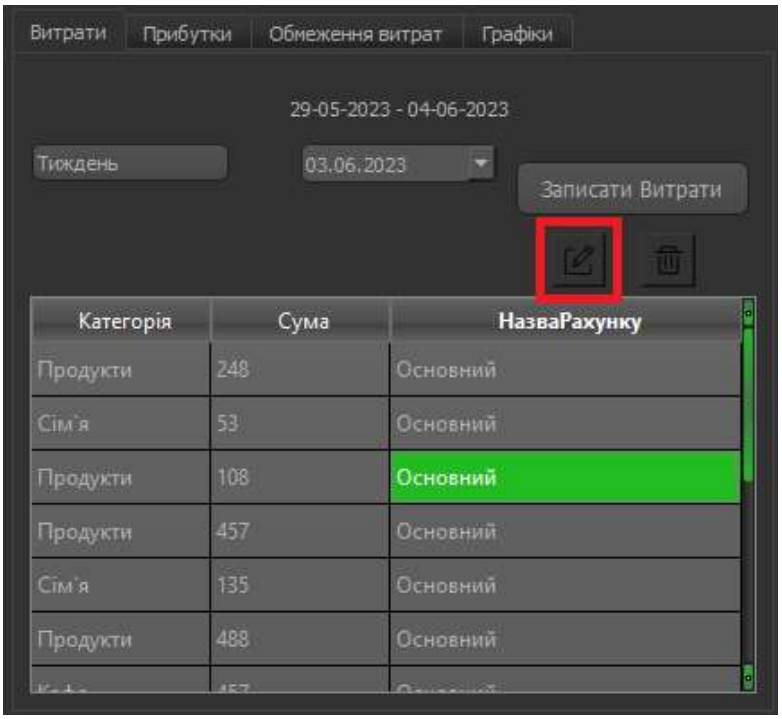

Рис. 3.28. Виділено одну комірку в рядку та кнопку редагування

Після натискання кнопки редагування (див. рис. 3.28) відкривається відповідне вікно, яке показанне на рисунку 3.29.

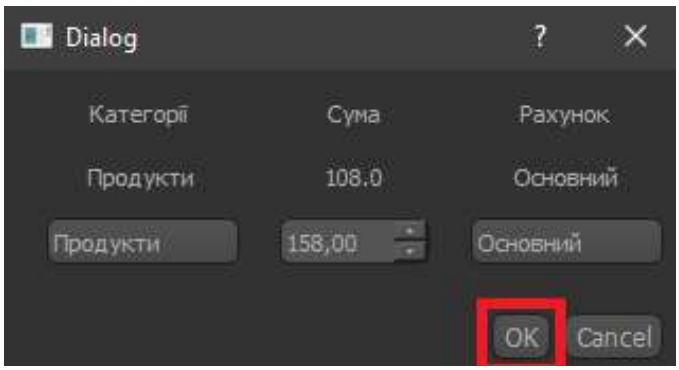

Рис. 3.29. Вікно редагування запису

Далі можна відедагувати потрібний параметр і натиснути кнопку «ОК» (див. рис. 3.29). Після цього у вікні підтвердження (див. рис. 3.30) можна перевірити дані і підтвердити зміни.

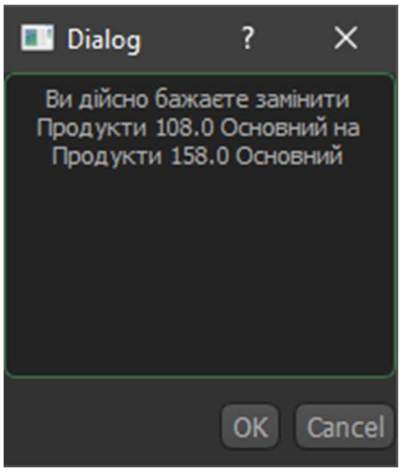

Рис. 3.30. Вікно підтвердження змін та відправки запиту до БД

На рисунку 3.31 показано, що зміна даних дійсно відбулася у базі даних і в таблиці.

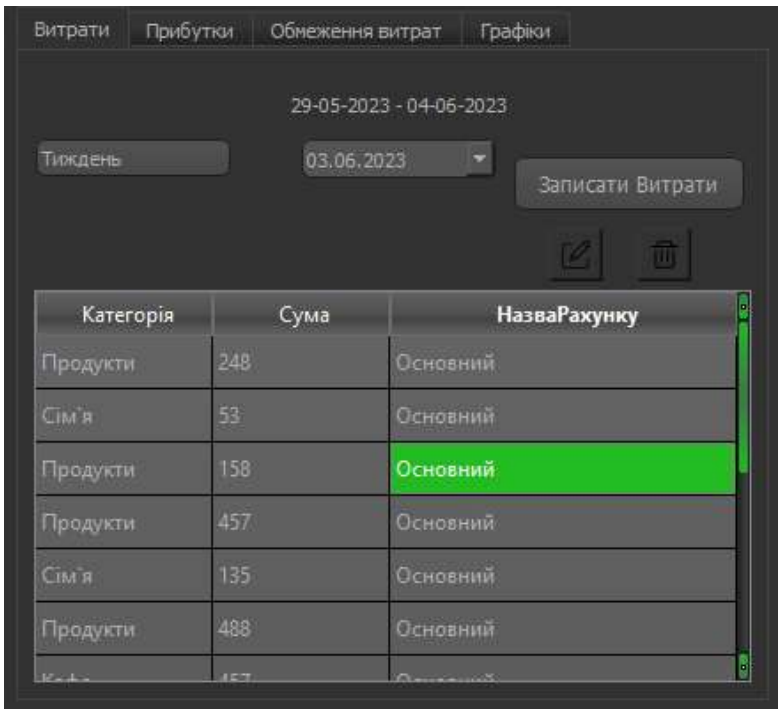

Рис. 3.31. Стан застосунку після внесення змін

Видалення даних. Для перших трьох вкладок, на яких присутні таблиці з даними є можливість видалення даних з цих таблиць. Для цього спочатку потрібно виділити хоча б одну комірку з бажаного рядку як показано на рисунку 3.32.

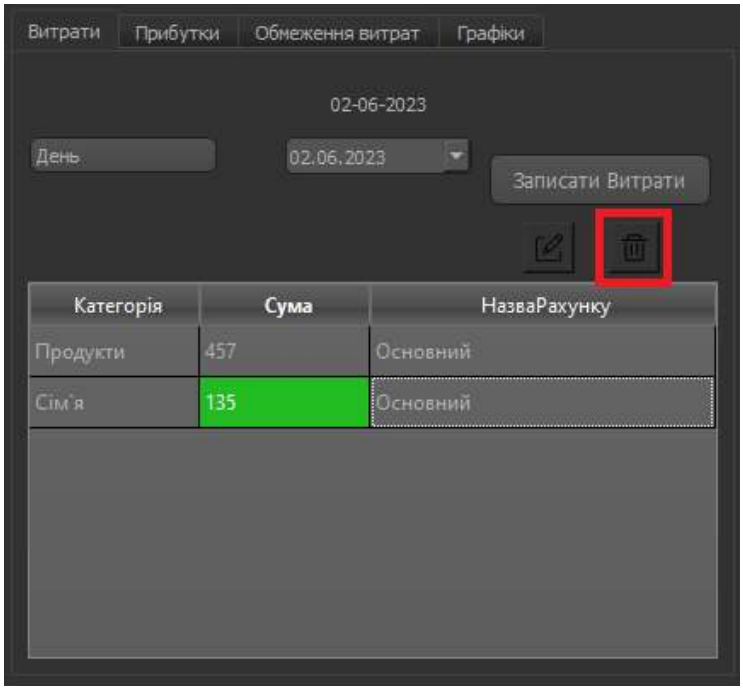

Рис. 3.32. Виділена комірка з бажаного рядку та кнопка видалення

Після натискання кнопки видалення також відкривається вікно підтвердження, в якому можна перевірити дані і підтвердити видалення натиснувши кнопку «ОК» (див. рис. 3.33).

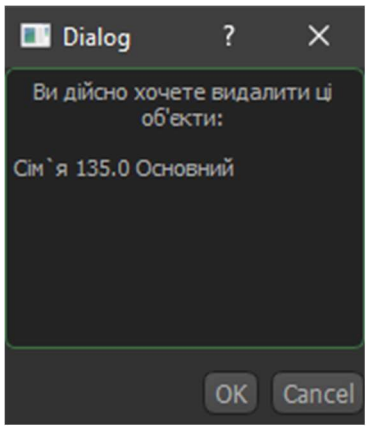

Рис. 3.33. Вікно підтвердження видалення

На рисунку 3.34 показано, що зміна даних дійсно відбулася у базі даних і в таблиці.

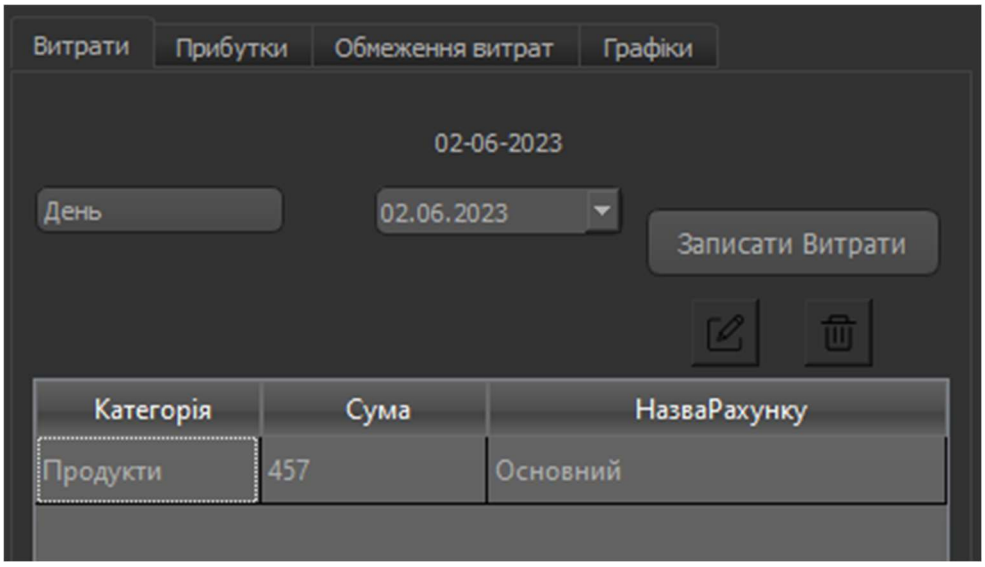

Рис. 3.34. Стан застосунку після видалення даних

Перевірка перевищення обмежень. Для перегляду перевищень обмежень треба натиснути відповідну кнопку, яка показана на рисунку 3.35.

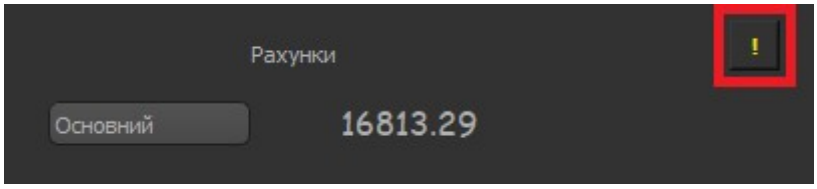

Рис. 3.35. Спеціальна кнопка сповіщень

Після натискання кнопки сповіщень відображається спеціальне поле (див. рис. 3.36), в якому відображаються витрати, які наближаються або перевищили поріг. Написи мають кольорові градації, як і символ зображений на кнопці. При цьому, колір символу на кнопці залежить від кольору напису який найбільш наблизився або перевищив поріг.

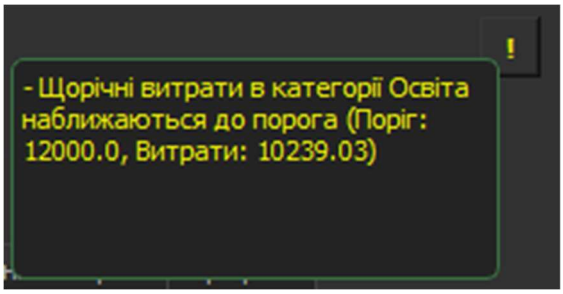

Рис. 3.36. Поле у якому відображаються сповіщення

Таким чином розроблений застосунок має достатній функціонал для ефективної класифікації та моніторингу персональних витрат. Можна стверджувати, що застосунок має практичну цінність, може бути використаний кінцевим користувачем та має перспективи для подальшого розвитку.

Висновки до розділу 3

В рамках реалізації системи обрано мову програмування Python та ряд допоміжних сторонніх бібліотек, таких як PyQt для GUI, Matplotlib для відображення даних у графічному вигляді.

На основі концепту інтерфейсу розроблено форми вікон, які використані при розробці додатку. За допомогою Microsoft Access створено базу даних і заповнено її тестовими даними. Реалізовано класи вікон: ExpenseManager, RecordWindow, EditWindow, ConfirmWindow, MatplotlibWidget. Також для цих класів створено програмні функції, за допомогою яких додаток оброблює дані та повноцінно взаємодіє з базою даних при виконанні основних операцій при класифікації та моніторингу персональних витрат.

Розроблений застосунок надає достатній функціонал, та має перспективу для подальшого розвитку та практичного впровадження.

ВИСНОВКИ

Під час виконання роботи було проаналізовано методи обліку персональних фінансів. Внаслідок чого визначили головні атрибути моніторингу фінансового стану, якими є: розуміння свого фінансового стану, встановлення фінансових цілей, контроль та оцінка результатів.

На стадії моделювання було розроблено структуру БД та ряд інформативних діаграм, які на різних рівнях описують взаємодії як між об'єктами всередині системі, так і між системою та користувачем. Також спроєктовано концепт користувацького інтерфейсу.

В рамках реалізації системи обрано мову програмування та ряд допоміжних сторонніх бібліотек. На основі концепту інтерфейсу розроблено форми вікон, які будуть використовуватися для розробки додатку. Створено базу даних і заповнено її тестовими даними. Реалізовано класи вікон та функції за допомогою яких додаток оброблює дані та взаємодіє з БД.

Результатом роботи є система класифікації та моніторингу персональних витрат, за допомогою якої можна проводити облік витрат користувача у вигляді програмного засобу для системи Windows. Додатком можна користуватись автономно, він приймає дані витрат та прибутків користувача. Наявна можливісь встановлювати обмеження витрат на певні відрізки часу для обраної категорії, що поліпшує процес спостереження за витратами. Також дані зберігаються у БД, що дозволяє користувачу аналізувати свій фінансовий стан у різних проміжках часу, переглядаючи звіти у вигляді діаграм.

У додаток закладено основи методів і підходів до класифікації та моніторингу персональних витрат. Це робить додаток надзвичайно корисним при менеджменті персональних фінансів.

Отже, програмний засіб, створений у рамках цієї роботи є повноцінним засобом для контролю та аналізу персональних витрат, і також він може бути основою для подальшої розробки кращого та мультиплатформенного додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Костирко Р.О. Фінансовий аналіз: навч. посіб., 2007. 784 с.
2. Кійосакі Р.Т. Квадрант грошового потоку, 2019. 368 с.
3. Л.О. Пташенко, Віталій Сержанов Фінансовий контролінг, 2021. 344 с.
4. Рендал Рей Сучасна теорія грошей, 2017. 480 с.
5. Клейсон Дж. С. Найбагатший чоловік у Вавилоні, 2017. 136 с.
6. Хоакім де Посада, Еллен Сінгер Не накидайтесь на мармелад, 2013. 160 с.
7. Тарасова А. Сам собі фінансист, 2018. 190 с.
8. Ха-Юн Чанг Економіка. Інструкція з використання, 2016. 400 с.
9. PyQt5 Reference Guide URL: https://www.riverbankcomputing.com/static/Docs/PyQt5/ (дата звернення: 19.05.2023)
10. Python 3.11.3 Documentation URL: https://docs.python.org/3/ (Дата звернення: 20.05.2023)
11. Володимир Гайджарі, Ігор Ізварін Бази даних в інформаційних системах, 2018. 418 с.
12. Jenifer Tidwell Designing Interfaces: Patterns for Effective Interaction Design, 2020. 599 p.
13. Еллен Лаптон, Дженніфер Коул Філліпс Графічний дизайн. Нові основи, 2020. 264 с.
14. Скотт Берінато Хороші діаграми. Поради, інструменти та вправи для кращої візуалізації даних, 2022. 288 с.
15. Алан Больє Вивчаємо SQL. Генерація, вибірка та обробка даних. 3-тє видання, 2023. 400 с.
16. Донелла Медовз Мистецтво мислити системно. Розв'язання проблем від особистого до глобального масштабу, 2023. 304 с.
17. Book accounts URL: https://book-accounts.en.softonic.com/? (Дата звернення: 22.05.2023)

18. Moneyble Personal Finance URL: https://moneyble-personal-finance.en.softonic.com/? (Дата звернення: 22.05.2023)

19. Для чого використовується Java: 12 реальних програм Java URL: https://uk.myservername.com/what-is-java-used (Дата звернення: 22.05.2023)

20. Основні відомості про мову програмування C# URL: https://armedsoft.com/ua/blog/osnovni-vidomosti-pro-movu-programuvannya-c (Дата звернення: 22.05.2023)

21. Розробка сайтів та додатків на Python / Django URL: https://secl.com.ua/all-solutions/python/ (Дата звернення: 22.05.2023)

22. Що таке Python – можливості мови програмування URL: https://freehost.com.ua/ukr/faq/wiki/chto-takoe-jazik-programmirovanija-python/ (Дата звернення: 22.05.2023)

Додаток А1

Код для класу основного вікна

```
class ExpenseManager(QtWidgets.QMainWindow):
    def __init__(self):
        super(ExpenseManager, self).__init__()
        # main window
        self.ui = Ui_MainWindow()
        self.ui.setupUi(self)
        self.setWindowTitle('Window')
        self.setWindowIcon(QIcon('images/main.png'))
        #  db connection
        self.cursor = self.conn_db()
        #  additional windows
        self.addExpenseWindow = RecordWindow(self.conn,
self.cursor, (self.update, self.updateRestrictions), 0)
        self.addIncomeWindow = RecordWindow(self.conn,
self.cursor, (self.update, self.updateRestrictions), 1)
        self.confirmWindow = ConfirmWindow(None)
        self.editWindow = EditWindow(self.conn, self.cursor,
(self.update, self.updateRestrictions))
        self.addExpenseWindow.setStyleSheet(style)
        self.addIncomeWindow.setStyleSheet(style)
        self.confirmWindow.setStyleSheet(style)
        self.editWindow.setStyleSheet(style)

        # msg box
        self.msgBox = QtWidgets.QMessageBox()
        self.msgBox.setIcon(QtWidgets.QMessageBox.Icon.Warning)
        self.msgBox.setWindowTitle('Message')

self.msgBox.setStandardButtons(QtWidgets.QMessageBox.StandardButto
n.Ok)
        self.msgBox.setStyleSheet(style)
        self.init_UI()

    def init_UI(self):
        self.setWindowTitle('')
        self.ui.tabWidget.currentChanged.connect(self.update)

self.ui.tableView.setSelectionMode(QtWidgets.QTableWidget.Selectio
nMode.MultiSelection)

        # date widgets setup
        self.ui.dateEdit.setDate(datetime.now())
        self.ui.dateEdit.dateChanged.connect(self.onDEchange)
        self.ui.dateEdit_2.setDate(datetime.now())
        self.ui.dateEdit_2.dateChanged.connect(self.onDEchange)
        self.ui.dateEdit_2.setVisible(False)
        self.ui.dateEdit_3.setDate(datetime.now())
```

Продовження додатку А

```
        self.ui.dateEdit_3.dateChanged.connect(self.onDEchange)
        self.ui.dateEdit_4.setDate(datetime.now())
        self.ui.dateEdit_4.dateChanged.connect(self.onDEchange)
        self.ui.dateEdit_4.setVisible(False)
        #
        self.ui.textEdit.setVisible(False)

self.ui.addExpenseButton.clicked.connect(self.openAddExpenseWindow
)

self.ui.addIncomeButton.clicked.connect(self.openAddIncomeWindow)

self.ui.addRestrictionButton.clicked.connect(self.openRestrictionC
onfirm)
        self.ui.toolButton.clicked.connect(self.msgTglState)

self.ui.updateGraphButton.clicked.connect(self.graphTabUpdate)
        self.ui.toolButton_2.clicked.connect(self.delTableRows)
        self.ui.toolButton_3.clicked.connect(self.delTableRows)
        self.ui.toolButton_4.clicked.connect(self.delTableRows)
        self.ui.toolButton_2.setIcon(QIcon('images/del.png'))
        self.ui.toolButton_3.setIcon(QIcon('images/del.png'))
        self.ui.toolButton_4.setIcon(QIcon('images/del.png'))
        self.ui.toolButton_5.clicked.connect(self.editCurrTable)
        self.ui.toolButton_6.clicked.connect(self.editCurrTable)
        self.ui.toolButton_7.clicked.connect(self.editCurrTable)
        self.ui.toolButton_5.setIcon(QIcon('images/edit.png'))
        self.ui.toolButton_6.setIcon(QIcon('images/edit.png'))
        self.ui.toolButton_7.setIcon(QIcon('images/edit.png'))
        # combo box
        self.ui.tablesComboBox.addItems(['Витрати', 'Прибуток'])

self.ui.tablesComboBox.activated.connect(self.changeCategory)

        request = 'SELECT СписокПеріодів.Назва FROM
СписокПеріодів;'
        period_arr = [item[0] for item in
self.cursor.execute(request).fetchall()]
        self.ui.periodComboBox.addItems(period_arr)
        self.ui.periodComboBox.activated.connect(self.onCBchange)
        self.ui.periodComboBox_2.addItems(period_arr)

self.ui.periodComboBox_2.activated.connect(self.onCBchange)
        self.ui.periodComboBox_3.addItems(period_arr[:-1])
        self.ui.periodComboBox_4.addItems(period_arr[:-1])

        request = f'SELECT СписокКатегорійВитрат.Назва FROM
СписокКатегорійВитрат ;'
        self.ui.categoryComboBox.addItems([item[0] for item in
self.cursor.execute(request).fetchall()][1:])
        self.ui.categoryComboBox_2.addItems([item[0] for item in
```

Продовження додатку А

```
self.cursor.execute(request).fetchall()])

self.ui.categoryComboBox_2.setCurrentText(self.ui.categoryComboBox
_2.itemText(1))

        request = 'SELECT Рахунок.НазваРахунку FROM Рахунок;'
        self.ui.accountComboBox.addItems([item[0] for item in
self.cursor.execute(request).fetchall()] + ['Всі'])
        self.ui.accountComboBox.activated.connect(self.update)
self.ui.accountComboBox_2.addItems([item[0] for item in
self.cursor.execute(request).fetchall()] + ['Всі'])

        request = f"SELECT Рахунок.Сума FROM Рахунок WHERE
Рахунок.НазваРахунку='{self.ui.accountComboBox.currentText()}';"

self.ui.currAccValLabel.setText(str(round(self.cursor.execute(requ
est).fetchone()[0], 2)))
        # fill default data

        request = f"SELECT Витрати.Категорія, Витрати.Сума,
Витрати.НазваРахунку FROM Витрати WHERE Витрати.Дата =
#{getDate()}# and Витрати.НазваРахунку =
'{self.ui.accountComboBox.currentText()}' ORDER BY Витрати.Дата
DESC;"
        self.setTable(self.ui.tableView, request)
        self.ui.periodLabel.setText(getDate())
        #
        self.initGraph()
        self.graphTabUpdate()
        self.updateRestrictions()

    def initGraph(self):
        self.ui.frame_2.setVisible(False)
        content_widget = QtWidgets.QWidget()
        layout = QtWidgets.QVBoxLayout(content_widget)
        frame = QtWidgets.QFrame()
        frame.setFixedSize(100,100)
        self.matplotlibWidget = MatplotlibWidget()
        self.matplotlibWidget.canvas.setMinimumSize(380, 220)
        self.matplotlibWidget.canvas.setMaximumSize(14000, 220)
        self.matplotlibWidget.canvas.setContentsMargins(0, 0, 0,
0)

self.matplotlibWidget.canvas.setSizePolicy(QtWidgets.QSizePolicy.M
inimumExpanding, QtWidgets.QSizePolicy.MinimumExpanding)


        layout.addWidget(frame)
        layout.addWidget(self.matplotlibWidget.canvas)
```

```
        self.scroll_area = QtWidgets.QScrollArea()
        self.scroll_area.setGeometry(10,90,380,220)

self.scroll_area.setVerticalScrollBarPolicy(QtCore.Qt.ScrollBarPol
icy.ScrollBarAlwaysOff)

self.scroll_area.setHorizontalScrollBarPolicy(QtCore.Qt.ScrollBarP
olicy.ScrollBarAlwaysOn)
        self.scroll_area.setWidgetResizable(True)
        self.scroll_area.setWidget(content_widget)

        self.ui.tabWidget.addTab(self.scroll_area, 'Графіки')

    def editCurrTable(self):
        rows = sorted(set(index.row() for index in
self.ui.tableView.selectedIndexes()))
        cols_num = self.ui.tableView.model().columnCount(0)
        if self.ui.tabWidget.currentIndex() == 0:
            tableName = 'Витрати'
        elif self.ui.tabWidget.currentIndex() == 1:
            tableName = 'Прибуток'
        elif self.ui.tabWidget.currentIndex() == 2:
            tableName = 'СписокОбмежень'
            cols_num = 4
        if rows:
            for i in rows:
                arr = [str(self.ui.tableView.model().index(i,
j).data()) for j in
                       range(cols_num)]
                if '' in set(arr):
                    self.showMsg('Таблиця порожня.')
                    return None
                # elements got
                if cols_num == 3:
                    arr[1] = float(arr[1])
                    attr = arr
                elif cols_num == 4:
                    arr[3] = float(arr[3])
                    attr = (arr[2], arr[3], arr[1], arr[0])
                self.editWindow.reset(attr, tableName)
                self.editWindow.show()
                break
        else:
            self.showMsg('Рядок не вибрано.')
            pass

    def onCBchange(self):
        tab = self.ui.tabWidget.currentIndex()
        accName = self.ui.accountComboBox.currentText()
        if tab == 0:
            self.ui.dateEdit.setDate(datetime.now())
```

Продовження додатку А

```
            self.ui.dateEdit_2.setDate(datetime.now())
            date = self.ui.dateEdit.dateTime().toPyDateTime()
            currCB = self.ui.periodComboBox

            currLabel = self.ui.periodLabel
            dateEditBot = self.ui.dateEdit_2
            tableName = 'Витрати'
            pass
        elif tab == 1:
            self.ui.dateEdit_3.setDate(datetime.now())
            self.ui.dateEdit_4.setDate(datetime.now())
            date = self.ui.dateEdit_3.dateTime().toPyDateTime()
            currCB = self.ui.periodComboBox_2
            currLabel = self.ui.periodLabel_2
            dateEditBot = self.ui.dateEdit_4
            tableName = 'Прибуток'
            pass
        else:
            return None
        currTable = self.ui.tableView
        if accName == 'Всі':
            accStr = ''
            self.ui.currAccValLabel.setText('')
        else:
            accStr = f"and {tableName}.НазваРахунку = '{accName}'"
            request = f"SELECT Рахунок.Сума FROM Рахунок WHERE
Рахунок.НазваРахунку='{accName}';"
            print('update accval')

self.ui.currAccValLabel.setText(str(round(self.cursor.execute(requ
est).fetchone()[0], 2)))
        if currCB.currentText() == 'День':
            if dateEditBot.isEnabled():
                dateEditBot.setEnabled(False)
                dateEditBot.setVisible(False)
            request = f"SELECT {tableName}.Категорія,
{tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE
{tableName}.Дата = #{date.strftime('%m-%d-%Y')}# {accStr} ORDER BY
{tableName}.Дата DESC;"
            print('table update')
            self.setTable(currTable, request)
            currLabel.setText(getDate())
        elif currCB.currentText() == 'Тиждень':
            if dateEditBot.isEnabled():
                dateEditBot.setEnabled(False)
                dateEditBot.setVisible(False)
            firstweekday = date -
timedelta(days=datetime.weekday(date))
            lastweekday = firstweekday + timedelta(days=6)
            request = f"SELECT {tableName}.Категорія,
{tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE
```

Продовження додатку А

```
{tableName}.Дата >= #{firstweekday.strftime('%m-%d-%Y')}# and
{tableName}.Дата <= #{lastweekday.strftime('%m-%d-%Y')}# {accStr}
ORDER BY {tableName}.Дата DESC;"
            print('table update')
            self.setTable(currTable, request)
            currLabel.setText(f'{firstweekday.strftime("%d-%m-
%Y")} - {lastweekday.strftime("%d-%m-%Y")}')
        elif currCB.currentText() == 'Місяць':
            dictMonth = {1: 'Січень', 2: 'Лютий', 3: 'Березень',
4: 'Квітень', 5: 'Травень', 6: 'Червень', 7: 'Липень',
                         8: 'Серпень', 9: 'Вересень', 10:
'Жовтень', 11: 'Листопад', 12: 'Грудень', }
            if dateEditBot.isEnabled():
                dateEditBot.setEnabled(False)
                dateEditBot.setVisible(False)
            firstmonthday = date - timedelta(days=date.day - 1)
            next_month = date.replace(day=28) + timedelta(days=4)
            lastmonthday = next_month -
timedelta(days=next_month.day)
            request = f"SELECT {tableName}.Категорія,
{tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE
{tableName}.Дата >= #{firstmonthday.strftime('%m-%d-%Y')}# and
{tableName}.Дата <= #{lastmonthday.strftime('%m-%d-%Y')}# {accStr}
ORDER BY {tableName}.Дата DESC;"
            print('table update')
            self.setTable(currTable, request)
            currLabel.setText(f"{dictMonth[date.month]}
{date.year}")
        elif currCB.currentText() == 'Рік':
            if dateEditBot.isEnabled():
                dateEditBot.setEnabled(False)
                dateEditBot.setVisible(False)
            request = f"SELECT {tableName}.Категорія,
{tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE
YEAR({tableName}.Дата) = YEAR(#{getDate()}#) {accStr} ORDER BY
{tableName}.Дата DESC;"
            print('table update')
            self.setTable(currTable, request)
            currLabel.setText(date.strftime("%Y рік"))
        elif currCB.currentText() == 'Період':
            currLabel.setText('Період')
            if not dateEditBot.isEnabled():
                dateEditBot.setEnabled(True)
                dateEditBot.setVisible(True)

    def onDEchange(self):
        tab = self.ui.tabWidget.currentIndex()
        accName = self.ui.accountComboBox.currentText()
        if tab == 0:
            date = self.ui.dateEdit.dateTime().toPyDateTime()
            currCB = self.ui.periodComboBox
```

Продовження додатку А

```
            currLabel = self.ui.periodLabel
            dateEditTop = self.ui.dateEdit
            dateEditBot = self.ui.dateEdit_2
            tableName = 'Витрати'
            pass
        elif tab == 1:
            date = self.ui.dateEdit_3.dateTime().toPyDateTime()
            currCB = self.ui.periodComboBox_2
            currLabel = self.ui.periodLabel_2
            dateEditTop = self.ui.dateEdit_3
            dateEditBot = self.ui.dateEdit_4
            tableName = 'Прибуток'
            pass
        else:
            return None
        currTable = self.ui.tableView
        if accName == 'Всі':
            accStr = ''
            self.ui.currAccValLabel.setText('')
        else:
            accStr = f"and {tableName}.НазваРахунку = '{accName}'"
            request = f"SELECT Рахунок.Сума FROM Рахунок WHERE
Рахунок.НазваРахунку='{accName}';"
            print(self.cursor.execute(request).fetchone())

self.ui.currAccValLabel.setText(str(round(self.cursor.execute(requ
est).fetchone()[0], 2)))
        if currCB.currentText() == 'День':
            request = f'SELECT {tableName}.Категорія,
{tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE
{tableName}.Дата = #{date.strftime("%m-%d-%Y")}# {accStr} ORDER BY
{tableName}.Дата DESC;'
            print('table update')
            self.setTable(currTable, request)
            currLabel.setText(date.strftime("%d-%m-%Y"))
            pass
        elif currCB.currentText() == 'Тиждень':
            firstweekday = date -
timedelta(days=datetime.weekday(date))
            lastweekday = firstweekday + timedelta(days=6)
            request = f'SELECT {tableName}.Категорія,
{tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE
{tableName}.Дата >= #{firstweekday.strftime("%m-%d-%Y")}# and
{tableName}.Дата <= #{lastweekday.strftime("%m-%d-%Y")}# {accStr}
ORDER BY {tableName}.Дата DESC;'
            print('table update')
            self.setTable(currTable, request)
            currLabel.setText(f'{firstweekday.strftime("%d-%m-
%Y")} - {lastweekday.strftime("%d-%m-%Y")}')
            pass
        elif currCB.currentText() == 'Місяць':
```

Продовження додатку А

```
            dictMonth = {1: 'Січень', 2: 'Лютий', 3: 'Березень',
4: 'Квітень', 5: 'Травень', 6: 'Червень', 7: 'Липень',
                         8: 'Серпень', 9: 'Вересень', 10:
'Жовтень', 11: 'Листопад', 12: 'Грудень', }
            firstmonthday = date - timedelta(days=date.day - 1)
            next_month = date.replace(day=28) + timedelta(days=4)
            lastmonthday = next_month -
timedelta(days=next_month.day)
            request = f'SELECT {tableName}.Категорія,
{tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE
{tableName}.Дата >= #{firstmonthday.strftime("%m-%d-%Y")}# and
{tableName}.Дата <= #{lastmonthday.strftime("%m-%d-%Y")}# {accStr}
ORDER BY {tableName}.Дата DESC;'
            print('table update')
            self.setTable(currTable, request)
            currLabel.setText(f"{dictMonth[date.month]}
{date.year}")
            pass
        elif currCB.currentText() == 'Рік':
            request = f'SELECT {tableName}.Категорія,
{tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE
YEAR({tableName}.Дата) = YEAR(#{date}#) {accStr} ORDER BY
{tableName}.Дата DESC;'
            print('table update')
            self.setTable(currTable, request)
            currLabel.setText(date.strftime("%Y рік"))
            pass
        elif currCB.currentText() == 'Період':
            date_1 = dateEditTop.dateTime().toPyDateTime()
            date_2 = dateEditBot.dateTime().toPyDateTime()
            print('1', date_1)
            print('2', date_2)
            if date_1 > date_2:
                dateEditTop.setDate(dateEditBot.date())
                date_1 = date_2

            request = f'SELECT {tableName}.Категорія,
{tableName}.Сума, {tableName}.НазваРахунку FROM {tableName} WHERE
{tableName}.Дата >= #{date_1.strftime("%m-%d-%Y")}# and
{tableName}.Дата <= #{date_2.strftime("%m-%d-%Y")}# {accStr} ORDER
BY {tableName}.Дата DESC;'
            print('table update')
            self.setTable(currTable, request)
            currLabel.setText(f'{date_1.strftime("%d-%m-%Y")} -
{date_2.strftime("%d-%m-%Y")}')
            pass

def restrictionTabUpdate(self):
    accName = self.ui.accountComboBox.currentText()
    if accName == 'Всі':
```

Продовження додатку А

```
            self.ui.currAccValLabel.setText('')
        else:
            request = f"SELECT Рахунок.Сума FROM Рахунок WHERE
Рахунок.НазваРахунку='{accName}';"
            print(self.cursor.execute(request).fetchone())

self.ui.currAccValLabel.setText(str(round(self.cursor.execute(requ
est).fetchone()[0], 2)))
        request = f"SELECT СписокОбмежень.НазваОбмеження,
СписокОбмежень.Період, СписокОбмежень.Категорія,
СписокОбмежень.Сума FROM СписокОбмежень;"
        self.setTable(self.ui.tableView, request)

self.ui.periodComboBox_3.setCurrentText(self.ui.periodComboBox_3.i
temText(0))

self.ui.categoryComboBox.setCurrentText(self.ui.categoryComboBox.i
temText(0))
        self.ui.doubleSpinBox.setValue(0.0)
        self.ui.lineEdit.setText('')

    def graphTabUpdate(self):
        def dateDay(date, tableName):
            return f"Витрати.Дата = #{date.strftime('%m-%d-%Y')}#"

        def incDay(date):
            return date + timedelta(days=1)

        def dateWeek(date, tableName):
            firstweekday = date -
timedelta(days=datetime.weekday(date))
            lastweekday = firstweekday + timedelta(days=6)
            return f"{tableName}.Дата >= #{firstweekday.strftime('%m-
%d-%Y')}# and {tableName}.Дата <= #{lastweekday.strftime('%m-%d-
%Y')}#"

        def incWeek(date):
            return date + timedelta(days=7)

        def dateMonth(date, tableName):
            firstmonthday = date - timedelta(days=date.day - 1)
            next_month = date.replace(day=28) + timedelta(days=4)
            lastmonthday = next_month - timedelta(days=next_month.day)
            return f"{tableName}.Дата >= #{firstmonthday.strftime('%m-
%d-%Y')}# and {tableName}.Дата <= #{lastmonthday.strftime('%m-%d-
%Y')}#"

        def incMonth(date):
            date = date + timedelta(days=31)
            return datetime(day=1, month=date.month, year=date.year)
```

Продовження додатку А

```
    def dateYear(date, tableName):
        return f"YEAR({tableName}.Дата) =
Year(#{date.strftime('%m-%d-%Y')}#)"

    def incYear(date):
        date = date + timedelta(days=366)
        return datetime(day=1, month=1, year=date.year)

    dateFormat = "%d-%m"
    # get date range
    request = f"SELECT MIN(Дата) AS min_date FROM Витрати;"
    min_date = self.cursor.execute(request).fetchone()[0]
    request = f"SELECT MAX(Дата) AS min_date FROM Витрати;"
    max_date = self.cursor.execute(request).fetchone()[0]

    currCB = self.ui.periodComboBox_4
    tableName = self.ui.tablesComboBox.currentText()
    accName = self.ui.accountComboBox_2.currentText()
    category = self.ui.categoryComboBox_2.currentText()

    if accName == 'Всі':
        accStr = ''
        self.ui.currAccValLabel.setText('')
    else:
        accStr = f"and {tableName}.НазваРахунку = '{accName}'"

    x = []
    y = []
    if currCB.currentText() == 'День':
        date = min_date
        dateStr = dateDay
        dateInc = incDay
    elif currCB.currentText() == 'Тиждень':
        date = min_date -
timedelta(days=datetime.weekday(min_date))
        dateStr = dateWeek
        dateInc = incWeek
        pass
    elif currCB.currentText() == 'Місяць':
        date =
datetime(day=1,month=min_date.month,year=min_date.year)
        dateStr = dateMonth
        dateInc = incMonth
        pass
    elif currCB.currentText() == 'Рік':
        date = datetime(day=1,month=1,year=min_date.year)
        dateStr = dateYear
        dateInc = incYear
        dateFormat = "%Y"
        pass
```

Продовження додатку А

```
    while date <= max_date:
        request = f"SELECT Sum({tableName}.Сума) AS [Sum-Сума]
FROM {tableName} WHERE {dateStr(date, tableName)} AND
{tableName}.Категорія='{category}'{accStr} GROUP BY
{tableName}.Категорія;"
        result = self.cursor.execute(request).fetchone()
        if result is None:
            if not self.ui.checkBox.isChecked():
                y.append(0)
                x.append(date.strftime(dateFormat))
        else:
            y.append(result[0])
            x.append(date.strftime(dateFormat))
        date = dateInc(date)
    # check restrictions
    if self.ui.checkBox_2.isChecked():
        request = f"SELECT СписокОбмежень.Сума FROM СписокОбмежень
WHERE
СписокОбмежень.Період='{self.ui.periodComboBox_4.currentText()}'
AND
СписокОбмежень.Категорія='{self.ui.categoryComboBox_2.currentText(
)}';"
        summ = self.cursor.execute(request).fetchone()
        if summ is not None:
            summ = summ[0]
    else:
        summ = None

    periodLabels = {'День': ['Щоденні','Щоденний'], 'Тиждень':
['Щотижневі','Щотижневий'], 'Місяць': ['Щомісячні','Щомісячний'],
'Рік': ['Щорічні','Щорічний']}
    tableDict = {'Витрати': 0, 'Прибуток': 1}
    title =
f"{periodLabels[currCB.currentText()][tableDict[tableName]]}
{tableName.lower()} в категорії {category}"

    print(title)
    # draw
    self.ui.label_3.setText(title)
    self.matplotlibWidget.plot(x, y, summ)
    pass

def update(self):
    # table update
    if self.ui.tabWidget.currentIndex() in [0, 1]:
        self.setFrameState(False)
        self.setTableState(True)
        self.onDEchange()
    elif self.ui.tabWidget.currentIndex() == 2:
        self.setFrameState(False)
```

Продовження додатку А

```
            self.setTableState(True)
            self.restrictionTabUpdate()
        elif self.ui.tabWidget.currentIndex() == 3:
            self.setFrameState(True)
            self.setTableState(False)
    def updateRestrictions(self):
        request = f"SELECT СписокОбмежень.НазваОбмеження,
    СписокОбмежень.Період, СписокОбмежень.Категорія,
    СписокОбмежень.Сума FROM СписокОбмежень;"
        restrictionList = self.cursor.execute(request).fetchall()
        colors = {-1: '#000000', 0: '#000000', 1: '#f0f009', 2:
    '#ffc400', 3: '#ff0000'}
        htmlStr = ""
        colorId = []
        for item in restrictionList:
            answer = self.checkRestriction(*item,)
            if answer[0] in [1,2,3]:
                htmlStr += f"<div style='color:{colors[answer[0]]};
    margin-top: 3px; margin-bottom: 3px;'>-  {answer[1]}</div><br>"
            colorId.append(answer[0])
            pass
        maxId = max(colorId)
        if maxId < 1:
            self.ui.toolButton.setText('...')
        else:
            self.ui.toolButton.setText(' ! ')
        self.ui.toolButton.setStyleSheet(f"color: {colors[maxId]};")
        self.ui.textEdit.setHtml(htmlStr)
    def checkRestriction(self, name, period, category, summ,
    date=None, accName=None):
        if date is None:
            date = datetime.now()
        if period == 'День':
            dateStr = f"Витрати.Дата = #{date.strftime('%m-%d-%Y')}#"
        elif period == 'Тиждень':
            firstweekday = date -
    timedelta(days=datetime.weekday(date))
            lastweekday = firstweekday + timedelta(days=6)
            dateStr = f"Витрати.Дата >= #{firstweekday.strftime('%m-
    %d-%Y')}# and Витрати.Дата <= #{lastweekday.strftime('%m-%d-
    %Y')}#"
            pass
        elif period == 'Місяць':
            firstmonthday = date - timedelta(days=date.day - 1)
            next_month = date.replace(day=28) + timedelta(days=4)
            lastmonthday = next_month - timedelta(days=next_month.day)
            dateStr = f"Витрати.Дата >= #{firstmonthday.strftime('%m-
    %d-%Y')}# and Витрати.Дата <= #{lastmonthday.strftime('%m-%d-
    %Y')}#"
```

```
        pass
    elif period == 'Рік':
        dateStr = f"YEAR(Витрати.Дата) = Year(#{date.strftime('%m-
%d-%Y')}#)"
        pass

    if accName is None:
        accStr = ''
    else:
        accStr = f" and Витрати.НазваРахунку = '{accName}'"

    request = f"SELECT Sum(Витрати.Сума) AS [Sum-Сума] FROM
Витрати WHERE {dateStr} AND Витрати.Категорія='{category}'{accStr}
GROUP BY Витрати.Категорія;"
    periodSum = self.cursor.execute(request).fetchone()
    if periodSum is not None: periodSum = round(periodSum[0], 2)
    print(name, periodSum)
    print(summ)
    periodLabels = {'День': 'Щоденні', 'Тиждень': 'Щотижневі',
'Місяць': 'Щомісячні', 'Рік': 'Щорічні'}
    if periodSum is None:
        return -1,''
    elif periodSum <= summ*0.75:
        return 0,''
    elif periodSum > summ*0.75 and periodSum < summ*0.9:
        return 1, f"{periodLabels[period]} витрати в категорії
{category} наближаються до порога (Поріг: {summ}, Витрати:
{periodSum})"
    elif periodSum >= summ*0.9 and periodSum < summ:
        return 2, f"{periodLabels[period]} витрати в категорії
{category} критично наближаються до порога (Поріг: {summ},
Витрати: {periodSum})"
    else:
        return 3, f"{periodLabels[period]} витрати в категорії
{category} перевищують поріг (Поріг: {summ}, Витрати:
{periodSum})"

def openRestrictionConfirm(self):
    name = self.ui.lineEdit.text().replace("'","`")
    if name != '':
        request = f"SELECT СписокОбмежень.НазваОбмеження FROM
СписокОбмежень WHERE СписокОбмежень.НазваОбмеження='{name}';"
        if self.cursor.execute(request).fetchone() is None:
            if self.ui.doubleSpinBox.value() != 0.0:
                request = f"INSERT INTO СписокОбмежень (
НазваОбмеження, Період, Категорія, Сума ) " \
                          f"VALUES ('{name}',
'{self.ui.periodComboBox_3.currentText()}',
'{self.ui.categoryComboBox.currentText()}',
{self.ui.doubleSpinBox.value()} );"
```

Продовження додатку А

```
                label = f"Ви бажаєте додати у " \
                        f"Список Обмежень<br>" \
                        f"НазваОбмеження: {name}<br>" \
                        f"Період:
{self.ui.periodComboBox_3.currentText()}<br>" \
                        f"Категорія:
{self.ui.categoryComboBox.currentText()}<br>" \
                        f"Сума: {self.ui.doubleSpinBox.value()}"
                print('table update')
                self.confirmWindow.reset(self.simpleRequest,
request, label, onclose=(self.update, self.updateRestrictions,),
parentClose=False)
                self.confirmWindow.show()
            else:
                self.showMsg('Число не вибрано.')
        else:
            self.showMsg('Така назва вже існує.')
    else:
        self.showMsg('Назву не вибрано.')

def openAddExpenseWindow(self):
    self.addExpenseWindow.reset()
    self.addExpenseWindow.show()
    pass

def openAddIncomeWindow(self):
    self.addIncomeWindow.reset()
    self.addIncomeWindow.show()
    pass

def setTable(self, table=None, request='', empty=False):
    if self.ui.tabWidget.currentIndex() == 2:
        headerLabels = ['НазваОбмеження', 'Період', 'Категорія',
'Сума']
        emptyData = [('', '', '', '')]
    else:
        headerLabels = ['Категорія', 'Сума', 'НазваРахунку']
        emptyData = [('', '', '')]
    if empty is True or table is None:
        data = emptyData
    else:
        data = self.cursor.execute(request).fetchall()
        if not data:
            data = emptyData
    model = TableModel(data)
    model.setHorizontalHeaderLabels(headerLabels)

    table.setModel(model)
    table.horizontalHeader().setStretchLastSection(True)
```

Продовження додатку А

```
def delTableRows(self):
    def sendReq(requests):
        for item in requests:
            print(item)
            self.cursor.execute(item)
            self.conn.commit()
    rows = sorted(set(index.row() for index in
self.ui.tableView.selectedIndexes()))
    cols_num = self.ui.tableView.model().columnCount(0)
    if self.ui.tabWidget.currentIndex() == 0:
        tableName = 'Витрати'
    elif self.ui.tabWidget.currentIndex() == 1:
        tableName = 'Прибуток'
    elif self.ui.tabWidget.currentIndex() == 2:
        tableName = 'СписокОбмежень'
        cols_num = 4
    requestsToDel = []
    label = """<p align="center">Ви дійсно хочете видалити ці
об'єкти:</p>"""
    if rows:
        for i in rows:
            arr = [str(self.ui.tableView.model().index(i,
j).data()) for j in range(cols_num)]
            if '' in set(arr):
                self.showMsg('Таблиця порожня.')
                return None
            label = label + ' '.join(arr) + '<br>'
            if len(arr) == 3:
                requestsToDel.append(f"DELETE FROM {tableName}
WHERE {tableName}.Категорія = '{arr[0]}' and {tableName}.Сума =
{arr[1]} and {tableName}.НазваРахунку = '{arr[2]}';")
            else:
                requestsToDel.append(f"DELETE FROM {tableName}
WHERE {tableName}.НазваОбмеження = '{arr[0]}' and
{tableName}.Період = '{arr[1]}' and {tableName}.Категорія =
'{arr[2]}' and {tableName}.Сума = {arr[3]};")

        self.confirmWindow.reset(sendReq, requestsToDel, label,
onclose=(self.update, self.updateRestrictions,),
parentClose=False)
        self.confirmWindow.show()
    else:
        self.showMsg('Не виділено жодного рядка.')

def changeCategory(self):
    tableName = self.ui.tablesComboBox.currentText()
    tableDict = {'Витрати': 'Витрат', 'Прибуток': 'Прибутку'}
    request = f'SELECT СписокКатегорій{tableDict[tableName]}.Назва
FROM СписокКатегорій{tableDict[tableName]};'
```

```
        self.ui.categoryComboBox_2.clear()
        self.ui.categoryComboBox_2.addItems([item[0] for item in
self.cursor.execute(request).fetchall()])

self.ui.categoryComboBox_2.setCurrentText(self.ui.categoryComboBox
_2.itemText(1))

def setTableState(self, toThisState):
    if self.ui.tableView.isVisible() is not toThisState:
        self.ui.tableView.setVisible(toThisState)
        self.ui.tableView.setEnabled(toThisState)

def setFrameState(self, toThisState):
    if self.ui.frame_2.isVisible() is not toThisState:
        self.ui.frame_2.setVisible(toThisState)
        self.ui.frame_2.setEnabled(toThisState)

def conn_db(self):
    self.conn = pyodbc.connect(r'Driver={Microsoft Access Driver
(*.mdb, *.accdb)};' +

rf'DBQ={os.path.dirname(os.path.abspath(__file__))}\bd.accdb;')
    return self.conn.cursor()

def showMsg(self, label=''):
    self.msgBox.setText(label)
    self.msgBox.show()

def msgTglState(self):
    if self.ui.textEdit.isVisible():
        self.ui.textEdit.setVisible(False)
    else:
        self.ui.textEdit.setVisible(True)

def simpleRequest(self, request):
    self.cursor.execute(request)
    self.conn.commit()

def closeEvent(self, a0: QtGui.QCloseEvent):
    self.addExpenseWindow.close()
    self.addExpenseWindow.confirmWindow.close()
    self.addIncomeWindow.close()
    self.addIncomeWindow.confirmWindow.close()
    self.conn.close()
```

Додаток Б

Код для класу вікна нового запису

```
class RecordWindow(QtWidgets.QDialog):
    def __init__(self, db, dbCurs, mainUpdate, tab):
        super(RecordWindow, self).__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.db = db
        self.dbCursor = dbCurs
        self.mainUpdate = mainUpdate
        self.tab = tab
        # additional windows
        self.confirmWindow = ConfirmWindow(self.close)
        self.msgBox = QtWidgets.QMessageBox()
        self.msgBox.setIcon(QtWidgets.QMessageBox.Icon.Warning)
        self.msgBox.setWindowTitle('Message')

self.msgBox.setStandardButtons(QtWidgets.QMessageBox.StandardButto
n.Ok)
        self.confirmWindow.setStyleSheet(style)
        self.msgBox.setStyleSheet(style)
        self.initUI()

    def initUI(self):
        self.setWindowTitle('')
        self.ui.label.setText('Категорія')
        self.ui.label_2.setText('Дата')
        self.ui.label_3.setText('Рахунок')
        self.ui.label_4.setText('Сума')
        self.ui.label_5.setText('Другий рахунок')
        # wigets
        request = f"SELECT
СписокКатегорій{self.getTableName(self.tab,1)}.Назва FROM
СписокКатегорій{self.getTableName(self.tab,1)};"
        self.ui.categoryComboBox.addItems([item[0] for item in
self.dbCursor.execute(request).fetchall()])

self.ui.categoryComboBox.activated.connect(self.onCBchange)
        request = 'SELECT Рахунок.НазваРахунку FROM Рахунок;'
        self.ui.accountComboBox.addItems([item[0] for item in
self.dbCursor.execute(request).fetchall()])
        self.ui.accountComboBox_2.addItems([item[0] for item in
self.dbCursor.execute(request).fetchall()])
        self.ui.lineEdit.setPlaceholderText('Нова Категорія')
        self.ui.lineEdit.setVisible(False)
        # buttons
        self.ui.pushButton.clicked.connect(self.showLineEdit)
        self.ui.pushButton.setIcon(QIcon('images/add.png'))
        self.ui.pushButton_2.clicked.connect(self.addNewCategory)
```

Продовження додатку Б

```
        self.ui.pushButton_2.setVisible(False)
        self.ui.pushButton_2.setIcon(QIcon('images/check.png'))
        self.ui.buttonBox.accepted.connect(self.confirmWin)
        self.ui.buttonBox.rejected.connect(self.close)

    def showLineEdit(self):
        if self.ui.lineEdit.isVisible():
            self.ui.lineEdit.setVisible(False)
            self.ui.pushButton_2.setVisible(False)
            pass
        else:
            self.ui.lineEdit.setVisible(True)
            self.ui.pushButton_2.setVisible(True)
        pass

    def addNewCategory(self):
        name = self.ui.lineEdit.text().replace("'","`")
        if name != '':
            tableName =
f"СписокКатегорій{self.getTableName(self.tab,1)}"
            request = f"SELECT {tableName}.Назва FROM {tableName}
WHERE {tableName}.Назва='{name}';"
            if self.dbCursor.execute(request).fetchone() is None:
                request = f"INSERT INTO {tableName} (Назва) " \
                          f"VALUES ('{name}');"
                print(request)
                label = f"Ви бажаєте додати у <br>{tableName}<br>
<br>" \
                        f"Категорію: {name}"
                self.ui.categoryComboBox.addItem(name)
                self.ui.lineEdit.setText('')
                self.confirmWindow.reset(self.simpleRequest,
request, label, (*self.mainUpdate, self.reset), parentClose=False)
                self.confirmWindow.show()
                pass
            else:
                self.showMsg('Така категорія вже існує.')
        else:
            self.showMsg('Назву категорії не вказано.')

    def simpleRequest(self, request):
        self.dbCursor.execute(request)
        self.db.commit()

    def confirmWin(self):
        if self.ui.doubleSpinBox.value() != 0.0:
            request = f"SELECT Рахунок.Сума FROM Рахунок WHERE
Рахунок.НазваРахунку='{self.ui.accountComboBox.currentText()}';"
            acc_1_value =
round(self.dbCursor.execute(request).fetchone()[0], 2)
```

Продовження додатку Б

```
            print(self.ui.doubleSpinBox.value())
            print(acc_1_value)
            if self.ui.categoryComboBox.currentText() != 'Переказ
Коштів':
                tableName = self.getTableName(self.tab)
                if self.tab == 0:
                    compare = self.ui.doubleSpinBox.value() <
acc_1_value
                else:
                    compare = True
                if compare:
                    request = f"INSERT INTO {tableName} (
Категорія, Сума, Дата, НазваРахунку ) " \
                              f"VALUES
('{self.ui.categoryComboBox.currentText()}',
{self.ui.doubleSpinBox.value()},
#{self.ui.dateEdit.dateTime().toPyDateTime().strftime('%m-%d-
%Y')}#, '{self.ui.accountComboBox.currentText()}');"
                    print(request)
                    confirmlabel = f"Ви бажаєте додати у
{tableName.lower()} <br>" \
                                   f"Категорія:
'{self.ui.categoryComboBox.currentText()}'<br>Рахунок:
'{self.ui.accountComboBox.currentText()}'<br>" \
                                   f"Дата:
'{self.ui.dateEdit.dateTime().toPyDateTime().strftime('%m-%d-
%Y')}'<br>Сума '{self.ui.doubleSpinBox.value()}'"
                    self.confirmWindow.reset(self.sendRequest,
[request], confirmlabel, (*self.mainUpdate,))
            else:
                compare = self.ui.doubleSpinBox.value() <
acc_1_value
                if compare:
                    if self.ui.accountComboBox.currentText() !=
self.ui.accountComboBox_2.currentText():
                        request1 = f"INSERT INTO Витрати (
Категорія, Сума, Дата, НазваРахунку ) " \
                                   f"VALUES
('{self.ui.categoryComboBox.currentText()}',
{self.ui.doubleSpinBox.value()},
#{self.ui.dateEdit.dateTime().toPyDateTime().strftime('%m-%d-
%Y')}#, '{self.ui.accountComboBox.currentText()}');"
                        request2 = f"INSERT INTO Прибуток (
Категорія, Сума, Дата, НазваРахунку ) " \
                                   f"VALUES
('{self.ui.categoryComboBox.currentText()}',
{self.ui.doubleSpinBox.value()},
#{self.ui.dateEdit.dateTime().toPyDateTime().strftime('%m-%d-
%Y')}#, '{self.ui.accountComboBox_2.currentText()}');"
                        print(request1,'\n' ,request2)
```

Продовження додатку Б

```
confirmlabel = f"Ви бажаєте перевести<br>" \
                                          f"Суму:
{self.ui.doubleSpinBox.value()}<br>" \
                                          f"З рахунку:
{self.ui.accountComboBox.currentText()}<br>" \
                                          f"На рахунок:
{self.ui.accountComboBox_2.currentText()}<br>"
                        self.confirmWindow.reset(self.sendRequest,
[request1, request2], confirmlabel, (*self.mainUpdate,))
                    else:
                        return None
            self.confirmWindow.show()

        else:
            self.showMsg('Число не вибрано.')
        pass

    def onCBchange(self):
        if self.ui.categoryComboBox.currentText() == 'Переказ
Коштів':
            self.resize(QtCore.QSize(280,255))
            self.ui.accountComboBox_2.setVisible(True)
            self.ui.label_5.setVisible(True)
            self.ui.frame.setGeometry(QtCore.QRect(10, 120,
self.ui.frame.geometry().width(),
self.ui.frame.geometry().height()))
            self.ui.buttonBox.setGeometry(
                QtCore.QRect(40, 210,
self.ui.buttonBox.geometry().width(),
self.ui.buttonBox.geometry().height()))
        else:
            self.resize(QtCore.QSize(280,220))
            self.ui.accountComboBox_2.setVisible(False)
            self.ui.label_5.setVisible(False)
            self.ui.frame.setGeometry(QtCore.QRect(10, 70,
self.ui.frame.geometry().width(),
self.ui.frame.geometry().height()))
            self.ui.buttonBox.setGeometry(
                QtCore.QRect(40, 160,
self.ui.buttonBox.geometry().width(),
self.ui.buttonBox.geometry().height()))

    def sendRequest(self, request):
        symb_dict = {0: '-', 1: '+'}
        for i, item in enumerate(request):
            if i == 0:
                accName = self.ui.accountComboBox.currentText()
                pass
            elif i == 1:
```

Продовження додатку Б

```
                accName = self.ui.accountComboBox_2.currentText()
                    pass
                print("send", item)
                # update expense table
                self.dbCursor.execute(item)
                self.db.commit()
                # add value to account
                symbol = symb_dict[i]
                if self.tab == 1:
                    if symbol == '-':
                        symbol = '+'
                    else:
                        symbol = '-'
                request = f"UPDATE Рахунок SET Рахунок.Сума =
Рахунок.Сума{symbol}{self.ui.doubleSpinBox.value()} WHERE
Рахунок.НазваРахунку='{accName}';"
                print('change', request)
                self.dbCursor.execute(request)
                self.db.commit()
            for item in self.mainUpdate:
                item()
    def showMsg(self, label=''):
        self.msgBox.setText(label)
        self.msgBox.show()
    def getTableName(self,var1=0, var2=0):
        dict = {0: ['Витрати', 'Витрат'], 1: ['Прибуток',
'Прибутку']}
        return dict[var1][var2]
    def reset(self):
        # resetable values
        self.resize(QtCore.QSize(280,220))
        self.ui.dateEdit.setDate(datetime.now())
        self.ui.doubleSpinBox.setValue(0.0)

self.ui.categoryComboBox.setCurrentText(self.ui.categoryComboBox.i
temText(1))
self.ui.accountComboBox.setCurrentText(self.ui.accountComboBox.ite
mText(0))
self.ui.accountComboBox_2.setCurrentText(self.ui.accountComboBox.i
temText(1))
        self.ui.accountComboBox_2.setVisible(False)
        self.ui.label_5.setVisible(False)
self.ui.frame.setGeometry(QtCore.QRect(10,70,self.ui.frame.geometr
y().width(),self.ui.frame.geometry().height()))self.ui.buttonBox.s
etGeometry(QtCore.QRect(40,160,self.ui.buttonBox.geometry().width(
),self.ui.buttonBox.geometry().height()))

    def closeEvent(self, a0: QtGui.QCloseEvent):
        self.confirmWindow.close()
```

Додаток В

Код для класу вікна підтвердження

```
class ConfirmWindow(QtWidgets.QDialog):
    def __init__(self, parentDialogClose,):
        super(ConfirmWindow, self).__init__()
        self.ui = Ui_Confirm()
        self.ui.setupUi(self)
        #
        self.request = ''
        self.sendFunc = None
        self.parentDialogClose = parentDialogClose
        self.isParentClose = True
        self.onClose = None
        # buttons
        self.ui.buttonBox.rejected.connect(self.close)
        self.ui.buttonBox.accepted.connect(self.send)

    def reset(self, outerSendFunc, request, label, onclose=None,
parentClose=True):
        self.sendFunc = outerSendFunc
        self.request = request
        label = f'''<p align="center">{label}</p>'''
        self.ui.textEdit.setHtml(label)
        self.isParentClose = parentClose
        if onclose is not None:
            self.onClose = onclose

    def send(self):
        self.sendFunc(self.request)
        if self.onClose is not None:
            for func in self.onClose:
                func()
        if self.isParentClose:
            self.parentDialogClose()
```

Додаток Г

Код для класу вікна редагування

```
class EditWindow(QtWidgets.QDialog):
    def __init__(self, db, dbCurs, mainUpdate):
        super(EditWindow, self).__init__()
        self.ui = Ui_Dialog()
        self.ui.setupUi(self)
        self.db = db
        self.dbCursor = dbCurs
        self.mainUpdate = mainUpdate
        self.confirmWindow = ConfirmWindow(self.close)
        self.attr = None
        self.tableName = None
        self.ui.buttonBox.accepted.connect(self.confirmWin)
        self.ui.buttonBox.rejected.connect(self.close)
        self.init_UI()

    def init_UI(self):
        self.msgBox = QtWidgets.QMessageBox()
        self.msgBox.setIcon(QtWidgets.QMessageBox.Icon.Warning)
        self.msgBox.setWindowTitle('Message')

self.msgBox.setStandardButtons(QtWidgets.QMessageBox.StandardButto
n.Ok)
        self.confirmWindow.setStyleSheet(style)
        self.msgBox.setStyleSheet(style)
        pass

    def reset(self, attr, tableName):
        # reset old val
        self.tableName = tableName
        self.attr = attr
        self.ui.doubleSpinBox.setValue(attr[1])
        self.ui.label_5.setText(attr[0])

        self.ui.label_6.setText(str(attr[1]))
        self.ui.label_7.setText(attr[2])
        #
        tableDict = {'Витрати': 'Витрат', 'Прибуток': 'Прибутку',
'СписокОбмежень': 'Витрат'}
        print(tableName)
        request = f'SELECT
СписокКатегорій{tableDict[tableName]}.Назва FROM
СписокКатегорій{tableDict[tableName]};'
        self.ui.comboBox.clear()
        self.ui.comboBox.addItems([item[0] for item in
self.dbCursor.execute(request).fetchall()])
        self.ui.comboBox.setCurrentText(attr[0])
        if len(attr) == 4:
            self.ui.label_8.setText(attr[3])
```

Продовження додатку Г

```
            self.ui.lineEdit.setText(attr[3])
            self.ui.label_3.setText("Період")
            self.ui.label_4.setText("Назва Обмеження")
            # period
            request = 'SELECT СписокПеріодів.Назва FROM
СписокПеріодів;'
            period_arr = [item[0] for item in
self.dbCursor.execute(request).fetchall()]
            self.ui.comboBox_2.clear()
            self.ui.comboBox_2.addItems(period_arr)
            self.ui.comboBox_2.setCurrentText(attr[2])
            self.resize(451,141)
        else:
            self.ui.label_3.setText("Рахунок")
            # account
            request = 'SELECT Рахунок.НазваРахунку FROM Рахунок;'
            self.ui.comboBox_2.clear()
            self.ui.comboBox_2.addItems([item[0] for item in
self.dbCursor.execute(request).fetchall()])
            self.ui.comboBox_2.setCurrentText(attr[2])

            self.resize(321,141)

    def confirmWin(self):
        category = self.ui.comboBox.currentText()
        summ = self.ui.doubleSpinBox.value()
        if summ != 0.0:
            extra = self.ui.comboBox_2.currentText()
            onUpdate = []

            if category != self.attr[0]:
onUpdate.append(f"{self.tableName}.Категорія = '{category}'")
            if summ != self.attr[1]:
onUpdate.append(f"{self.tableName}.Сума = {summ}")

            if len(self.attr) == 4:
                if extra != self.attr[2]:
onUpdate.append(f"{self.tableName}.Період = '{extra}'")
                name = self.ui.lineEdit.text()
                if name == '':
                    self.showMsg('Порожнє значення в полі Назва є
неприпустимим.')
                    return None
                if name != self.attr[3]:
onUpdate.append(f"{self.tableName}.НазваОбмеження = '{name}'")
            else:
                if extra != self.attr[2]:
onUpdate.append(f"{self.tableName}.НазваРахунку = '{extra}'")
                pass
```

Продовження додатку Г

```
            if onUpdate:
                onUpdate = ", ".join(onUpdate)
                print("on update", onUpdate)
                if len(self.attr) == 3:
                    onCheck = f"{self.tableName}.Категорія =
'{self.attr[0]}' and {self.tableName}.Сума = {self.attr[1]}" \
                              f"and {self.tableName}.НазваРахунку
= '{self.attr[2]}'"
                    label = f"Ви дійсно бажаєте замінити<br>
{self.attr[0]} {str(self.attr[1])} {self.attr[2]} на <br>" \
                            f"{category} {str(summ)} {extra}"
                    pass
                elif len(self.attr) == 4:
                    onCheck = f"{self.tableName}.Категорія =
'{self.attr[0]}' and {self.tableName}.Сума = {self.attr[1]}" \
                              f" and {self.tableName}.Період =
'{self.attr[2]}' and {self.tableName}.НазваОбмеження =
'{self.attr[3]}'"
                    label = f"Ви дійсно бажаєте замінити<br>
{self.attr[0]} {str(self.attr[1])} {self.attr[2]} {self.attr[3]}
на <br>" \
                            f"{category} {str(summ)} {extra}
{name}"
                    pass
                request = f"UPDATE {self.tableName} SET {onUpdate}
WHERE {onCheck};"

                print(request)
                self.confirmWindow.reset(self.sendRequest,
request, label, (*self.mainUpdate,))
                self.confirmWindow.show()
                pass
            else:
                self.showMsg('Старі значення не були змінені.')
                print()
                # todo show msg
        else:
            self.showMsg('Неприпустиме значення 0.0 в полі
\'Сума\'.')

    def sendRequest(self, request):
        self.dbCursor.execute(request)
        self.db.commit()

    def showMsg(self, label=''):
        self.msgBox.setText(label)
        self.msgBox.show()
```

Додаток Д

Код для класу відмальовування діаграми

```
class MatplotlibWidget(QWidget):
    def __init__(self, parent=None):
        super().__init__(parent)
        self.figure = Figure(dpi=100)
        self.ax = self.figure.add_subplot(111)
        self.figure.set_facecolor('#323232')
        self.canvas = FigureCanvas(self.figure)
        #self.canvas.setMinimumSize()
        self.layout = QVBoxLayout()
        self.layout.addWidget(self.canvas)
        self.setLayout(self.layout)

    def plot(self, x=None, y=None, y_restr=None):
        self.ax.clear()
        # Generate some sample data
        if x is None:
            x = [i for i in range(1,16)]
        if y is None:
            y = x
        if len(x) < 6:
            width = 0.1
        else:
            width = 0.5

        self.figure.tight_layout()
        self.figure.subplots_adjust(left=0.05, bottom=0.22,
right=0.95, top=0.78)
        self.ax.bar(x, y, width=width)

        # Add values above the columns
        for i, v in enumerate(y):
            self.ax.text(x[i], v + 0.1, str(round(v, 2)),
ha='center')
        self.ax.set_xticklabels(x, rotation=45)
        self.ax.set_yticklabels([])
        # correct width
        if len(x) > 6:
            inches = corr_func(len(x))
        else:
            inches = 3.4
        self.figure.set_figwidth(inches)
        # set width with inches
        width = int(inches * 90 + 50)
        if width < 440: width = 440
        self.canvas.setMinimumSize(width, 220)
        if y_restr is not None:
```

Продовження додатку Д

```
            self.ax.axhline(y_restr, color='red')

            self.ax.annotate(f'{y_restr}    ', xy=(x[0], y_restr),
xytext=(x[0], y_restr), ha='right', va='bottom')

        print("figwidth", self.figure.get_figwidth())
        print(len(x))
        print(x)

        # Refresh the canvas
        self.canvas.draw()
```

ЗГОДА здобувачки вищої освіти

Державного університету економіки і технологій про
перевірку кваліфікаційної роботи на прояви
академічного плагіату
та розміщення в Репозитарії Університету

Я, **Сердюк Олександра Олегівна**,

підтримую політику Державного університету економіки і технологій з академічної доброчесності і відкритого доступу.

Засвідчую, що кваліфікаційна бакалаврська робота

«Розробка застосунку для класифікації та моніторингу персональних витрат»

виконана самостійно та не містить академічного плагіату. Я не надавала і не одержувала недозволену допомогу під час підготовки цієї роботи. Робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

Із чинним Положенням про запобігання та виявлення академічного плагіату в роботах здобувачів вищої освіти Державного університету економіки і технологій ознайомлена. Чітко усвідомлюю, що в разі виявлення у кваліфікаційній роботі порушення норм академічної доброчесності робота не допускається до захисту або оцінюється незадовільно.

Також я поінформована, що відповідно до «Положення про Репозитарій (електронну базу даних) Державного університету економіки і технологій» зазначена робота буде розміщена в Електронному архіві Університету (Репозитарії ДУЕТ). З умовами такого розміщення ознайомлена.

15.06.26 р  Сердюк О.О.