МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ФУРМАН Віталій Павлович

Програмне забезпечення для контролю персональних фінансів фізичних осіб для мобільної платформи/ Software for control the personnel finances of individuals using mobile platform

напрям підготовки: 6.050103 - Програмна інженерія
фахове спрямування - Програмне забезпечення систем

Бакалаврська дипломна робота

Виконав студент групи ПЗС-42
В. П. Фурман
_____

Науковий керівник:
викладач ШПАК В.Б.
_____

Бакалаврську дипломну роботу
допущено до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

РЕЗЮМЕ

Дипломна робота містить сторінки, 10 таблиць, 29 рисунків, список використаних джерел із 19 найменувань.

Метою дипломної роботи є розробка мобільного програмного забезпечення для контролю персональних фінансів фізичних осіб для мобільної платформи.

Об'єктом дослідження є процес внесення інформації про витрати та прибутки та обробка цих записів, тобто контролю персональних фінансів фізичних осіб.

Предметом дослідження є програмний продукт для контролю персональних фінансів фізичних осіб.

Методи розробки базуються на технології Java Android Studio.

Одержані результати полягають в розробці мобільного програмного забезпечення для контролю персональних фінансів фізичних осіб, яке може використовуватися на мобільних операційних системах.

Ключові слова: бюджет, прибутки, витрати, фінансовий звіт, валюта, категорії, кредит, заборгованість .

RESUME

Thesis contains pages, 10 tables, 29 figures, list of references with 19 titles.

The aim of the thesis is to develop mobile software for controlling personal finances of individuals for the mobile platform.

The object of research is the process of entering information on the costs and benefits and processing these records, that the control of personal finances of individuals.

The subject of the study is the software for controlling personal finances of individuals.

The methods of making technology based on Java Android Studio.

The results are in developing mobile software for controlling personal finances of individuals that can be used on mobile operating systems.

Keywords: budget, income, expenses, financial report, currency, category loan debt.

ЗМІСТ

ВСТУП

Актуальність теми

Правильні інструменти обліку та аналізу особистих фінансів, стануть основою благополуччя і швидкого накопичення коштів на реалізацію своїх цілей. Стануть основою того способу життя, який особа вважає достатнім для себе. В інших випадках, персональна бухгалтерія допомагає вирішити і проблему з постійними боргами і кредитами, які насправді заважають стати успішною людиною.

Мета і задачі розробки

Метою розробки є створення програмного забезпечення для ведення персональної бухгалтерії. Програма повинна відповідати наступним критеріям:

- Мати зручну форму для підрахунку доходів і витрат;
- Мати зручну форму для ведення статистики у вигляді діаграм;
- Мати можливість розподілу всіх статей доходів і витрат за категоріями і групами;
- Мати можливість чітко планувати всі статті доходів і витрат;
- Мати можливість ведення обліку боргів і виплат по кредитах;
- Створення простої бази даних;
- Мати здатність програми імпортувати і експортувати дані;
- Наявність різних додаткових інструментів (нагадувалки);
- Якісна система захисту даних і синхронізація з усіма пристроями.

В процесі розробки необхідно вирішити наступні задачі:

зробити огляд існуючих програм для організації персональної бухгалтерії;

визначити необхідні функції для реалізації програми;

провести тестування програмного засобу;

Методи, засоби та технології розробки

У даному випадку було вибрано методи аналізу та синтезу. Аналіз — це метод пізнання, який дає змогу поділити предмет на частини. Синтез, навпаки, є наслідком з'єднання окремих частин чи рис предмета в єдине ціле.

Аналіз та синтез взаємопов'язані, вони являють собою єдність протилежностей. Залежно від рівня пізнання об'єкта та глибини проникнення в його сутність застосовуються аналіз і синтез різного роду.

Практичне значення одержаних результатів

Практичне значення розробки полягає у створенні програми для організації інформаційного середовища, щодо прибутків та витрат в персональної бухгалтерії.

РОЗДІЛ І
РОЗДІЛ АНАЛІЗУ ПРЕДМЕТНОЇ ОБЛАСТІ ТА СПЕЦИФІКАЦІЇ ВИМОГ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОНТРОЛЮ ПЕРСОНАЛЬНИХ ФІНАНСІВ ФІЗИЧНИХ ОСІБ

1.1. Коротка характеристика об’єкту управління

Рано чи пізно постає питання переглянути особистого бюджету, оптимізувати його і відмовитися від неконструктивних витрат. Що ж, завдання не нове і з ним цілком можна впоратися.

Планування персонального бюджету дисциплінує у питанні фінансів, відкриває очі на не завжди конструктивні пріоритети розподілу коштів, а також може стати інструментом поліпшення фінансового становища сім’ї.

Проте, перш ніж братися за планування персонального бюджету, необхідно врахувати, що це також певна робота, яка потребує часу, уваги, концентрації і рішучості.

Багато людей досить добре почуваються і без ведення обліку персонального бюджету і при цьому успішно справляються з розподілом коштів. Тому, при прийнятті рішення в першу чергу необхідно, вияснити причини того, що спонукає взяти під контроль сімейний бюджет.

Також слід врахувати, що ведення обліку і планування персонального бюджету досить трудомістка справа і без конкретної цілі займатися нею немає сенсу.

Існує безліч схем, порад, методів ведення і планування сімейного бюджету. Проте пропонувати якийсь метод, як універсальний, недоречно. Мінливість життя і його мобільність навряд чи дозволять вам безклопітно вести сімейний бюджет по чітко обраній схемі. Зміна курсу валют, подорожчання харчових продуктів, незаплановані походи до лікаря, гаряча путівка на відпочинок та інші зваби і непередбачувані витрати здатні значно сколихнути

ваш персональний бюджет. Тому логічно обрати найбільш підходящу схему і керуватися нею.

1.2. Огляд і аналіз існуючих аналогів, що реалізують функції системи планування персональних фінансів

1.2.1. Програма «Monefy»

Як успішно вести облік фінансів? Ми знаємо, що це дуже просто. Потрібно всього лише записувати все, що Ви витрачаєте ... і не більше того! І Monefy допоможе Вам у цьому. Додавайте нові записи, коли купуєте каву або сідаєте в таксі. Всього один клік, так як не потрібно заповнювати нічого, крім суми. Це ніколи не було так швидко і зручно!

Вигляд діалогових вікон показано на рисунку 1.1.

Ви користуєтеся телефоном і планшетом? Або Ви хочете вести загальний облік фінансів з Вашою другою половинкою? Monefy ідеально підходить. Ви можете швидко і, що не менш важливо, безпечно синхронізувати будь-які пристрої через Ваш особистий Dropbox аккаунт. Це найбезпечніший спосіб, так як тільки Ви маєте доступ до даних. Додавайте або змінюйте записи, редагуйте категорії, і ці зміни тут же з'являться на інших Ваших пристроях!

Автори програми «Monefy» зазначають ключові моменти, що роблять облік ефективним і приємним:

1. Інтуїтивно зрозумілий і зручний інтерфейс. Нічого зайвого

2. Миттєве додавання нових записів

3. Зручні віджети з можливістю використання на екрані блокування

4. Перегляд витрат на красивому і інформативному графіку і детальна інформація в списку під ним

5. Управління категоріями

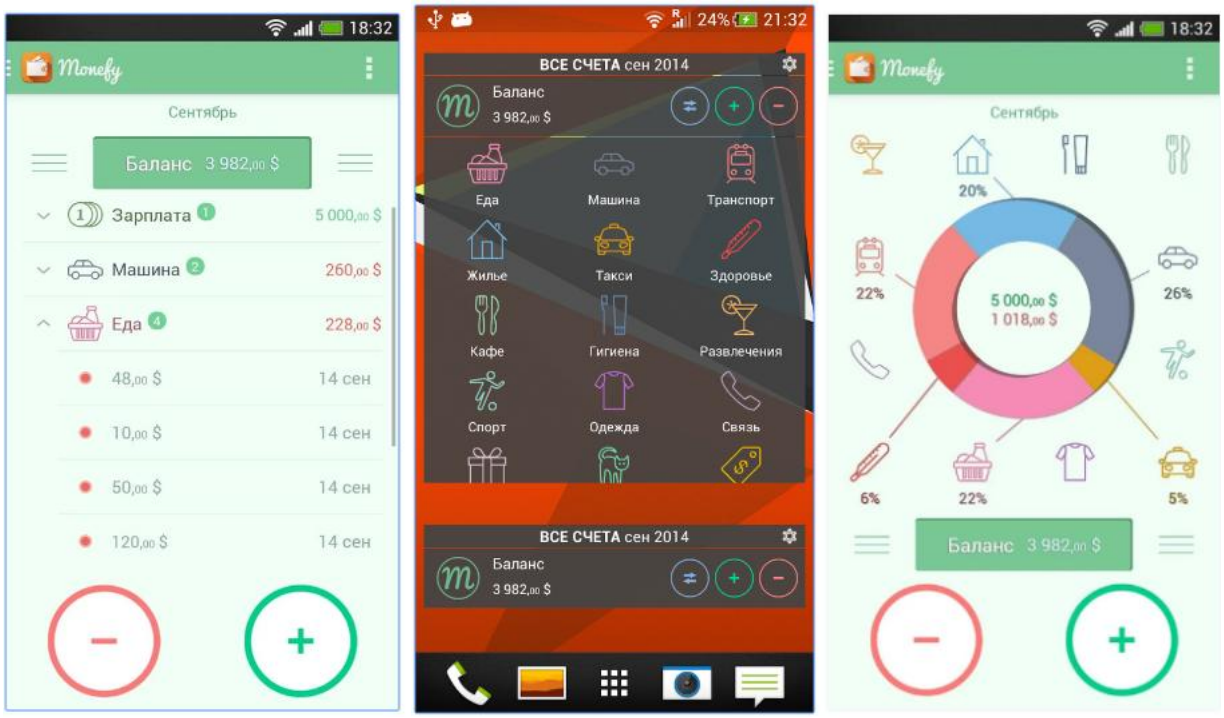

Рисунок 1.1- Діалогові вікна програми «Monefy»

6. Безпечна синхронізація через Ваш особистий Dropbox аккаунт

7. Вибір звітного періоду

8. Вибір валюти

9. Режим бюджету для більш ефективного контролю витрат за певний період

10. Створення резервних копій та експорт даних в один клік

11. Захист паролем

12. Підтримка декількох рахунків

13. Вбудований калькулятор

14. немає реклами

1.2.2. Програма «Журнал витрат»

Вигляд діалогових вікон програми «Журнал витрат» показано на рисунку 1.2.

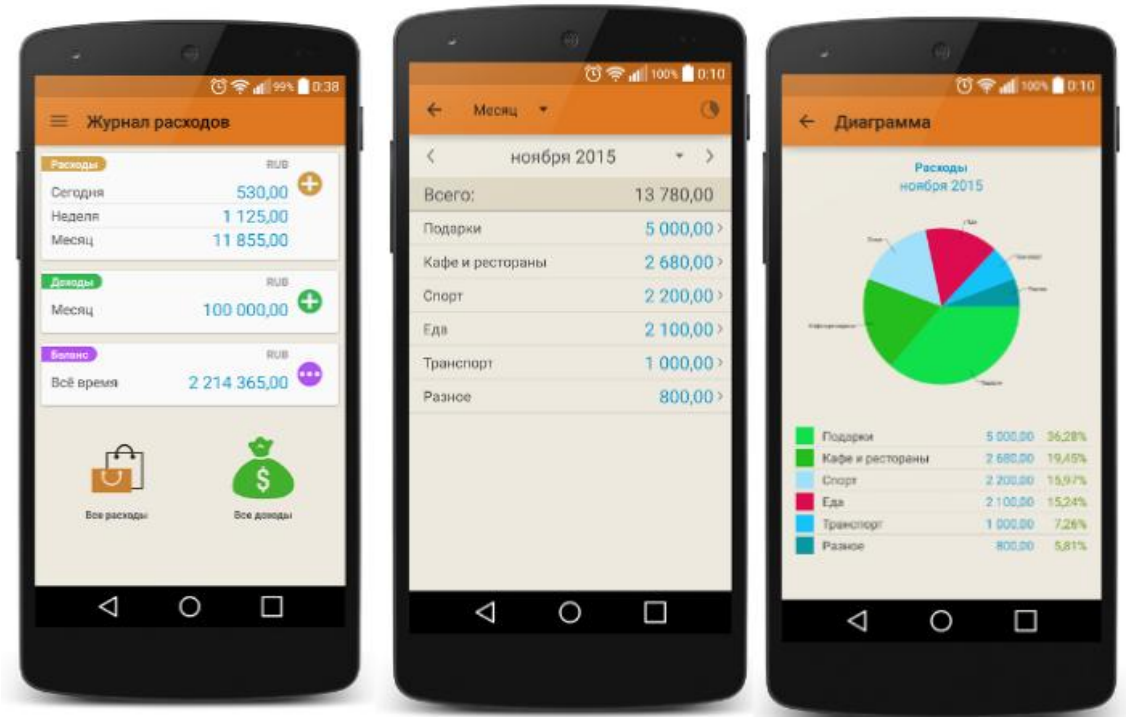

Рисунок 1.2 - Діалогові вікна програми «Журнал витрат»

Автори зазначають наступну інформацію про продукт:

2.

Журнал витрат - додаток, призначений для обліку витрат і доходів.

Додаток дозволяє просто і зручно додати інформацію про витрати в момент покупки або отримання доходу.

Зручний і інформативний інтерфейс допоможе контролювати ваші витрати і планувати бюджет.

У додатку доступні:

- додавання витрат і доходів в будь-якій валюті (список валют містить 171 позицій)

- перегляд статистики витрат і доходів за день, місяць, рік і за вказаний період

- побудова діаграм витрат і доходів за будь-який період часу

- розрахунок балансу з різниці доходів і витрат

- баланс по кожному місяцю

- можливість вибору основної валюти

- можливість додавати, видаляти і редагувати категорії

- можливість редагувати і видаляти раніше додані записи

- резервне копіювання даних

- захист паролем (бета)

- калькулятор

- експорт в csv

- курс валют онлайн

- віджети

Додаток постійно допрацьовується і додаються нові функції.

1.2.3 Програма «Мої фінанси»

Вигляд діалогових вікон програми «Мої фінанси» показано на рисунку 1.3.

Автори зазначають наступну інформацію про продукт:

Є платний контент.

Додаток сумісний з усіма вашими пристроями.

Додати в список бажань

встановити

Додаток «Мої фінанси» - це відмінний інструмент для всіх, хто хоче контролювати свої витрати. Тепер керувати домашнім бюджетом стало ще

простіше! Завдяки ретельно підібраним функцій ви зможете більше економити, а також ретельно аналізувати свої фінансові операції.

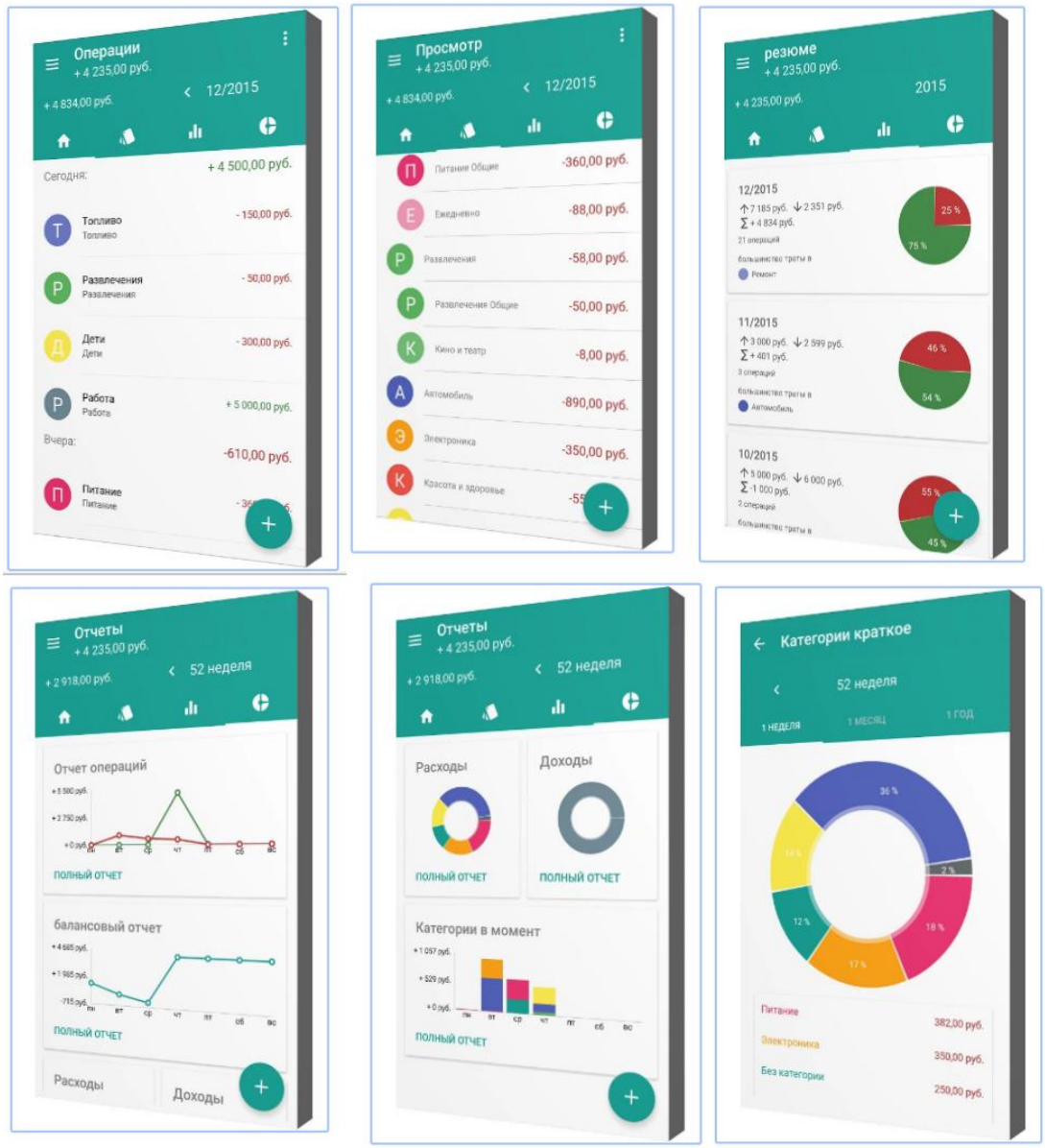

Рисунок 1.3 - Діалогові вікна програми «Мої фінанси»

Гарний і інтуїтивно зрозумілий інтерфейс допоможе вам швидко, легко і просто додавати нові і вже існуючі операції. За допомогою декількох торкань ви зможете додати доходи або витрати, перевірити баланс рахунку використовуваних рахунків (зареєстрованих вами в додатку), а також підсумувати всі витрачені і / або зароблені гроші з поділом на категорії або

періоди. Економити гроші стане простіше і приємніше. Основні функції програми:

• Додавання доходів і витрат швидко і інтуїтивно зрозумілим способом. За допомогою одного дотику можна перейти в вікно додавання нової фінансової операції, потім додати необхідну інформацію і зберегти результати. Простота і утилітарність - основні особливості програми «Мої фінанси».

• Можливість створення категорій і підкатегорій відповідно до своїх уподобань.У кожній категорії є свій колір, який можна вибрати, завдяки чому все наочно.

• Обслуговування декількох рахунків - щоб повністю контролювати власний гаманець, можна задати будь-яку кількість рахунків, але це ще не все ... Великим плюсом управлінням додатки для управління бюджетом є можливість фільтрації всіх операції по рахунках. Наше додаток дозволяє це зробити простим способом, вибравши рахунок, який буде видно в рамках всього програми в даний момент.

• Постійні доручення, завдяки яким вести домашній бюджет стає ще простіше. Можна задавати операції з певною періодичністю і додаток додасть їх за вас!Прикладом таких операцій є абонентська плата за телефон, квартплата або ваша зарплата.

• Заплановані операції, завдяки яким можна побачити суму доходів і витрат по окремих місяцях. Це особливо важливо, якщо ви плануєте інвестиції або просто любите планувати і хочете побачити, скільки ви можете заощадити в майбутніх місяцях.

• Історія з розширеними фільтрами дозволяє відстежувати свої доходи і витрати. Кожен, хто використовує подібні програми, хоче знати точну історію своїх витрат. Наше додаток надає таку можливість і дозволяє детально відстежувати домашній бюджет.

• Перегляд операцій з поділом на окремі категорії і підкатегорії дозволить вам побачити, на що ви витрачаєте найбільше грошей.Проаналізувавши ці дані, ви зрозумієте, на чому можна заощадити в майбутньому.

• Статистика - це дуже корисна функція при управлінні витратами і доходами. Вона дозволяє порівнювати по окремих місяцях співвідношення витрат і доходів.

• Графіки дозволяють аналізувати колишні операції з розбивкою на категорії або окремі роки / місяці.

• Це лише деякі функції програми. Крім того, «Мої фінанси» дозволяє вам задати перший день місяця (наприклад, якщо вам платять зарплату 10 числа), щоб підлаштуватися під ваші завдання, імпортувати або експортувати дані у файли в формат csv, щоб мати можливість ретельніше аналізувати фінанси, створювати і відновлювати резервні копії з метою забезпечення безпеки. «Мої фінанси» - це корисний додаток для управління домашнім бюджетом, яке дозволить вам отримати повний контроль над усіма фінансами. Якщо ви намагаєтеся зрозуміти, куди так швидко йдуть гроші, це додаток - саме те, що вам потрібно. Якщо у вас з'являться ідеї, як поліпшити додаток, або ви знайшли помилку, ми будемо раді отримати від вас лист на адресу info@7csolutions.com які права вимагає додаток

1.3. Специфікація вимог до програмного забезпечення для контролю персональних фінансів фізичних осіб для мобільної платформи»

Метою створення специфікації вимог до системи є визначення функцій системи та її не функціональних вимог. Першим кроком створення специфікації є опис глосарію проекту, який відображений у таблиці 1.1

Таблиця 1.1

Глосарій проекту

Термін	Опис терміну
1. Основні поняття та категорії предметної області та проекту	
Рахунок	Виражений у числах результат яких-небудь підрахунків, обчислень
Бюджет	Грошове вираження збалансованого розпису доходів і видатків об’єкту за певний період.
Витрати	Зменшення обсягу матеріальних цінностей, коштів тощо, які відбуваються в процесі свідомої людської діяльності; зменшення певних ресурсів у фізичних процесах
Дохід	Гроші або матеріальні цінності, одержувані фізичною особою внаслідок якої-небудь діяльності
2. Користувачі системи	
Приватна особа	Особа, яка здійснює розрахунки фінансів для свого персонального бюджету за допомогою проектованої системи
3. Вхідні та вихідні документи	
Панель керування	Електронна форма для переходу до основних функцій («Баланс», «Додати рахунок», «Додати ціль», «Внесення витрат», «Внесення доходу») та

	ін.
Записи	Електронна форма для перегляду записів, сортування додавання та пошуку записів.
Аналіз	Електронна форма для перегляду графіків, грошових потоків, річного балансу та звіту.
Рахунки	Електронна форма для додавання рахунку.
Бюджет	Електронна форма для створення електронного бюджету.
Цілі	Електронна форма для створення «Цілі» та визначення джерел її фінансування.
Категорії	Електронна форма для створення категорій витрат та доходів.

Опрацювавши джерела, що наведені вище приймаємо наступну специфіка вимог до системи обліку сімейного бюджету. Вимоги будемо специфікувати за допомогою діаграми варіантів використання.

Єдиний актор проектованої системи, буде взаємодіяти із всіма варіантами використання. У модель включено асоціації, що відповідають за напрямки передачі інформації між актором і варіантами використання. Здійснюємо розподіл варіантів використання по пакетам та визначаємо їх взаємодію із актором, тобто представляємо діаграму варіантів використання функціональної моделі.

На рисунку 1.4 зображено загальну діаграму варіантів використання проектованого програмного забезпечення.

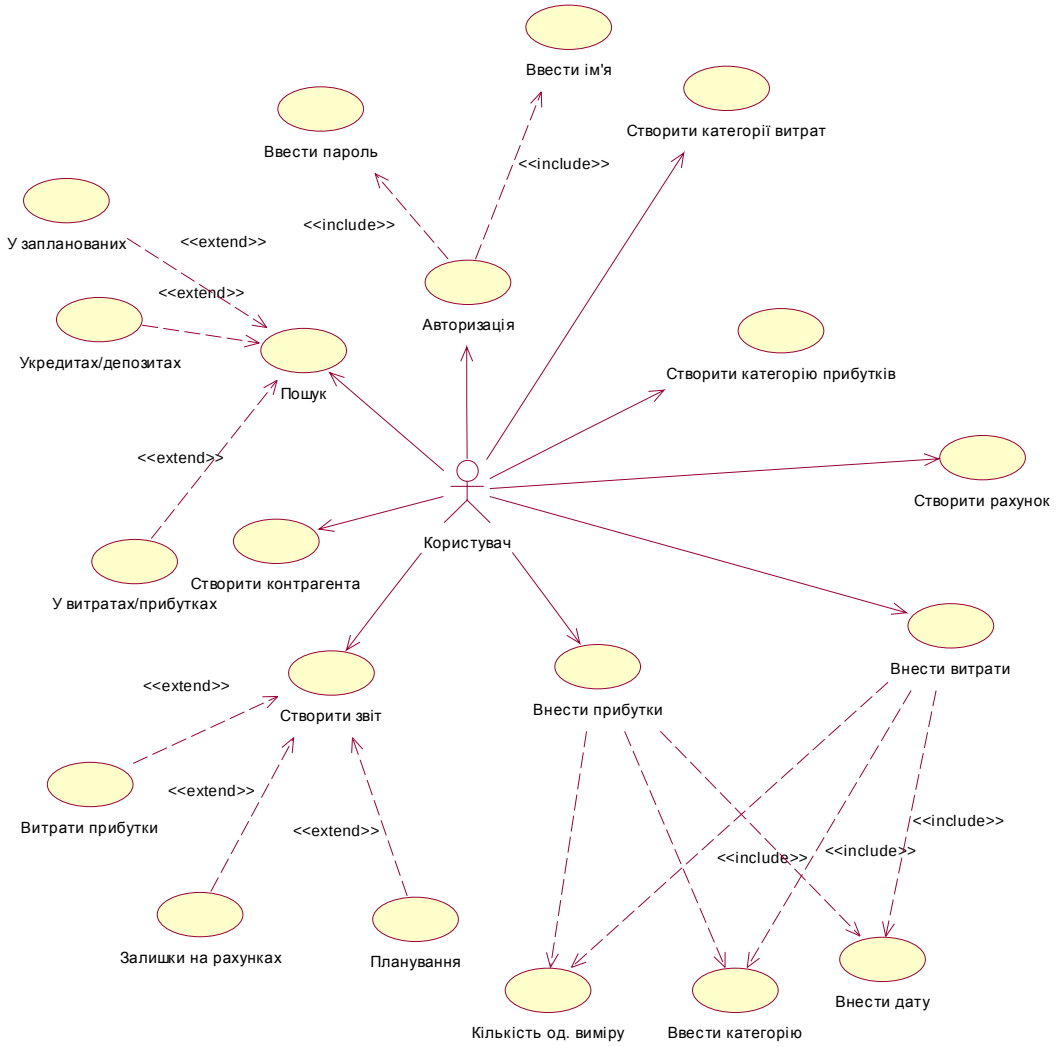

Рисунок 1.4.- Загальна діаграма варіантів використання

Далі наводимо опис варіантів використання, що реалізують основну функціональність у вигляді таблиць.

Таблиця 1.2

Варіант використання «Авторизація»

Контекст використання	Вхід в систему
Дійові особи	Користувач системи
Передумови	Початок роботи з системою
Триггер	Запуск програми на виконання
Сценарій	Ввести логін Ввести пароль Натиснути кнопку «Ok»

Постумова	Отримання доступу до елементів системи Стартове вікно «Панель керування»

Таблиця 1.3

Варіант використання «Пошук»

Контекст використання	Пошук записів за певними парамертами
Дійові особи	Користувач системи
Передумови	Перейти у вікно «Записи»
Триггер	Завдання параметрів пошуку
Сценарій	Вибрати категорію пошуку Ввести шукане слово Натиснути кнопку «Шукати»
Постумова	Отримання переліку знайдених категорій, якщо такі відсутні то повідомлення пр. відсутність.

Таблиця 1.4

Варіант використання «Створити звіт»

Контекст використання	Створення звіту за певний період та і певному вигляді
Дійові особи	Користувач системи
Передумови	Перейти у вікно «Аналіз»
Триггер	Завдання параметрів звіту
Сценарій	Вибрати період звіту Вибрати вид звіту Вибрати форму відображення звіту
Постумова	Отримати візуалізацію звіту за вибраними параметрами

Таблиця 1.5

Варіант використання «Внести прибутки»

Контекст використання	Подія отримання прибутку
Дійові особи	Користувач системи
Передумови	Перейти у вікно «Панель керування»
Триггер	Необхідність внести прибуток
Сценарій	Вибрати закладку «Дохід» Вибрати «Рахунок» Вибрати «Категорію» Внести кількість Натиснути кнопку «Записати»
Постумова	Запис у довідник прибутки Повернення у вікно «Панель керування»

Таблиця 1.6

Варіант використання «Внести витрати»

Контекст використання	Подія здійснення витрат
Дійові особи	Користувач системи
Передумови	Перейти у вікно «Панель керування»
Триггер	Необхідність внести інформацію про витрати
Сценарій	Вибрати закладку «Витрата» Вибрати «Рахунок» Вибрати «Категорію» Внести кількість Натиснути кнопку «Записати»
Постумова	Запис у довідник витрати Повернення у вікно «Панель керування»

Таблиця 1.7

Варіант використання «Створити категорію»

Контекст використання	Додати нову категорію
Дійові особи	Користувач системи
Передумови	Вибрати закладку «Ще»-> «Категорії»
Триггер	У системі відсутня необхідна категорія
Сценарій	Вибрати закладку «Витрати» або «Доходи» Внести відсутню категорію Натиснути кнопку зберегти.
Постумова	Повернутися у «Панель керування»

Далі виконаємо розкадровку варіантів використання для опису необхідних екранних форм, що використовуються для реалізації функцій.

Розкадровка варіанту використання «Авторизація» показана на рисунку 1.5.

Рисунок 1.5.- Розкадровка варіанту використання «Авторизація»

Розкадровка варіанту використання «Пошук» показана на рисунку 1.6.

Рисунок 1.6.- Розкадровка варіанту використання «Пошук»

Розкадровка варіанту використання «Створити звіт» показана на рисунку 1.7.

Рисунок 1.7.- Розкадровка варіанту використання «Створити звіт»

Розкадровка варіанту використання «Внести прибутки» показана на рисунку 1.8.

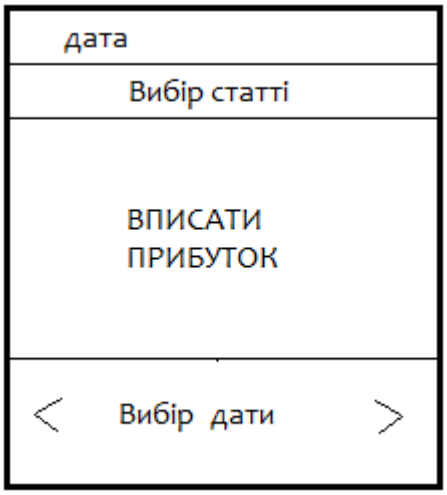

Рисунок 1.8.- Розкадровка варіанту використання «Внести прибутки»

Розкадровка варіанту використання «Внести витрати» показана на рисунку 1.9.

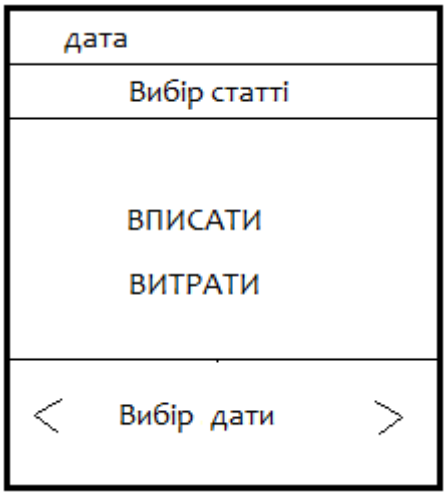

Рисунок 1.9.- Розкадровка варіанту використання «Внести витрати»

Розкадровка варіанту використання «Створити категорію» показана на рисунку 1.10.

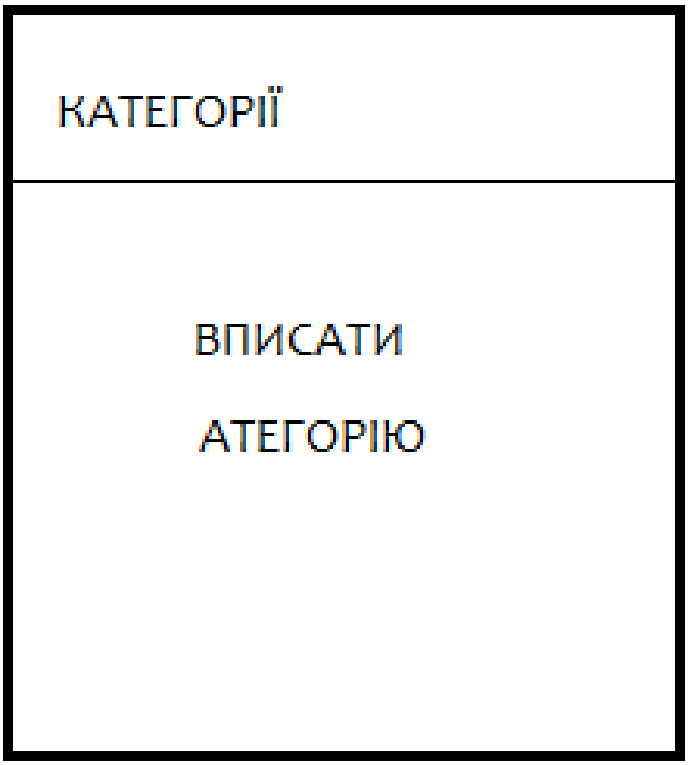

Рисунок 1.10.- Розкадровка варіанту використання «Створити категорію»

Таблиця 1.8

Специфікацію функціональних вимог

Іденти-фікатор вимог	Назва вимоги	Атрибути вимоги		
		Пріоритет	Складність	Контакт
1	Авторизація	Обов’язкова	Середня	
2	Пошук	Обов’язкова	Середня	
3	Створити звіт	Обов’язкова	Висока	
4	Внести прибутки	Обов’язкова	Середня	
5	Внести витрати	Обов’язкова	Середня	
6	Створити категорію	Рекомендована	Середня	

Таблиця 1.9

Специфікацію нефункціональних вимог

Іденти-фікатор вимог	Назва вимоги	Атрибути вимоги		
		Пріоритет	Складність	Контакт
Застосовність				
1	Авторизація	Обов’язкова	5	
2	Пошук	Обов’язкова	20	
3	Створити звіт	Обов’язкова	30	
4	Внести прибутки	Обов’язкова	15	
5	Внести витрати	Обов’яз кова	15	
6	Створити категорію	Рекомендована	20	
Надійність				
1	Авторизація	Обов’язкова	99	
2	Пошук	Обов’язкова	98	
3	Створити звіт	Обов’язкова	98	
4	Внести прибутки	Обов’язкова	98	
5	Внести витрати	Обов’язкова	98	
6	Створити категорію	Рекомендована	95	
Експлуатаційна придатність				
1	Авторизація	Обов’язкова	100	
2	Пошук	Обов’язкова	100	
3	Створити звіт	Обов’язкова	100	
4	Внести прибутки	Обов’язкова	100	
5	Внести витрати	Обов’язкова	100	
6	Створити категорію	Рекомендована	100	

Висновки до першого розділу

Проведено огляд існуючих напрацювань в області створення програмних засобів для ведення персональної бухгалтерії . Базуючись на цьому було поставлено завдання для створення власного програмного засобу, визначено варіанти використання програми та висунуто функціональні та нефункціональні вимого.

РОЗДІЛ ІІ
ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОНТРОЛЮ ПЕРСОНАЛЬНИХ ФІНАНСІВ ФІЗИЧНИХ ОСІБ ДЛЯ МОБІЛЬНОЇ ПЛАТФОРМИ

2.1. Розроблення архітектури програмної системи

Провівши ієрархічний аналіз вимог до програмного забезпечення, було виділено пакети, що відповідають підсистемам, які будуть використовуватися для представлення користувацького інтерфейсу, зв'язку з базою даних, реалізації класів предметної області.

На рисунку 2.1 показано діаграму, на якій зображено архітектуру програмного засобу для контролю персональних фінансів фізичних осіб для мобільної платформи.

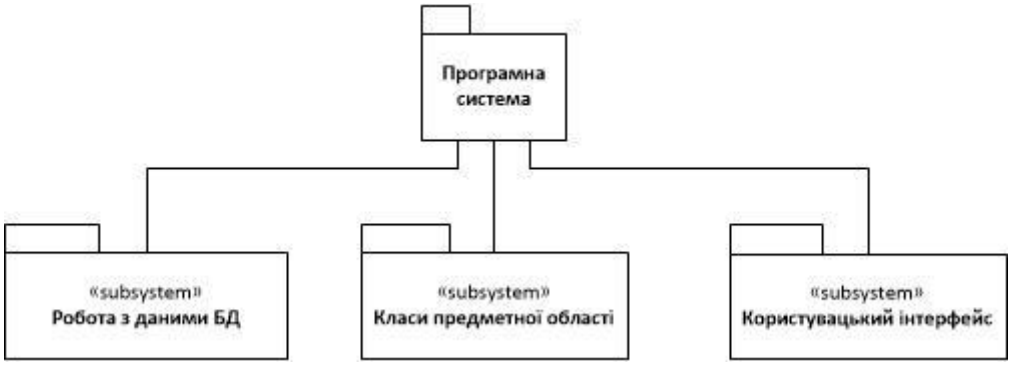

Рисунок-2.1. Архітектура програмної системи для планування справ

Для знаходження акторів, визначаємо безпосереднього користувача системи. Проаналізувавши вимоги до програмного забезпечення, встановлюємо, що єдиним актором є користувач. Отже, єдиним актором у нашій функціональній моделі буде користувач("user").

Виходячи з функції системи, визначених у попередньому розділі, встановлюємо наступні функціональні блоки системи, які співпадають з варіантами використання:

Авторизація- «Authorization».

Пошук - «Search».

Створити звіт - «Create_Report».

Внести прибутки- «Make_profits».

Внести витрати- «Make_costs».

Створити категорію- «New_Category».

Деталізуємо обрані варіантів використання і модулі системи з використанням діаграми модулів системи рисунок 2.2.

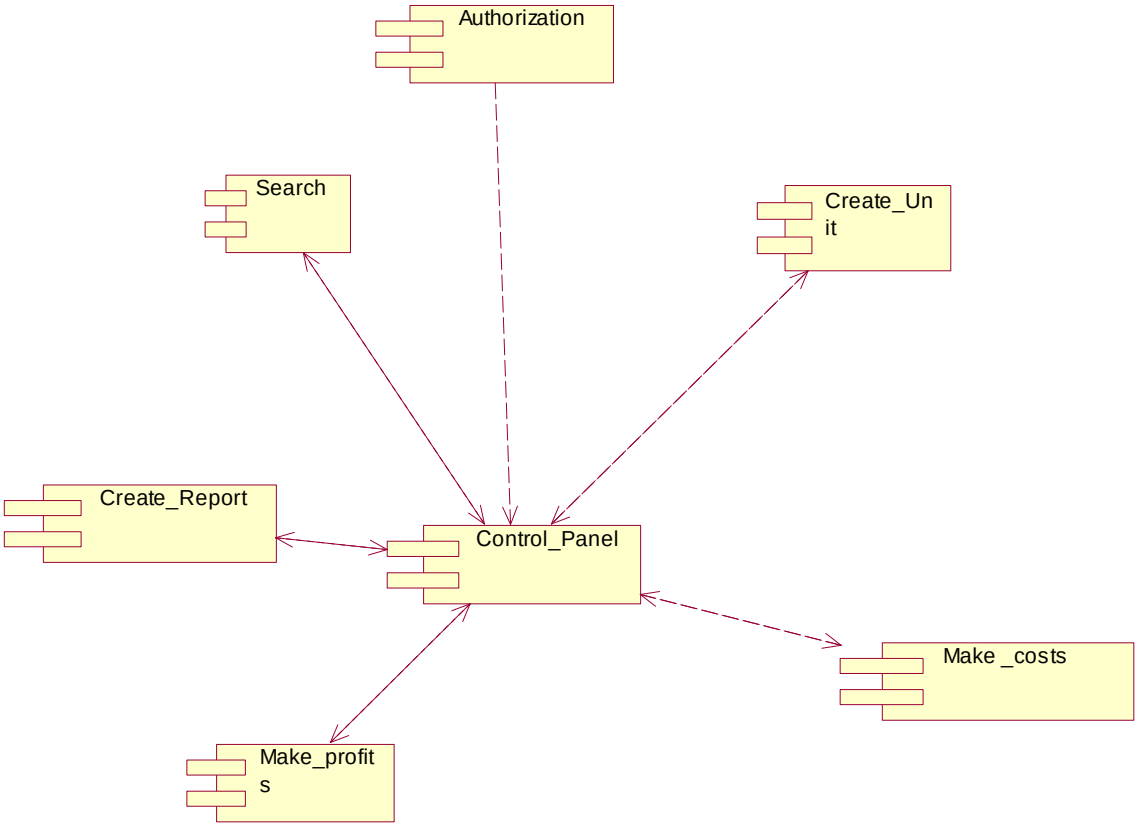

Рисунок-2.2 Структура схеми програмної системи для контролю персональних фінансів фізичних осіб для мобільної платформи

Для реалізації вибраної системи вибираємо об’єктно-орієнтований підхід до проектування та програмування. Тобто наступним кроком є визначення діаграми класів (рисунок 2.3)

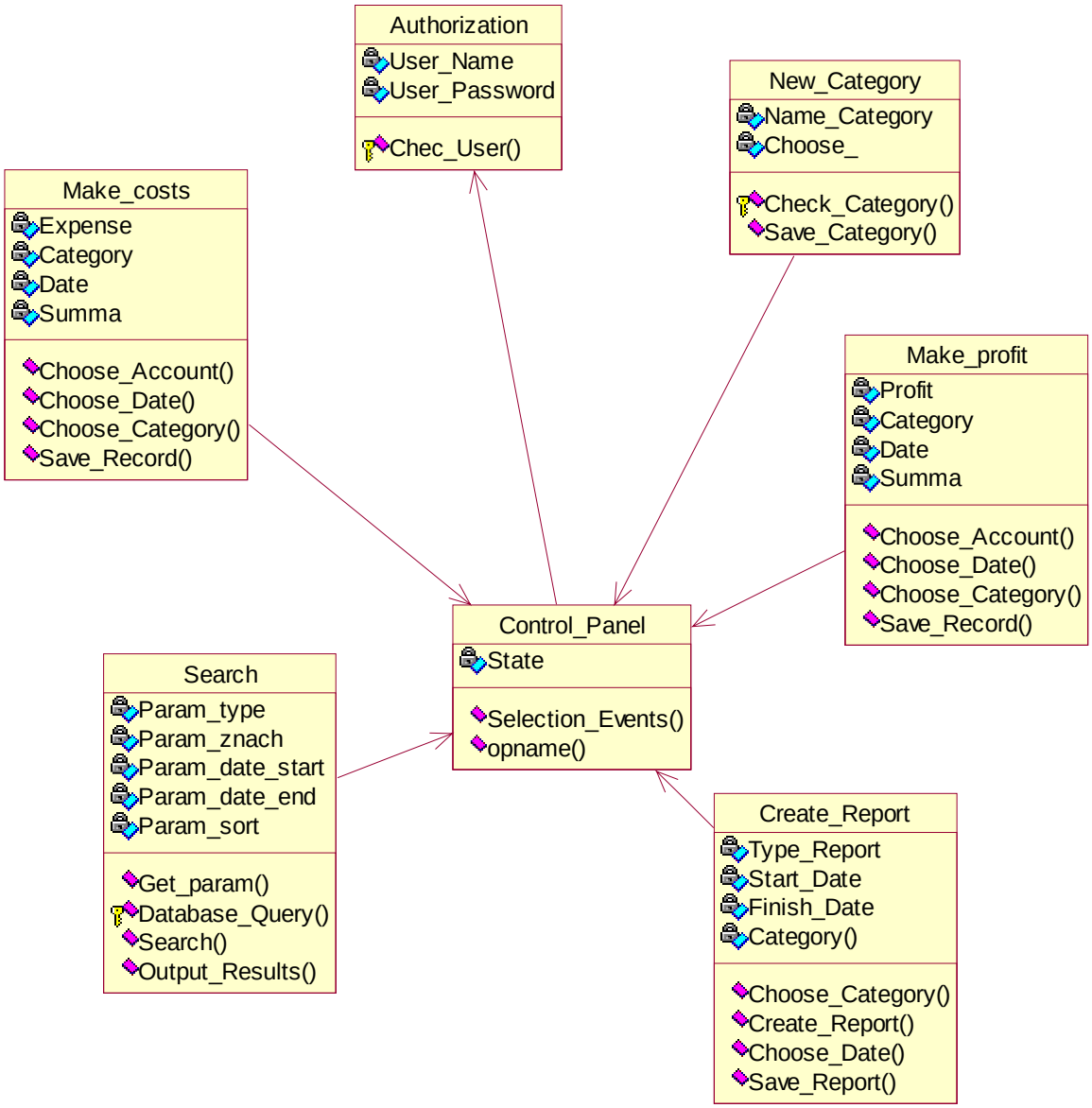

Рисунок-2.3 Діаграма класів програмної системи для контролю персональних фінансів фізичних осіб для мобільної платформи

2.2. Проектування структури бази даних

Проектуємо архітектуру баз даних (БД). Всі архітектури представляються нотацією UML і при потребі, доповнюються текстовими описами.

В ході проектування архітектури БД, було обрано для її відтворення реляційну модель даних та виділено наступні відношення, що відображені на наступних рисунках рисунки 2.4-2.8.

Users | Category | Unit | Event | Report | Схема данных

Имя поля	Тип данных	Описание
ID	Счетчик	
User_name	Текстовый	Ім'я користувача
Password	Текстовый	Пароль користувача

Рисунок 2.4- Таблиця бази даних «Users»

Users | Category | Unit | Event | Report | Схема данных

Имя поля	Тип данных	Описание
ID	Счетчик	
Name_Category	Текстовый	Назва категорії за якими здіснюються олперації

Рисунок 2.5- Таблиця бази даних «Category»

Users | Category | Unit | Event | Report | Схема данных

Имя поля	Тип данных	Описание
ID	Счетчик	
Date_Event	Дата/время	Дата події
Type_Event	Текстовый	Тип події (прибуток, витрати)
Sum	Числовой	Сума в грошових одиницях
Cantegory_Event	Числовой	Категогія події (стаття витрати, прибутку)
Unit_event	Числовой	Одиниця виміру категорії події

Рисунок 2.6- Таблиця бази даних «Event»

Users | Category | Unit | Event | Report | Схема данных

Имя поля	Тип данных	Описание
ID	Счетчик	
Date_Start	Дата/время	Дата початку формування звіту
Date_end	Дата/время	Дата кінця формування звіту
Category	Числовой	Категорія за якою формується звіт
Income	Денежный	Прибутки за категорією
Costs	Денежный	Витрати за категорією
Balance	Денежный	Баланс за категорією
User	Числовой	Користувач за яким формується звіт

Рисунок 2.7- Таблиця бази даних «Report»

Діаграма реляційної моделі даних зображена на рисунку 2.8.

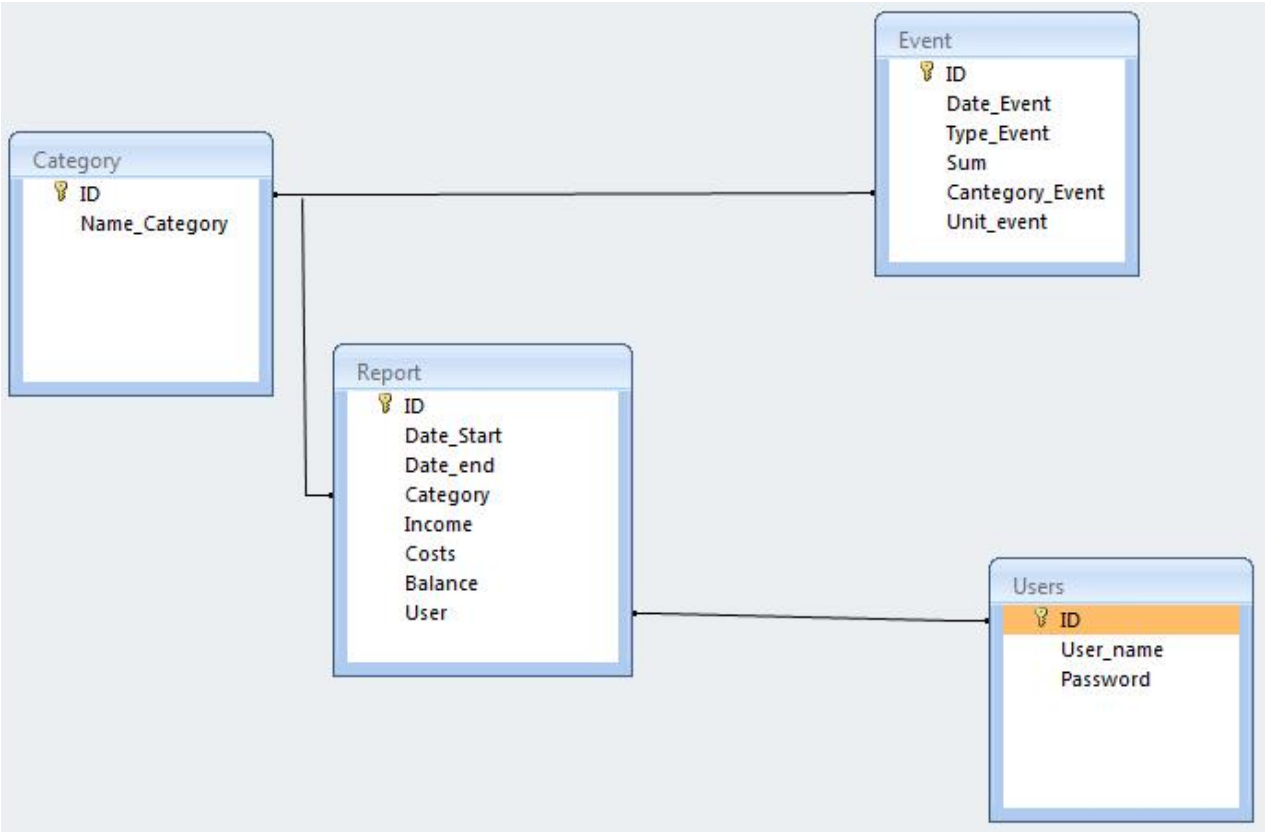

Рисунок 2.8- ER діаграма бази даних програмного забезпечення для контролю персональних фінансів фізичних осіб для мобільної платформи

Висновки до другого розділу

В другому розділі запроектовано програму для Програмного забезпечення контролю персональних фінансів фізичних осіб для мобільної платформи, також запроектовано до неї база даних. Проектування здійснювалося з використанням об’єктно-орієнтованого підходу до проектування, а також використовувалася UML діаграми для нотацій.

РОЗДІЛ III

РОЗДІЛ РЕАЛІЗАЦІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ КОНТРОЛЮ ПЕРСОНАЛЬНИХ ФІНАНСІВ ФІЗИЧНИХ ОСІБ ДЛЯ МОБІЛЬНОЇ ПЛАТФОРМИ

3.1. Програмна реалізація проекту

3.1.1. Вибір засобу створення системи

Порівняємо Visual Studio і з аналогічним Eclipse.

При розробці програми Android необхідно запускати середовище Eclipse для Android і створвати новий проект (File (Файл)>New (Створити)> Android Application Project (Проект програми Android), як показано на наступному рисунку (рисунок 3.1).

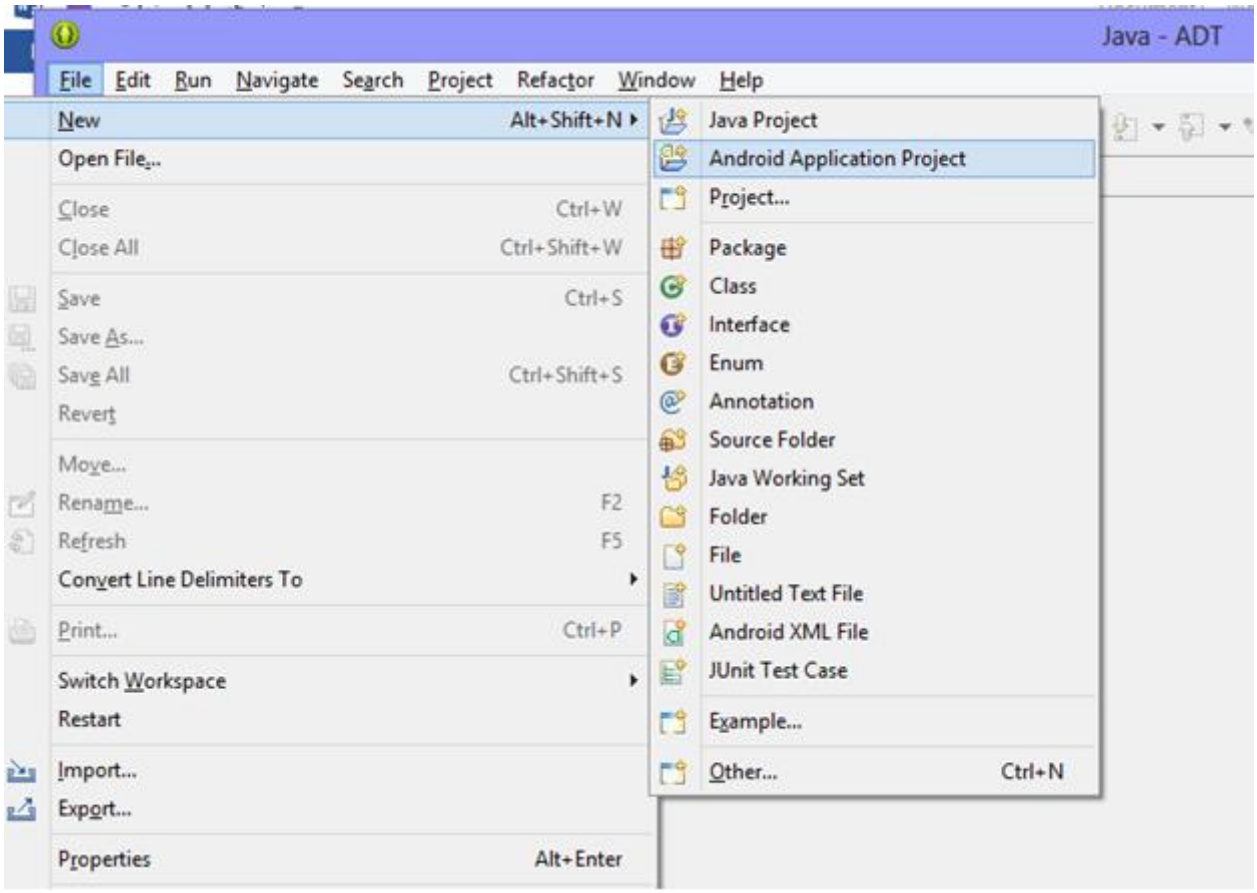

Рисунок 3.1- Вікностворення проекту за допомогою Eclipse

Після створення нового проекту Android в Eclipse можна вибрати тип дій, який показано на наступному рисунку рисунок 3.2.

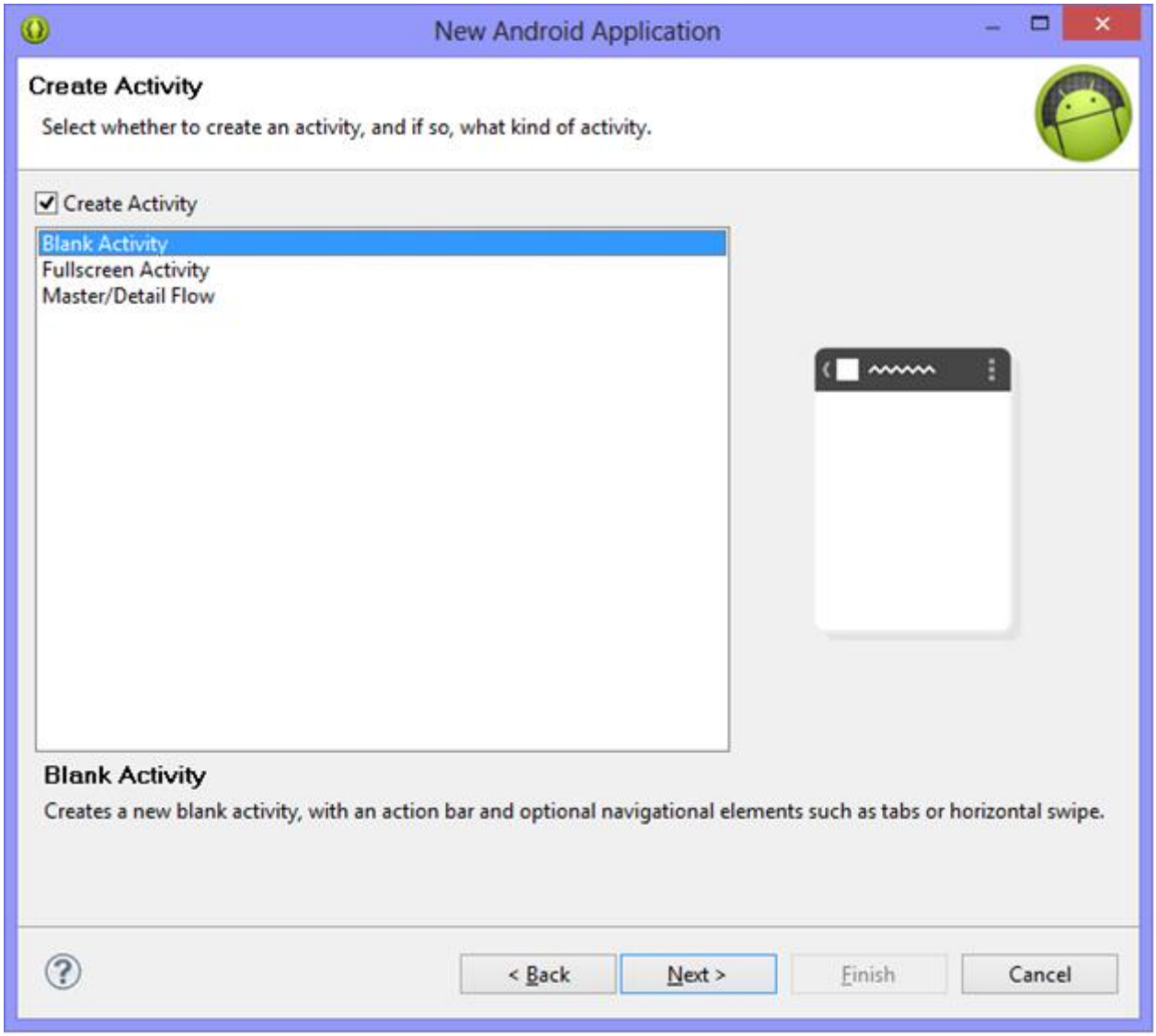

Рисунок 3.2- Вікно створення проекту за допомогою Eclipse

При запуску Microsoft Visual Studio можна побачити схожу початкову сторінку , яка показана на наступному рисунку рисунок 3.3.

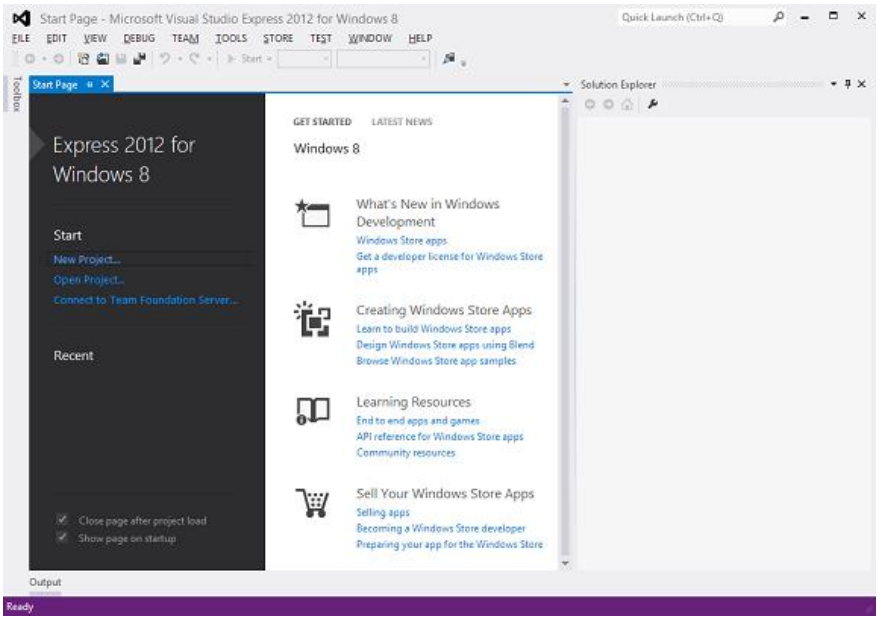

Рисунок 3.3- Вікно створення проекту за допомогою Microsoft Visual Studio

Щоб створити новий додаток, необхідно почати зі створення проекту за допомогою однієї з наступних дій.

На початковій сторінці клацніть Створити проект .

У меню Файл виберіть Створити проект .

Тут можна вибрати один з декількох шаблонів проекту (див. рисунок 3.4).

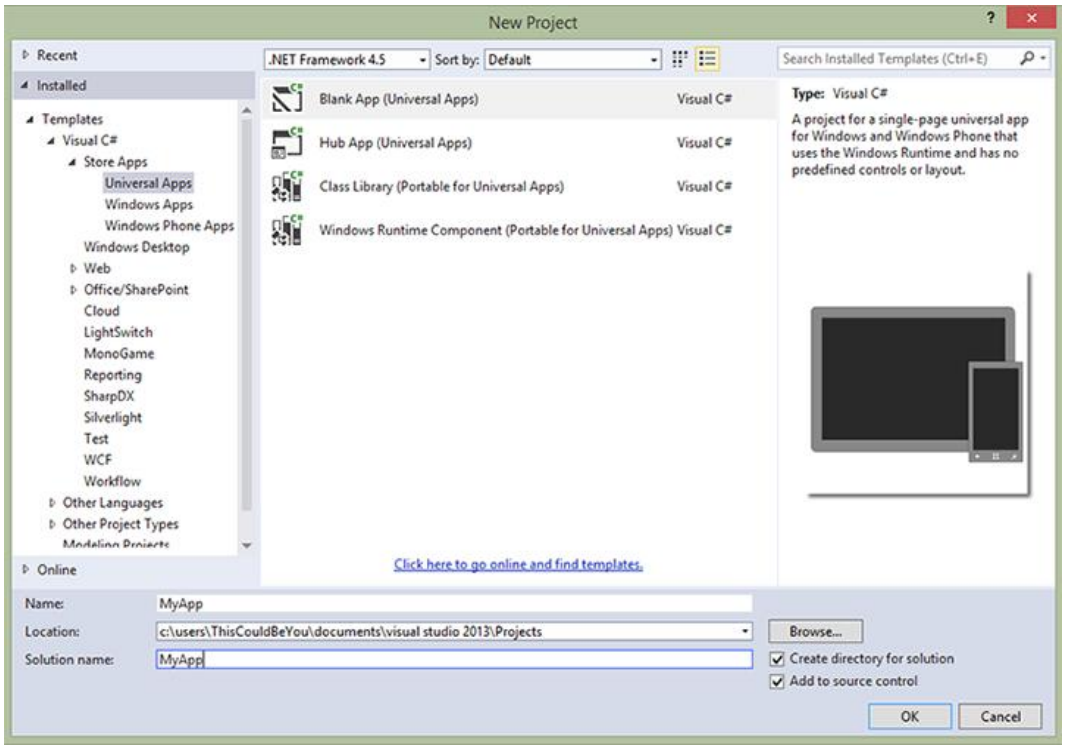

Рисунок 3.4- Вікно створення проекту за допомогою Microsoft Visual Studio, вибір типу проекту

Для роботи необхідно покроково вибрати наступні елементи Visual C# , а потім послідовно елементи Додатки Магазину, Універсальні додатки і «Порожній додаток» (універсальні додатки). У полі Ім'явведіть "MyApp" і натисніть кнопку ОК. Visual Studio створить проект і відобразить його на екрані. Тепер все готово для проектування програми і додавання коду.

На відміну від проекту Android в Eclipse, де ми створюємо по одному дії за раз, Visual Studio включає шаблони проектів з декількома сторінками і переходами між цими сторінками. Шаблон «Порожній додаток» , з іншого боку, створює додаток тільки з однією сторінкою. Але не хвилюйтеся - ми додамо другу сторінку і переходи пізніше в цій статті.

3.1.2. Вибір мови програмування

Перш ніж продовжити, необхідно дізнатися, які мови програмування можна вибрати для розробки додатків Windows і додатків Windows Phone. Крім мови C #, можна використовувати мови програмування Visual Basic, C++ і JavaScript.

Вибираємо мову програмування C #, слід зазначити, що інші мови надають унікальні переваги, якими ви, можливо, захочете скористатися. Наприклад, якщо для застосування першочерговим завданням є продуктивність (особливо для додатків, вимогливих до графічних ресурсів), то рекомендується вибрати мову C ++. Для розробників додатків на Visual Basic найкращим чином підійде Microsoft Visual Basic версії Microsoft .NET. Веб-розробники гідно оцінять JavaScript з HTML5 і бібліотекою Windows для JavaScript (WinJS).

3.1.3. Реалізація архітектури ПЗ

Відповідно до запроектованих діалогових форм, у першому розділі (рисунки 1.6-1.12) реалізовуємо відповідні форми у вибраному середовищі програмування.

На рисунку 3.5 діалогове вікно входу в систему

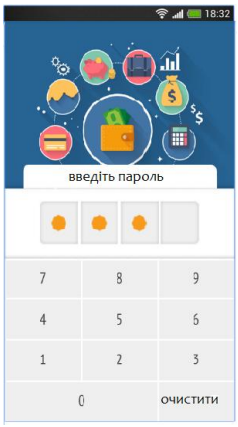

Рисунок 3.5- Діалогове вікно для реалізації варіанту використання «Авторизація»

На рисунку 3.6 діалогове вікно пошуку необхідної інформації

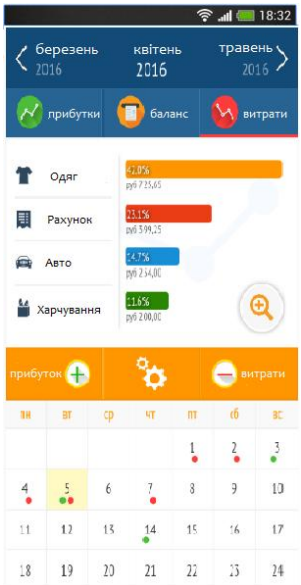

Рисунок 3.6- Діалогове вікно для реалізації варіанту використання «Пошук»

На рисунку 3.7 діалогове вікно створення звіту

Рисунок 3.7- Діалогове вікно для реалізації варіанту використання «Створити звіт»

На рисунку 3.8 показано діалогове вікно внесення прибутків

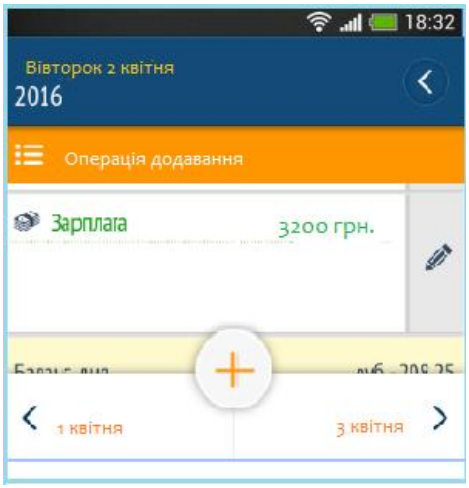

Рисунок 3.8- Діалогове вікно для реалізації варіанту використання «Внести прибутки»

На рисунку 3.9 показано діалогове вікно внесення витрат

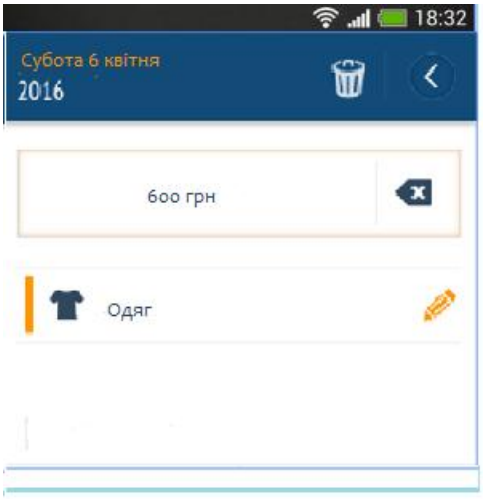

Рисунок 3.9- Діалогове вікно для реалізації варіанту використання «Внести витрати»

На рисунку 3.10 показано діалогове вікно для внесення нової категорії витрат

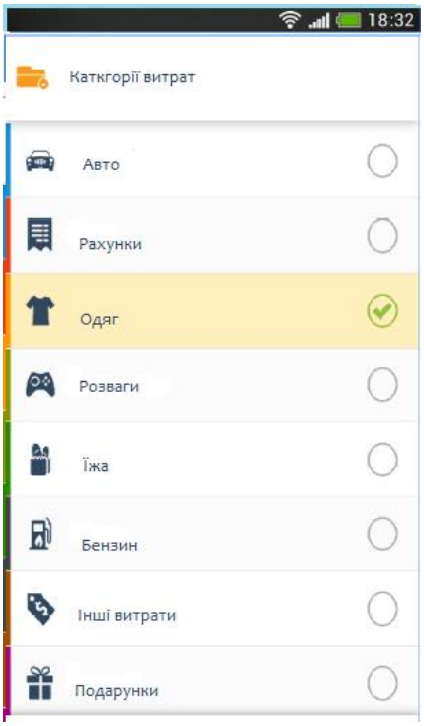

Рисунок 3.10- Діалогове вікно для реалізації варіанту використання «Створити категорію»

3.2. . Програмна реалізація бази даних

Реалізація архітектури баз даних

Визначивши та описавши архітектуру баз даних, її було реалізовано у середовищі керування базами даних MS SQL Server 2008, дивись рисунок 3.12.

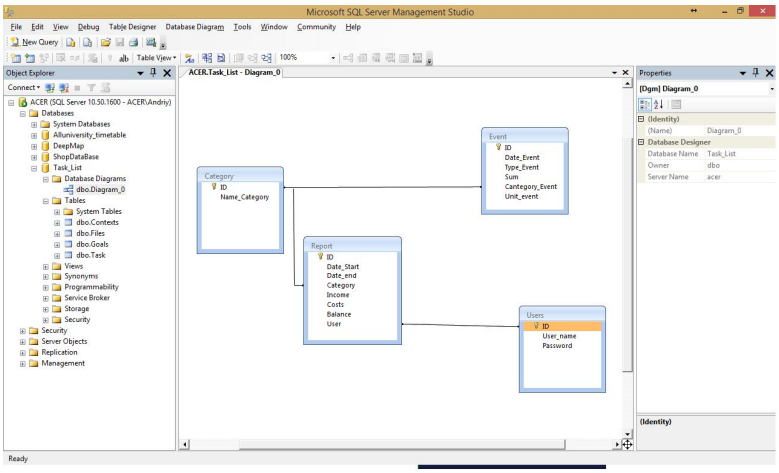

Рисунок 3.12- Реалізація бази даних

На рисунку 3.12 зображено структуру основного відношення у нашій базі даних, що призначене для зберігання інформації по витратам за прибутках.

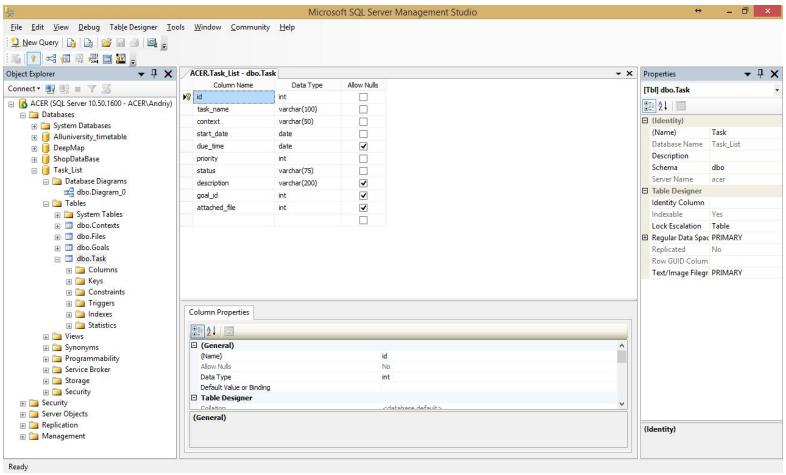

Рисунок 3.2- Відношення «Прибутки та Витрати» бази даних системи

Висновки до третього розділу

В третьому розділі описано процес кодування (створення програмного коду) програмного засобу для контролю персональних фінансів фізичних осіб для мобільної платформи з використанням мови програмування C#. В цьому розділі також показані діалогові вікна для вирішення конкретних необхідних задач.

РОЗДІЛ IV

РОЗДІЛ ТЕСТУВАННЯ ТА ДОСЛІДНОЇ ЕКСПЛУАТАЦІЇ

4.1. Тестування

Тестування є важливим етапом в життєвому циклі розробки програмного забезпечення. Тестування можна проводити на різних рівнях.

Важливі всі рівні тестування, незалежно від використовуваних моделей і методологій.

1 Модульне тестування (Unit testing). Цей рівень тестування дозволяє перевірити функціонування окремо взятого елемента системи. Що вважати елементом - модулем системи визначається контекстом. Найбільш повно даний вид тестів описаний в стандарті IEEE 1008-87 "Standard for Software Unit Testing", що задає інтегровану концепцію систематичного і документованого підходу до модульного тестування.

2 Інтеграційний тестування (Integration testing). Даний рівень тестування є процесом перевірки взаємодії між програмними компонентами / модулями. Класичні стратегії інтеграційного тестування - "зверху-вниз" і "знизу-вгору" - використовуються для традиційних, ієрархічно структурованих систем і їх складно застосовувати, наприклад, до тестування слабосвязанних систем, побудованих в сервісно-орієнтованих архітектурах (SOA).

3 Системне тестування (System testing). Системне тестування охоплює цілком всю систему. Більшість функціональних збоїв повинна бути ідентифікована ще на рівні модульних і інтеграційних тестів. У свою чергу, системне тестування, зазвичай фокусується на нефункціональних вимогах - безпеки, продуктивності, точності, надійності тощо На цьому рівні також тестуються інтерфейси до зовнішніх додатків, апаратного забезпечення, операційному середовищі тощо.

Цілі тестування. Тестування проводиться відповідно до визначених цілей (можуть бути задані явно або неявно) і різним рівнем точності. Визначення мети точним чином дозволяє забезпечити контроль результатів тестування.

Тестові сценарії можуть розроблятися як для перевірки функціональних вимог (відомі як функціональні тести), так і для оцінки функціональних вимог. При цьому, існують такі тести, коли кількісні параметри і результати тестів можуть лише опосередковано говорити про задоволення цілям тестування (наприклад, "usability" - легкість, простота використання, в більшості випадків, не може бути явно описана кількісними характеристиками).

Можна виділити наступні, найбільш поширені і обгрунтовані цілі (а, відповідно, види) тестування:

1 Приймальне тестування. Перевіряє поведінку системи на предмет задоволення вимог замовника. Це можливо в тому випадку, якщо замовник бере на себе відповідальність, пов'язану з проведенням таких робіт, як сторона "приймаюча" програмну систему, або специфіковані типові завдання, успішна перевірка (тестування) яких дозволяє говорити про задоволення вимог замовника. Такі тести можу проводитися як із залученням розробників системи, так і без них.

2 Установче тестування. З назви випливає, що дані тести проводяться з метою перевірки процедури інсталяції системи в цільовому оточенні.

3 Альфа- і бета-тестування. Перед тим, як випускається програмне забезпечення, як мінімум, воно повинно проходити стадії альфа (внутрішнє пробне використання) і бета (пробне використання з залученням відібраних зовнішніх користувачів) версій. Звіти про помилки, що надходять від користувачів цих версій продукту, обробляються відповідно до визначених процедур, що включають підтверджуючі тести (будь-якого рівня), що

проводяться фахівцями групи розробки. Даний вид тестування не може бути заздалегідь спланований.

4 Функціональні тести / тести відповідності. Ці тести можуть називатися по різному, проте, їх суть проста - перевірка відповідності системи, що пред'являються до неї вимогам, описаним на рівні специфікації поведінкових характеристик.

5 Досягнення і оцінка надійності. Допомагаючи ідентифікувати причини збоїв, тестування має на увазі і підвищення надійності програмних систем. Випадково генеруються сценарії тестування можуть застосовуватися для статистичної оцінки надійності. Обидві цілі - підвищення і оцінка надійності - можуть досягатися при використанні моделей підвищення надійності.

6 Регресійне тестування. Визначення успішності регресійних тестів (IEEE 610-90 "Standard Glossary of Software Engineering Terminology") говорить: "повторне вибіркове тестування системи або компонент для перевірки зроблених модифікацій не повинно призводити до непередбачуваних ефектів". На практиці це означає, що якщо система успішно проходила тести до внесення модифікацій, вона повинна їх проходить і після внесення таких. Основна проблема регресійного тестування полягає в пошуку компромісу між наявними ресурсами і необхідністю проведення таких тестів в міру внесення кожної зміни. Певною мірою, завдання полягає в тому, щоб визначити критерії "масштабів" змін, з досягненням яких необхідно проводити регресивні тести.

7 Тестування продуктивності. Спеціалізовані тести перевірки задоволення специфічних вимог, що пред'являються до параметрів продуктивності. Існує особливий підвид таких тестів, коли робиться спроба досягнення кількісних меж, обумовлених характеристиками самої системи і її операційного оточення.

8 Навантажувальне тестування. Необхідно розуміти відмінності між розглянутим вище тестуванням продуктивності з метою досягнення її

реальних (досяжних) можливостей продуктивності і виконанням програмної системи с підвищенням навантаження, аж до досягнення запланованих показників і далі, з відстеженням поведінки на всьому протязі підвищення завантаження системи.

9 Порівняльне тестування. Одиничний набір тестів, що дозволяють порівняти дві версії системи.

10 ремонтно-відновлювальні тести. Мета - перевірка можливостей рестарту системи в разі непередбачуваної катастрофи (disaster), що впливає на функціонування операційного середовища, в якій виконується система.

11 Конфігураційне тестування. У випадках, якщо програмне забезпечення створюється для використання різними користувачами (в термінах "ролей"), даний вид тестування спрямований на перевірку поведінки і працездатності системи в різних конфігураціях.

12 Тестування зручності і простоти використання. Мета - перевірити, наскільки легко кінцевий користувач системи може її освоїти, включаючи не тільки функціональну складову - саму систему, а й її документацію; наскільки ефективно користувач може виконувати завдання, автоматизація яких здійснюється з використанням даної системи; нарешті, наскільки добре система застрахована (з точки зору потенційних збоїв) від помилок користувача.

13 Розробка, керована тестуванням. По-суті, це не стільки техніка тестування, скільки стиль організації процесу розробки, життєвого циклу, коли тести є невід'ємною частиною вимог (і відповідних специфікацій) замість того, щоб розглядатися незалежної діяльністю з перевірки задоволення вимог програмної системою.

4.1.1. Ручне тестування

Ручне тестування призначеного для користувача інтерфейсу проводиться тестувальником-оператором, який керується в своїй роботі описом тестових прикладів у вигляді набору сценаріїв. Кожен сценарій включає перерахування послідовності дій, які повинні виконати оператор, і опис важливих для аналізу результатів тестування у відповідь реакцій системи, відбиваних в призначеному для користувача інтерфейсі.

Форма запису сценарію для проведення ручного тестування - таблиця, в якій в одній колонці описані дії (кроки сценарію), в іншій - очікувана реакція системи, а третя призначена для запису того, чи співпала очікувана реакція системи з реальною і перерахування неспівпадань.

Результати проведення тестування наведені в таблиці 4.1.

Таблиця 4.1.

Результати проведення ручного тестування

№ п/п	Дія	Реакція системи	Результат
1.	Запустіть програму	Появляється стартове вікно програми рисунок 3.5	Вірно
2.	Ввести логін та пароль та натисніть «Вхід»	Перехід до діалогового вікна пошуку необхідної інформації рис.3.6	Вірно
3.	Вибрати необхідні параметри пошуку вибрати команду «шукати»	Появитьсь інформація про записи з необхідною інформацією рис.3.6.	Вірно
4.	У вікні створення звіту вбрати аналіз витрат,	Поява звіту у вибраній формі	Вірно

	період, вибрати категорії та рахунки		
5.	У вікні внесення прибутків вбрати рахунок, категорію, дату, натиснути кнопку «+»	Відповідна інформація буде записана в базу даних	Вірно
6.	У вікні внесення витрат вбрати рахунок, категорію, дату, натиснути кнопку «Записати»	Відповідна інформація буде записана в базу даних рис.3.8	Вірно
7.	У ві кні внесення нової категорії внести назву категорії та натиснути кнопку «Зберегти»	Відповідна інформація буде записана в базу даних рис 3.9	Вірно
8.	У вікні внесення нової категорії витрам внести назву одиниці виміру та натиснути кнопку «Зберегти»	Відповідна інформація буде записана в базу даних	Вірно

4.2. Розгортання програмного продукту

Для розгортання програмного продукту необхідно отримати інсталяційний пакет, зберегти його на постійному носії та запустити. В результаті запуску такого файлу в режимі діалогу на комп’ютер буде скопійовано необхідні для роботи файли у вказані директорії.

Висновки до четвертого розділу

В четвертому розділі проведено тестування програмного засобу, тут показано його придатність до використання, а також надано інформацію, що до його розгортання. Також є інструкція користувачу, яка охоплює всі сценарії роботи з засобом.

ВИСНОВКИ

В результаті проведеного вивчення питання стану програмного забезпечення для контролю персональних фінансів фізичних осіб для мобільної платформи було поставлена задача створити своє аналогічне програмне забезпечення з використанням технології .Java для операційної системи Android з використанням системи розробки Android Studio. Це дало можливість створити програмне забезпечення мобільного типу. Тестування показало коректну роботу додатку. В четвертому розділі також наведено інструкцію до використання програмного забезпечення.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Г. Шилдт. Полный справочник по C#. – М.: Издательский дом "Вильямс", 2008.

2. Вигерс Карл Разработка требований к программному обеспечению. Пер, с англ. - М.:Издательско-торговый дом «Русская Редакция», 2004. - 576с.

3. Марка Д.А. Методология структурного анализа и проектирования – С.-Пб.: Питер, 1995. – 235 с.

4. Лешек А. Мацяшек Анализ требований и проектирование систем. Разработка информационных систем с использованием UML.: Пер. с англ. - М.: Издательский дом "Вильямс". - 2002. - 432 с.

5. Орлик С., Булуй Ю. Введение в программную инженерию и управление жизненным циклом ПЗ Программная инженерия. Программные требования. Режим доступу: http://www.sorlik.ru/swebok/3-1-software_engineering_requirements.pdf

6. Фаулер Скотт До. UML в короткому викладі. Застосування стандартної мови об'єктного моделювання: Пер. з англ. – М.:Мир, 1999. – 191 с.

7. Алістер Коберн. Сучасні методи опису функціональних вимог до систем

8. Л.Новіков. Введення в Rational Unified Process. - Режим доступу: http://www.interface.ru/rational/interface/151199/rup/main.htm

9. Фаулер М., Скотт К. UML в кратком изложении. Применение стандартного языка объектного моделирования: Пер. с англ. – М.:Мир, 1999. – 191 с.

10. Маклаков С.В. Bpwin, Erwin, Case-средства разработки информационных систем. – Москва ―ДиалогМифи‖ – 2000

11. Орлов С. Технологии разработки программного обеспечения: Учебник. – СПб.: Питер, 2002. – 464 с.

12. Каменова, Громов. Моделирование бизнеса. Методология ARIS. — М.: Весть-МетаТехнология, 2001.

13. Э. Троелсен. C# и платф орма .NET. Библиотека программиста. – СПб. : Питер, 2007.

14. Т.П. Караванова. Основи алгоритмізації та програмування. 750 задач з рекомендаціями та прикладами. – К.: Форум, 2002.

15. Э. Кингсли-Хьюджес, К. Кингсли-Хьюджес. C# 2005. Справочник программиста. – М.: ООО «ИД Вильямс», 2007.

16. Б. Керниган, Р. Пайк. Практика программирования. – СПб.: «Невский диалект», 2001.

17. В. О. Грязнова, С. В. Єфіменко. Основи методології програмування. – К.: ВПЦ «Київський університет», 2005.

18. Г. С. Иванова. Основы программирования: Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002.

19. Г. С. Иванова. Технология программирования: Учебник для вузов. – М.: Изд-во МГТУ им. Н.Э. Баумана, 2002.

Додаток А

Код програми

```
<?xml  version = "1.0"  encoding = "utf-8" ?>
<LinearLayout
 xmlns:android =
"http://schemas.android.com/apk/res/android"
 android:layout_width = "fill_parent"
 android:layout_height = "fill_parent"
 android:orientation = "vertical" >
<LinearLayout
 android:id = "@+id/linearLayout1"
 android:layout_width = "match_parent"
 android:layout_height = "wrap_content" >
<TextView
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:text = "Name"
 android:layout_marginLeft = "5dp"
 android:layout_marginRight = "5dp" >
</TextView>
<EditText
 android:id = "@+id/etName"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:layout_weight = "1" >
<requestFocus >
```

```
</requestFocus>
</EditText>
</LinearLayout>
<LinearLayout
 android:id = "@+id/linearLayout3"
 android:layout_width = "match_parent"
 android:layout_height = "wrap_content" >
<TextView
 android:id = "@+id/textView2"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:text = "Email"
 android:layout_marginLeft = "5dp"
 android:layout_marginRight = "5dp" >
</TextView>
<EditText
 android:id = "@+id/etEmail"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:layout_weight = "1" >
</EditText>
</LinearLayout>
<LinearLayout
 android:id = "@+id/linearLayout2"
 android:layout_width = "match_parent"
 android:layout_height = "wrap_content" >
<Button
 android:id = "@+id/btnAdd"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
```

```
 android:text = "Add" >
</Button>
<Button
 android:id = "@+id/btnRead"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:text = "Read" >
</Button>
<Button
 android:id = "@+id/btnClear"
 android:layout_width = "wrap_content"
 android:layout_height = "wrap_content"
 android:text = "Clear" >
</Button>
</LinearLayout>
</LinearLayout>
class MainActivity extends Activity implements
OnClickListener {

  final String LOG_TAG = "myLogs" ;   Button btnAdd,
btnRead, btnClear;   EditText etName, etEmail;
DBHelper dbHelper;
  / ** Called when the activity is first created. * /
  @Override
  Public void onCreate ( Bundle savedInstanceState ) {
    super .onCreate ( savedInstanceState ) ;
    setContentView ( R.layout.main ) ;     btnAdd =

( Button ) findViewById ( R.id.btnAdd ) ;
    btnAdd.setOnClickListener ( this ) ;     btnRead =
```

```
( Button ) findViewById ( R.id.btnRead ) ;
    btnRead.setOnClickListener ( this ) ;     btnClear =

( Button ) findViewById ( R.id.btnClear ) ;
    btnClear.setOnClickListener ( this ) ;     etName =

( EditText ) findViewById ( R. id.etName ) ;
    etEmail = ( EditText ) findViewById ( R.id.etEmail )
;

    // створюємо об'єкт для створення і управління
версіями БД
    dbHelper = new DBHelper ( this ) ;
  }


  @Override
  public void onClick ( View v ) {

    // створюємо об'єкт для даних
    ContentValues cv = new ContentValues () ;

    // отримуємо дані з полів введення
    String name = etName.getText () .toString () ;
    String email = etEmail.getText () .toString () ;

    // підключаємося до БД
    SQLiteDatabase db = dbHelper.getWritableDatabase ()
;
```

```
    switch ( v.getId ()) {
    case R.id.btnAdd:
      Log.d ( LOG_TAG, "--- Insert in mytable: ---" ) ;
      // підготуємо дані для вставки у вигляді пар:
найменування стовпця - значення

      cv.put ( "name" , name ) ;
      cv.put ( "email" , email ) ;
      // вставляємо запис і отримуємо її ID
      long rowID = db.insert ( "mytable" , null, cv ) ;
      Log.d ( LOG_TAG, "row inserted, ID =" + rowID ) ;
      break ;
    case R.id.btnRead:
      Log.d ( LOG_TAG, "--- Rows in mytable: ---" ) ;
      // робимо запит всіх даних з таблиці mytable,
отримуємо Cursor
      Cursor c = db.query ( "mytable" , null, null,
null, null, null, null ) ;

      // ставимо позицію курсора на перший рядок вибірки
      // якщо в вибірці немає рядків, повернеться false
      if ( c.moveToFirst ()) {

        // визначаємо номери стовпців на ім'я в вибірці
        int idColIndex = c.getColumnIndex ( "id" ) ;
        int nameColIndex = c.getColumnIndex ( "name" ) ;
        int emailColIndex = c.getColumnIndex ( "email" )
;
```

```
        do {
          // отримуємо значення за номерами стовпців і
пишемо все в лог
          Log.d ( LOG_TAG,
              "ID =" + c.getInt ( idColIndex ) +
              ", name =" + c. getString ( nameColIndex )
+
              ", email =" + c.getString ( emailColIndex
)) ;
          // перехід на наступний рядок
          // а якщо наступної немає (поточна - остання),
то false - виходимо з циклу
        } while ( c.moveToNext ( )) ;
      } else
        Log.d ( LOG_TAG, "0 rows" ) ;
      c.close () ;
      break ;
    case R.id.btnClear:
      Log.d ( LOG_TAG, "--- Clear mytable: ---" ) ;
      // видаляємо все записи
      int clearCount = db.delete ( "mytable" , null,
null ) ;
      Log.d ( LOG_TAG, "deleted rows count =" +
clearCount ) ;
      break ;
    }
    // закриваємо підключення до БД
    dbHelper.close () ;
  }
```

```
  class DBHelper extends SQLiteOpenHelper {

    public DBHelper ( Context context ) {
      // конструктор суперкласу
      super ( context, "myDB" , null, 1 ) ;
    }

    @Override
    public void onCreate ( SQLiteDatabase db ) {
      Log.d ( LOG_TAG, "--- onCreate database ---" ) ;
      // створюємо таблицю з полями
      db.execSQL ( "create table mytable ("
          + "id integer primary key autoincrement,"
          + "name text,"
          + "email text" + " ); " ) ;
    }

    @Override
    public void onUpgrade ( SQLiteDatabase db, int
oldVersion, int newVersion ) {     }   } }

private DatabaseHelper mDatabaseHelper;
private SQLiteDatabase mSqLiteDatabase;

public void onCreate (Bundle savedInstanceState) {
    super.onCreate (savedInstanceState);
```

```
    setContentView (R.layout.activity_main);

    mDatabaseHelper = new DatabaseHelper (this,
"mydatabase.db", null, 1);

    mSqLiteDatabase =
mDatabaseHelper.getWritableDatabase ();

    ContentValues values = new ContentValues ();
    // Задайте значення кожному за шпальти
    values.put (DatabaseHelper.CAT_NAME_COLUMN,
"Рижик");
    values.put (DatabaseHelper.PHONE_COLUMN,
"4954553443");
    values.put (DatabaseHelper.AGE_COLUMN, "5");
    // Вставляємо дані в таблицю
    mSqLiteDatabase.insert ( "cats", null, values);
}

public void onClick (View view) {
    Cursor cursor = mSqLiteDatabase.query ( "cats", new
String [] {DatabaseHelper.CAT_NAME_COLUMN,
                    DatabaseHelper.PHONE_COLUMN,
DatabaseHelper.AGE_COLUMN},
            null, null,
            null, null, null);

    cursor.moveToFirst ();
```

```
    String catName = cursor.getString
(cursor.getColumnIndex
(DatabaseHelper.CAT_NAME_COLUMN));
    int phoneNumber = cursor.getInt
(cursor.getColumnIndex (DatabaseHelper.PHONE_COLUMN));
    int age = cursor.getInt (cursor.getColumnIndex
(DatabaseHelper.AGE_COLUMN));

    TextView infoTextView = (TextView) findViewById
(R.id.textView);
    infoTextView.setText ( "Кот" + catName + "має
телефон" + phoneNumber + "і йому" +
            age + "років");

    // Не забуваємо закривати курсор
    cursor.close ();
}
```