Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи

магістр

(освітньо-кваліфікаційний рівень)

на тему: Розробка методу та програмних засобів автоматизованого ведення власних фінансів під Android систему

Виконав: студент II курсу
групи 1ПІ-18 м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Волошина А. В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Войтко В.В.
(прізвище та ініціали)

ВНТУ – 2019

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо-кваліфікаційний рівень – магістр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"____"_____2019 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Волошиній Анні Валеріївні

1. Тема роботи – розробка методу та програмних засобів автоматизованого ведення власних фінансів під Android систему.

Керівник роботи: Войтко Вікторія Володимирівна, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "__"_____2019 року №___

2. Строк подання студентом роботи

_____

3. Вихідні дані до роботи :

Мова програмування: Java

Технологія розробки: Android Studio

Браузери (або ОС): Android

4. Зміст розрахунково-пояснювальної записки: вступ; аналіз стану питання та постановка задачі; розробка модулів мобільного додатку; програмна реалізація мобільного додатку; тестування розробленого програмного продукту; економічна частина; висновки; перелік посилань; додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): моделі системи; інтерфейс; структура баз даних; ER-модель; алгоритм роботи авторизації системи; архітектура системи; приклад вигляду розробленої програми.

6. Консультанти розділів магістерської кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видано	завдання прийнято
1–4	к.т.н. Войтко В. В., доцент кафедри ПЗ		
5	к.е.н. Бальзан М. В., доцент кафедри ЕПВМ		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Техніко-економічне обґрунтування доцільності розробки мобільного додатку для ведення власних фінансів	07.10.2019 – 27.10-2019	Вик.
2	Розробка модулів автоматизованого ведення власних фінансів під Android	28.10.2019 – 8.11.2019	Вик.
3	Програмна реалізація мобільного додатку	9.11.2019 – 20.11.2019	Вик.
4	Тестування роботи мобільного додатку	21.11.2019 – 3.12.2019	Вик.
5	Економічне обґрунтування розробки програмного продукту	4.12.2019 – 7.12.2019	Вик.

Студент _____ **Волошина А. В.**

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Войтко В. В.**

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі проведено детальний аналіз процесу ведення власних фінансів в середовищі автоматизованої системи мобільного пристрою.

Запропоновано метод автоматизованого ведення власних фінансів для коректної роботи мобільного додатку. Розроблено модель автоматизованої системи ведення власного бюджету, яка відрізняється від існуючих використанням технологій моніторингу фінансового стану, орієнтованих на нові алгоритми обліку фінансів, що забезпечує підвищення ефективності процесів контролю і управління особистим бюджетом. Запропоновано нову організацію бази даних, яка базується на конвеєрному принципі обробки даних, що підвищує продуктивність програмного продукту.

Розроблений додаток автоматизованого ведення власних фінансів дозволяє моделювати відслідковувати надходження та витрати з мобільного додатку. Програма розроблена мовою Java, характеризується зручністю та зрозумілістю інтерфейсу, швидкістю та точністю опрацювання даних, що забезпечує всі вимоги користувача щодо ведення власних фінансів за допомогою свого мобільного пристрою.

ABSTRACT

In the master's qualification work, a detailed analysis of the process of managing one's own finances in the environment of an automated system of a mobile device was carried out.

The method of automated management of own finances for the correct operation of the mobile application is proposed. A model of an automated system for managing one's own budget has been developed, which differs from the existing ones using the technologies of financial status monitoring, focused on new algorithms for accounting of finances, which ensures the efficiency of the processes of control and management of the personal budget. A new database organization is proposed, based on the pipelined data processing principle, which increases the productivity of the software product.

The developed application of automated management of own finances allows to simulate tracking of receipts and expenses from the mobile application. Designed in Java, it features a user-friendly interface, speed and accuracy of data processing that meets all the user's requirements for managing their finances with their mobile device.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. На сьогоднішній день одним з основних питань людей постають гроші, скільки вони їх отримують та куди витрачають. Особисті фінанси для багатьох є болючим питанням. Якщо ми не володіємо мільйонами, то досить часто відчуваємо нестачу фінансів, і це не кажучи про той великий прошарок суспільства, яий взагалі живе в стані «від зарплати до зарплати».

На ринку техніки з'являється все більше портативних пристроїв. Для сучасної людини важливо постійно мати доступ до Інтернету, електронної пошти та соціальних мереж. Через те, що користувачі не завжди можуть мати доступ до стаціонарного комп'ютера чи ноутбука, все частіше на перший місце виходять мобільні телефони та планшети. Розробка мобільних додатків зараз набула великої популярності серед інших послуг цієї сфері. Кількість користувачів мобільних пристроїв на різних платформах зростає з кожним днем. Сьогодні користувачеві потрібний великий асортимент програмного забезпечення для роботи та особистого користування. Тому розробка мобільних додатків є актуальним питанням сьогодення.

Багато людей стикаються з ситуацією, коли їхній дохід тривалий час постійно збільшується, але заощадити не можуть. Сімейний бюджет також незмінний, а деякі навіть мають борги [1].

Причиною такої ситуації є не криза в країні, не зростання цін на споживчі товари, не коливання валютного ринку, як більшість людей часто виправдовують своє нестабільне фінансове становище. Всі причини цих ситуацій насправді криються в неможливості проаналізувати свій бюджет.

Враховуючи ці фактори, можна зробити висновок, що розробка мобільного додатку для ведення власних фінансів є актуальною.

Мета та завдання дослідження. Метою роботи є розширення можливостей моніторингу та аналізу власного бюджету за рахунок використання розробленої автоматизованої системи ведення власних фінансів під Android

систему, що дозволяє підвищити економічну самодисципліну та сприяє заощадженню коштів.

Основними **задачами** дослідження є:

- аналіз і порівняння існуючих аналогів та особливостей реалізації мобільних додатків;
- розробка інтуїтивно зрозумілого для користувачів інтерфейсу;
- удосконалення методу і моделей автоматизованої системи ведення фінансів;
- створення мобільного додатку;
- тестування створеного програмного продукту.

Об'єкт дослідження – процес ведення власних фінансів в середовищі автоматизованої системи мобільного пристрою.

Предметом дослідження є засоби реалізації мобільних додатків, призначених для ведення власних фінансів.

Методи дослідження:

- теорія баз даних та методи роботи з базою даних SQLite для створення і оновлення бази даних;
- методи теорії алгоритмів для розробки алгоритмів роботи програми;
- методи проектування програмного продукту для розробки мобільного додатку автоматизованого ведення власних фінансів;
- комп'ютерне моделювання для аналізу і перевірки достовірності отриманих теоретичних результатів.

Наукова новизна одержаних результатів:

1. Подальшого розвитку набули моделі автоматизованої системи ведення власного бюджету, які відрізняються від існуючих використанням технологій моніторингу фінансового стану, орієнтованих на нові алгоритми обліку фінансів, що забезпечує підвищення ефективності процесів контролю і управління особистим бюджетом.

2. Запропоновано нову організацію бази даних, яка, на відміну від

існуючих, базується на конвеєрному принципі обробки даних, що підвищує продуктивність програмного продукту.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що розроблена система автоматизованого ведення власних фінансів орієнтована на спрощення процесу ведення особистого бюджету з використанням мобільних пристроїв під платформу Android. Додаток призначений для внесення доходів та витрат, дозволяє керувати витратами, планувати покупки, що допомагає користувачеві слідкувати за станом своїх фінансів.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалася відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Особистий внесок здобувача. В публікації [2] автором розроблено алгоритм авторизації користувачів і системі атоматизованого ведення власних фінансів. В публікації [3] автором розроблено структуру веб-ресурсу. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто.

Достовірність теоретичних положень підтверджена результатами тестування розробленого мобільного додатку.

Апробація матеріалів магістерської кваліфікаційної роботи. Результати роботи доповідалися на двох науково-технічних конференціях: на Міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ – 2019» та на XLVII Міжнародній науково-технічній конференції ВНТУ – 2018.

Публікації. Результати роботи опубліковані в 2 наукових публікаціях: «Мобільний додаток для атоматизованого ведення власних фінансів» [2] та «Розробка засобів реалізації web-сайту навчального закладу» [3].

Графічні зображення, використані в роботі, подавалися на Міжнародний конкурс з комп'ютерної графіки «Творчість без меж» (Болгарія, м. Хасково) 2019р., де отримано 2 місце (срібну медаль).

Структура та обсяг роботи. Магістерська кваліфікаційна робота складається зі вступу, п’яти розділів, висновків, списку літератури, що містить 25 найменувань, 3 додатків. Робота містить 17 ілюстрацій, 5 таблиць.

Робота складається з 5 розділів. У першому розділі проаналізовано сучасний стан питання теми розробки, проведено порівняння аналогів і обґрунтовано вибір засобів реалізації мобільного додатку, сформульовано задачі дослідження.

Другий розділ містить аналіз структури, принципів та особливостей розробки мобільних додатків, особливостей розробки бази даних, а також здійснено розробку методу і засобів реалізації автоматизованої системи.

У третьому розділі розроблено інтерфейс, базу даних та проведена програмна реалізація мобільного додатку.

Четвертий розділ показує результати тестування розробленого мобільного додатку.

У п’ятому розділі розраховуються витрати на розробку програмного забезпечення, експлуатаційні витрати, обсяг роботи, пов'язаної з використанням програмного забезпечення, та економічний ефект від впровадження нового програмного продукту.

У висновках наведені основні результати дослідження.

У додатках міститься технічне завдання, лістинг коду основних частин програми та ілюстративний матеріал до захисту магістерської кваліфікаційної роботи.

1 АНАЛІЗ СТАНУ ПИТАННЯ ВЕДЕННЯ ВЛАСНИХ ФІНАНСІВ ТА ПОСТАНОВКА ЗАДАЧ ДОСЛІДЖЕННЯ

1.1 Аналіз особливостей мобільних додатків ведення фінансів

Стабільні особисті фінанси потребують хоча б мінімуму часу на планування і аналіз. Для цього можна завести щоденник або завантажити на смартфон один з додатків для обліку витрат. Але так як ми живемо в 21 столітті, де щоденники відходять на другий план, оскільки частіше за все можна забути вписати якісь витрати, то простіше скачати додаток на свій мобільний пристрій та користуватись ним [4].

Мобільний застосунок або додаток (англ. «Mobile app») [5] — програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях. Багато мобільних застосунків встановлені на самому пристрої або можуть бути завантажені на нього з онлайн магазинів мобільних застосунків, таких як App Store, Google Play, Windows Phone Store та інших, безкоштовно або за плату.

Спочатку мобільні додатки пропонувались як інструменти для контролю та управління загальними інформаційними потоками, включаючи електронну пошту, календар, контакти, інформацію про фондову біржу та погоду. Однак попит та доступність інструментів для розробників призвели до швидкого розповсюдження програм на інші категорії настільних електронних пристроїв.

За допомогою мобільних додатків ви можете отримати чіткі переваги:

- Повна взаємодія з користувачем. На відміну від мобільних версій веб-сайтів, у випадку з мобільним додатком існує можливість надсилання push-повідомлень. За допомогою мобільного додатку просто надсилаєте повідомлення всім активним користувачам, обмежуючи кількість символів, як це не відбувається у SMS-кампаніях;
- Локальні сповіщення працюють приблизно так само, встановлені лише на самому пристрої. Якщо користувач виконує дію в додатку, який розраховує відповісти протягом певного часу, програма нагадує йому;

- Він також пропонує більше відгуків - користувачі можуть залишати повідомлення, свої відгуки в магазинах додатків, особисто через додаток, соціальні мережі - і вони будуть автоматично передані на всі ресурси;
- Найкращий інтерфейс. Усі кнопки управління, текстові поля, посилання повинні легко клацати на мобільному пристрої, а не курсором миші. Екрани пристроїв відрізняються розміром і пікселями. Екран телефону може мати роздільну здатність майже як ноутбук, але фізичний розмір все ще невеликий. Мобільні операційні системи для цих ситуацій мають власну спеціально навчену логіку для розрізнення пристрою, з яким вони працюють, для боротьби з наслідками попадання пальців у маленькі кнопки. Але в результаті ця логіка працює по-різному на платформах і пристроях;
- Якість інтерфейсу відображається і в навігації. Кожна мобільна операційна система має свою логіку перемикання між екранами робочого столу в додатках;
- Високий рівень персоналізації. Мобільні телефони багато знають про свого власника, і використання цієї інформації для покращення обслуговування є однією з ключових причин швидкого зростання мобільних додатків. Мобільний додаток завжди має можливість зберігати всі дані користувача та змінювати інтерфейс відповідно до його потреб. Якщо користувач вводить будь-яку інформацію (наприклад, свою домашню адресу в додатку для доставки), їм не потрібно буде повторно вводити адресу. Навіть на різних пристроях, коли синхронізація включена, користувач побачить програми із заповненими персональними даними;
- Логіка мобільного сайту завжди простіша, вона може не враховувати безліч даних, які можуть надати мобільні пристрої;
- Працювати в автономному режимі. Інтернет є скрізь, але не завжди. Навіть при мобільному підключенні до Інтернету його якість в середньому гірша, ніж у домашніх та офісних лініях Інтернету. Якщо мета - не дозволити користувачам втрачати зв’язок із продуктом, як тільки розривається Інтернет-з'єднання, розробка мобільного додатка - єдине можливе рішення;

Однією з переваг управління бюджету за допомогою мобільного додатку – чіткість усіх витрат та можливість їх аналізу. Деякі недоліки включають витрати на час і взагалі не найцікавіші хвилини життя. Однак у будь-якому випадку вам доведеться вибрати: або погоджуватися зі своїми непланови витратами, або проводити звичайну, але необхідну роботу з планування.

Після аналізу своїх фінансових транзакцій, зможете по-справжньому керувати своїм бюджетом: відкласти, збільшити бюджет, знайти додаткове джерело доходу або вкласти гроші, створити додаткові статті особистого бюджету для великих придбань, навчання чи виходу на пенсію. Після дослідження своїх витрат можете досягнути певної самодисципліни, яка дозволяє економити і рятувати себе від непродуманих, спонтанних покупок.

1.2 Аналіз операційних систем для розробки мобільного додатку

У сучасному світі існує величезна різноманітність мобільних пристроїв. Як у будь-якого комп'ютерного пристрою, у мобільних є операційні системи (ОС) [6]. Найвідоміші з них: Android, IOS, ОС Windows, Blackberry OS.

Android - це операційна система та мобільна платформа, створена Google на основі ядра Linux. За підтримки Альянсу відкритих слухавок (OHA).

Переваги ОС - Android:

- різноманітні програми та ігри на Android;
- швидка інтеграція з сервісами Google
- наявність файлової системи;
- абсолютна незалежність від апаратних засобів мобільного пристрою;
- Android - екосистема з відкритим кодом;
- багатозадачність - це означає, що безліч додатків працює безперебійно;
- простота установки додатків з різних ресурсів;
- широкі можливості індивідуалізації;
- відсутність обмежень щодо вибору мобільного оператора;

- підтримка флеш-плеєра;
- зручне оновлення через Інтернет;
- можливість заміни / видалення програм за замовчуванням;
- спрощено виробництво додатків, ігор, усіх видів плагінів, оновлень.

Недоліки ОС - Android:

- пристрій на базі Android має заряджати / заряджати досить часто;
- є деякі проблеми із сумісністю нових версій операційної системи із припиненими, але застарілими пристроями;
- користувачі, які знаходять комфорт в першу чергу, можуть бути незадоволені кількістю налаштувань.

iOS - власна мобільна операційна система Apple. Спочатку розроблений для iPhone, він також став операційною системою для iPod Touch, iPad та Apple TV. Apple не дозволяє ОС працювати на мобільних телефонах інших компаній.

Переваги IOS:

- велика зручність використання, зручний інтерфейс;
- шикарний дизайн;
- акцент на надійності та якості ОС;
- оптимізація допомагає збільшити навантаження гаджета при повному навантаженні, заощаджуючи енергію та ресурси до 10 годин;
- відсутність гальм, глюків, як у додатках, так і на робочому столі пристроїв;
- величезний вибір додатків;
- У додатках, на відміну від Google Play, реклама є рідкісною;
- Оновлення доступні відразу після виходу нової ОС для всіх пристроїв одночасно;
- розробники щоразу оголошують свої додатки для iOS;
- можливість інтегрувати останні оновлення мобільних ОС;
- великий плюс - довготривала підтримка старих пристроїв;
- Сімейний доступ для покупок у App Store;

- багатозадачність.

Недоліки ОС - IOS:

- закрита, досить консервативна система - все програмне забезпечення, за невеликими винятками, практично платне (і ця опція характерна для всіх пристроїв Apple - від iPhone до iPad);
- іноді користувачеві доводиться переплачувати за додатки;
- пряма копія або передача файлів з'явилася лише нещодавно (лише iTunes міг рухатися);
- в iOS все підключено до Інтернету (підключення немає, втрачено багато функцій);
- працює лише на пристроях Apple, що значно зменшує поширення iOS по всьому світу. Ipads, iPhone, Ipods, Macbooks, Macs та IWatch - найпоширеніші у заможних країнах - у США, Європі.

Windows Phone - це операційна система для мобільних пристроїв із основним набором програм, таких як Windows Marketplace для мобільних пристроїв. Телефон Windows може працювати на багатьох пристроях, включаючи кишенькові ПК, смартфони, комунікатори.

Windows – найпоширеніша ОС ПК доки що у світі, і мобільні пристрої, що працюють на ОС Windows просто синхронізуються з комп’ютером, ноутбуком, планшетом;

Переваги для Windows:

- немає проблем з пам’яттю;
- достатня кількість програм за замовчуванням;
- працює безперебійно - гладкість інтерфейсу, швидкість запуску програм, перемикання між відкритими вікнами тощо;
- досить зручне оформлення інтерфейсу;
- прості та зрозумілі налаштування;
- підтримка великої кількості пристроїв;
- немає проблем з пам’яттю;

Недоліки Windows:

- мало програм або вони не функціонують повністю;
- Деякі старі користувачі підтримки смартфонів були покинуті, наприклад, Skype, який вже є в автономному режимі.

Проаналізувавши операційні системи для мобільних телефонів, можна зробити висновок, що найкращий вибір для вирішення питання – це вибір Android системи для розробки програмного продукту, оскільки більшість користувачів користуються мобільними телефонами на базі цієї операційної системи [7].

1.3 Порівняльний аналіз аналогів мобільних додатків ведення фінансів

Незважаючи на те, що вести облік власного бюджету досить важливо, інформації про це можна знайти досить багато, існує відносно невелика кількість програм, які можуть бути використані в цілях автоматизованого ведення власних фінансів. У кожного з наявних аналогів є певні переваги та недоліки.

Основним недоліком таких програм є те, що для повного функціоналу потрібно придбати або щомісячно вносити плату за користування додатком. Також не зрозумілий та незручний інтерфейс таких додатків.

Одним з таких аналогів є «Monefy: зручний облік витрат» [8] – простий інструмент для обліку особистих фінансів. Додаток дозволяє вносити доходи, витрати і перекази між рахунками в різних валютах, створювати бюджет та аналітичні звіти за категоріями. Але Monefy не має можливості прикріплювати до транзакцій теги, фото і геомітки, дублювати і планувати їх, встановлювати нагадування і вести облік боргів. Рівень безпеки програми відповідає середнім показникам. При безкоштовному використанні додаток накладає обмеження на створення нових категорій і синхронізацію. На рисунку 1.1 основну сторінку додатку Monefy.

Рисунок 1.1 – Основна сторінка додатку Monefy: зручний облік витрат

Ще одним аналогом є програмний продукт «ДзенМані» [9] (рис. 1.2) – облік фінансів для тих, хто хоче контролювати витрати без зайвих зусиль. Завдяки синхронізації з банками вам не доведеться вводити витрати вручну. Надає розширені можливості по управлінню категоріями і рахунками. Додаток дозволяє записувати витрати, доходи і перекази між рахунками в різних валютах, враховувати борги, встановлювати бюджети і розраховувати бюджет на день виходячи з доступних коштів на місяць.

«Дзен-мані» підтримує автоматичний імпорт транзакцій з онлайн-банків і розпізнавання СМС, дозволяє планувати транзакції і розбивати їх на декілька категорій. Однак відсутній калькулятор, пошук транзакцій і нагадування, перенесення залишків і імпорт з файлу, немає можливості додати геометки, теги і фото до транзакцій.

Рівень безпеки програми відповідає високим показникам. Мобільний додаток регулярно отримує оновлення та має веб-версію. При безкоштовному використанні додаток накладає обмеження на розділи «Категорії» та «Звіт за категоріями».

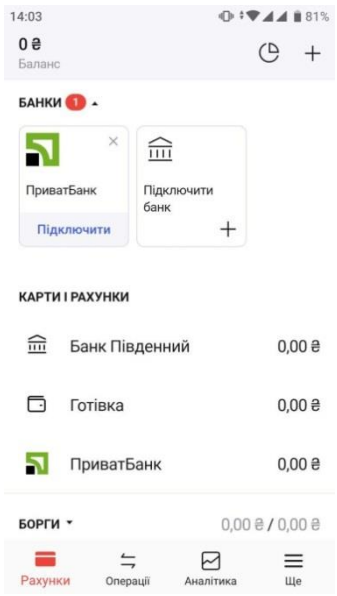


Рисунок 1.2 – Вікно програми ДзенМані

Проаналізувавши аналоги, визначимо їхні можливості та недоліки, які враховуємо при створенні власного програмного додатку з назвою «MyFin» (табл. 1.1).

Таблиця 1.1 – Порівняльні характеристики програмних продуктів

Критерій	Monefy: зручний облік витрат	ДзенМані	MyFin
Зручний інтерфейс	+	-	+
Не вимагає доступ до карток чи паролів	+	-	+
Високий рівень безпеки	-	+	+
Повний безкоштовний функціонал	-	-	+

Таблиця порівняльних характеристик показала, що розробка програмного продукту є доцільною. В результаті отримаємо продукт, який покриває недоліки

існуючих рішень і забезпечує більшу ефективність управління власними фінансами.

1.4 Аналіз середовищ розробки мобільного додатку

Для розробки мобільних додатків олзглянемо такі середовища розробки:

- Android Studio;
- IntelliJ IDEA;
- Eclipse;
- Intel XDK.

Android Studio [10] — інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O менеджером по продукції корпорації Google — Еллі Паверс.

Android Studio замінив плагін Eclipse ADT. Середовище побудовано на вихідному коді продукту IntelliJ IDEA Community Edition, розробленого JetBrains. Android Studio розробляється як модель розробки з відкритим кодом та має ліцензію на основі Apache 2.0.

Середовище розробки пристосоване для виконання типових завдань, які вирішуються під час розробки програм для платформи Android. Включаючи інструменти для спрощення тестування програмного забезпечення на сумісність з різними версіями платформи та інструменти для проектування додатків, що працюють на пристроях з різними екранами роздільної здатності (планшети, смартфони, ноутбуки, годинник, окуляри тощо) .

Надаються наступні функції:

- Макети: редактор WYSIWYG - кодування в реальному часі - візуалізація програми в режимі реального часу.
- Консоль розробника: поради щодо оптимізації, допомога в перекладі, відстеження напрямків, кампанії та акції - показники Google Analytics.
- Резерви бета-версії та покрокові випуски.
- Базування на Gradle.

- Рефакторинг, орієнтований на Android, та швидкі виправлення.
- Службові утиліти для продуктивності, зручності використання, сумісності версій та інші проблеми.
- Використовуйте можливості та підписи ProGuard для додатків.
- Шаблони для створення загальних дизайнів та компонентів Android.
- Різноманітний редактор макетів, що дозволяє користувачам перетягувати компоненти інтерфейсу користувача, як варіант, одночасно переглядати макети в різних конфігураціях екрана.

IntelliJ IDEA [11] - це комерційне інтегроване середовище розробки для різних мов програмування (Java, Python, Scala, PHP тощо) від JetBrains. Система випускається у вигляді вбудованого функціоналу безкоштовної версії "Community Edition" та повнофункціональної комерційної версії "Ultimate Edition", на яку активні розробники відкритих проектів мають можливість отримати безкоштовну ліцензію.

Версія спільноти IntelliJ IDEA підтримує додаткові інструменти для тестування TestNG та JUnit, CVS, Subversion, Mercurial і Git систем управління, Maven, Ant, Gradle, Java, Scala, Clojure, Groovy та Dart. Підтримується розробка додатків для мобільної платформи Android. Він включає модуль візуального дизайну для GUI дизайнера Swing, інтерфейс редактора XML, редактор регулярних виразів, систему перевірки коду, систему управління завданнями та додатки для імпорту та експорту проектів з Eclipse. Доступні інтеграції із системами відстеження помилок JIRA, Trac, Redmine, Pivotal Tracker, GitHub, YouTrack, Lighthouse.

Комерційна версія "Ultimate Edition" характеризується наявністю підтримки додаткових мов програмування (наприклад, PHP, Ruby, Python, JavaScript, CoffeeScript, HTML, CSS, SQL), підтримкою технологій Java EE, діаграм UML, коду охоплення, можливість роботи з рамками (Rails, Grails, Google Web Toolkit, Spring, Play Framework та Hibernate), за допомогою інструментів інтеграції від Perforce, Microsoft Team Foundation Server та Rational Zlearzase.

Eclipse [12] - це безкоштовне модульне інтегроване середовище розробки програмного забезпечення. Фонд Eclipse розроблений та підтримується та включає такі проекти, як платформа Eclipse, набір інструментів програмістів Java, системи управління версіями, конструктори GUI тощо.

Eclipse - це, перш за все, повноцінна Java IDE, орієнтована на груповий розвиток з інструментами для систем управління версіями (підтримка CVS включена в доставку Eclipse, активно розробляються декілька модулів SVN, доступна підтримка VSS тощо). Eclipse - це корпоративний стандарт для розробки програмного забезпечення Java у багатьох організаціях.

Eclipse базується на Java, тому це незалежний від платформи продукт, за винятком SWI GUI, який розробляється окремо для більшості поширених платформ. Бібліотека SWT використовує інструменти графічної платформи (ОС) для забезпечення швидкості та звичного вигляду інтерфейсу користувача.

Intel XDK [13] - програмне забезпечення корпорації Intel, яке забезпечує роботу над усім життєвим циклом розробки кросбраузерності мобільних додатків з використанням веб-технології (зокрема HTML5 і Cordova). Основною перевагою інструментарію Intel XDK є можливість розробки на CSS і JavaScript, після чого компілювати проект в файли для IOS, Android і Windows Phone - .ipa, .apk і .appx відповідно. Після чого дані файли можна завантажувати в магазини додатків.

Проаналізувавши платформи та їх можливості, було визначено, що Android Studio являється найсучаснішою платформою для створення мобільних додатків, має багато функцій, спрощує тестування, а також прийшов на заміну Eclipse та побудований на вихідному коді IntelliJ IDEA. Платформа перейняла найкращі характеристики, тому для розробки програмного продукту була обрана Android Studio.

1.5 Постановка задач роботи

На основі проведеного аналізу стану питання додаткі для автоматизованого ведення власних фінансів у магістерській кваліфікаційній роботі потрібно виконати такі задачі:

- розробити метод та засоби ведення фінансів з використанням мобільних технологій;
- проаналізувати особливості та розробити структуру бази даних для ведення власних фінансів у мобільному додатку;
- розробити алгоритм функціонування мобільного додатку;
- розробити інтерфейс автоматизованої системи;
- здійснити програмну реалізацію мобільного додатку;
- провести тестування створеного програмного продукту.

1.6 Висновки

У першому розділі магістерської кваліфікаційної роботи проведено аналіз стану питання мобільних додатків для ведення власних фінансів, також порівняно аналоги програмного продукту. Було проаналізовано операційні системи для розробки мобільних додатків, обрано платформу Android для розробки, оскільки вона зручна та близька більшості користувачам.

Мобільний додаток призначений для ведення власних фінансів за допомогою мобільного пристрою з системою Android. Проаналізовані аналоги дозволили сформувати вимоги до мобільного додатку, який орієнтований на зручність роботи користувача.

В якості ОС було обрано Android, середовищем програмування – інтегроване середовище розробки Android Studio.

2 РОЗРОБКА МЕТОДУ ТА МОДЕЛЕЙ СИСТЕМИ АВТОМАТИЗОВАНОГО ВЕДЕННЯ ВЛАСНИХ ФІНАНСІВ

2.1 Обгрунтування вхідних та вихідних даних системи автоматизованого ведення фінансів

Щоб додаток був працездатним та мав взаємодію з користувачем, потрібно приділити увагу взаємовідношенням дій. При певній дії користувача повинна відбуватися відповідна дія в додатку.

Для коректної роботи, а також для захисту персональних даних про витрати і надходження користувача необхідний захищений вхід в додаток. Для цього будуть потрібні вхідні дані, і від коректності їх введення будуть залежати вихідні.

Вхідні дані: логін і пароль користувача.

Вихідні дані: основне вікно додатку або повідомлення про невірне введення даних.

Для того, щоб додавати витрати чи надходження, також потрібні вхідні дані, на виході яких можна буде отримати результати, відображені в додатку.

Вхідні дані: категорія, сума, дата, нотатка (за бажанням).

Вихідні дані: відображення даних у вікні додатку.

2.2 Обгрунтування вибору структури для розробки мобільних додатків

З ростом кількості мобільних пристроїв збільшується і потреба в розробці додатків. Щодня створюються сотні мобільних додатків, які роблять роботу з мобільним пристроєм більш комфортною. Існує кілька типів додатків, які використовують розробники: нативні, веб-додатки та гібридні [14].

Нативні додатки називають нативними тому, що вони написані рідною для певної платформи мовою програмування. Для Android цією мовою є Java [15], тоді як для iOS - objective-C або Swift. Вони знаходяться на самому пристрої,

доступ до яких можна отримати, натиснувши на іконку. Вони встановлюються через магазин додатків (Play Market на Android, App Store на iOS і ін.).

Розроблені спеціально для конкретної платформи і можуть використовувати всі можливості пристрою - камеру, GPS-датчик, акселерометр, компас, список контактів і все інше. Також вони можуть розпізнавати стандартні жести, встановлені операційною системою або зовсім нові жести, які використовуються в конкретному додатку.

В силу того, що нативні додатки оптимізовані під конкретну ОС, вони органічно вписуються в будь-який смартфон, відрізняючись високою швидкістю роботи і продуктивністю.

Серед плюсів можна виділити такі:

- швидкість роботи і продуктивність;
- високий ступінь безпеки;
- розширений інтерфейс;
- відносно висока вартість розробки;
- максимально можлива функціональність;
- здатність працювати без Інтернету;
- зручність для користувача

Мінуси нативних додатків:

- охоплення платформ;
- тривалі терміни розробки;
- необхідність випускати оновлення в косметичних цілях.

Веб-додатки відрізняються кроссплатформенністю, тобто здатні функціонувати, незалежно від платформи девайса. Козирем в їх рукаві виступає і те, що вони не використовують його програмне забезпечення. А через те, що є мобільною версією сайту з розширеним інтерактивом, веб-додатки не відбирають дорогоцінний місце в пам'яті смартфона.

Веб-додатки стали широко популярні в той час, коли почав розвиватися HTML5 і люди усвідомили, що можуть отримати доступ до безлічі функцій

нативних додатків, просто зайшовши на веб-сайт через звичайний браузер. На сьогоднішній день складно сказати, де саме розташовується чітка межа між веб-додатками і звичайними веб-сторінками, оскільки функціонал HTML5 зростає з кожним днем і все більше і більше сайтів його використовують.

До плюсів можна віднести:

- повне охоплення платформ;
- простий і швидкий процес розробки;
- кількість компетентних розробників;
- відсутність необхідності завантаження з магазину додатків.

З мінусів можна виділити :

- обов'язкове підключення до Інтернету;
- убогий інтерфейс програми;
- неможливість відправити push-повідомлення;
- продуктивність і швидкість роботи;
- незадовільний рівень безпеки.

Гібридні додатки являють собою поєднання веб і нативних додатків. Особливо, мається на увазі їх кроссплатформенность і доступ до функціоналу смартфона. Такі додатки можуть бути завантажені виключно з маркетів на кшталт Google Play і App Store. Разом з тим вони мають у своєму розпорядженні опцією автономного оновлення інформації, а для їх роботи необхідно інтернет-підключення. Без наявності останнього веб-функції просто не працюють. Однак, на відміну від нативних, вартість створення гібридних на порядок нижче, а його швидкість – вище. Спорідненість гібридних додатків з веб-додатками, в свою чергу, дає плоди у вигляді того, що в них можна легко і оперативно вносити корективи.

Плюси гібридних додатків:

- вартість і швидкість розробки;
- кількість розробників;
- кроссплатформенность;

- опція автономного поновлення.

Мінуси гібридних додатків:

- некоректна робота при відсутності інтернет-з'єднання;
- середня швидкість роботи на тлі нативних;
- мінімалізм щодо візуальних елементів.

Проаналізувавши структури додатків, було прийняте рішення створити нативний додаток задля зручності у використанні.

2.3 Обгрунтування принципів розробки мобільних додатків для продуктивності додатку

Ефективні методи розробки додатків нерозривно пов'язані з нативним кодом, оптимізацією, дизайном, рефракторінгом. Кожна мова програмування є унікальною і копіює методи та класи. У парадигмі програмування кожна мова пов'язана з одним поняттям, принципом та абстракцією, що визначає фундаментальний стиль програмування. Рідними мовами для Android є мови розмітки XML, C ++ та C # [16].

Ефективні методи програмування під Android:

- програмування під Android – це Java, додатки якої транслюються в байт-код, що виконується віртуальною Java-машиною JVM;
- використання порожних елементів розмітки TextField з нульовою висотою і шириною з параметром «centerInParent» рівним «true» для вирівнювання елементів по центру екрану;
- використання елегантних способів доступу до даних, наприклад, values [SensorManager.DATA_X];
- використання розмітки RelativeLayout і властивість «fill_parent» для цієї розмітки;
- використання кількох робочих моніторів для програмування;
- уникнення створення купи об'єктів, використання існуючих об'єктів;

- зберігання даних в SQLite, якщо їх об'єм великий;
- використання кольорової розмітки для об'єктів. Встановлення кольору фону для деяких об'єктів. Це дозволяє виділити помилки та прискорити процес їх знаходження;
- використання методів onPause()/onResume() для збереження або закриття всього того, що цього вимагає;
- використання команди Source - Format для форматування XML-файлів, щоб привести їх у читабельний вигляд;
- використання плагіну XML Tools для Notepad++ для швидкого редагування XML-файлів і розуміння структури коду;
- використання Intents за допомогою окремого методу;
- використання шаблонів, які можна швидко використати для вставки в проект.

Основні принципи розробки продуктивних додатків під Android: економія апаратних ресурсів, протоколювання і аналіз ходу виконання додатку, ефективна праця з виділенням пам'яті, уникання зайвих об'єктів; для констант, класів необхідно використовувати модифікатор static final і не використовувати enum там, де модифікатор не вписується.

2.4 Обгрунтування використання бази даних для розробки автоматизованої системи

Використання баз даних є однією з особливостей більшості сучасних інформаційних систем. По суті, база даних - це та, навколо якої будується інформаційна система будь-якого програмного продукту.

База даних (БД) - це упорядкований набір логічно взаємопов’язаних даних, які спільно використовуються та призначені для задоволення інформаційних та технічних потреб користувачів, включаючи систему управління базами даних.

Основне завдання бази даних - забезпечити збереження значної кількості інформації та надання доступу до користувача чи програми. Таким чином, база даних складається з двох частин: збереженої інформації та її системи управління. Для забезпечення ефективності доступу записи організовуються як сукупність фактів (елемент даних).

Класифікація БД за моделлю даних:

- ієрархічні,
- мережеві,
- реляційні,
- об'єктні,
- об'єктно-орієнтовані,
- об'єктно-реляційні.

Класифікація БД за технологією фізичного зберігання:

- БД у вторинній пам'яті (традиційні);
- БД в оперативній пам'яті (in-memory databases);
- БД у третинній пам'яті (tertiary databases).

Класифікація БД за ступенем розподіленості:

- централізовані (зосереджені);
- розподілені.

У розробленій автоматизованій системі необхідно ефективно зберігати облікову інформацію. Щоб вирішити цю проблему в мобільному додатку, потрібно використовувати концепцію баз даних.

Реляційна модель орієнтована на організацію даних у вигляді двовимірних таблиць. Кожна реляційна таблиця є двовимірним масивом і має такі властивості:

- кожен елемент таблиці - один елемент даних;
- всі комірки стовпця таблиці однорідні, тобто всі елементи стовпця мають однаковий тип (числові, символьні тощо);
- кожен стовпець має унікальну назву;
- в таблиці немає однакових рядків;

- порядок рядків і стовпців може бути довільним.

Android використовує базу даних SQLite [17], яка є досить популярною, простою і швидкою реляційною базою даних. Особливістю SQLite є те, що він не використовує парадигму клієнт-сервер, тобто движок SQLite не є окремим процесом, з яким програма взаємодіє, а швидше надає бібліотеку, з якою програма компілюється, і двигун стає частиною програми. Таким чином, протокол Exchange використовує функціональні дзвінки (API) SQLite. Такий підхід скорочує накладні витрати, час реагування та спрощує програму.

SQLite є автономним засобом, тому він вимагає мінімальної підтримки з боку операційної системи або зовнішньої бібліотеки. Це робить SQLite застосованим у будь-яких умовах, особливо у вбудованих пристроях, таких як iPhone, телефони Android, ігрові консолі, портативні медіаплеєри тощо.

SQLite зберігає всю базу даних (включаючи визначення, таблиці, індекси та дані) в єдиному стандартному файлі на пристрої, на якому працює програма. Простота реалізації досягається блокуванням всього файлу, який зберігає базу даних до початку транзакції; Функції ACID досягаються створенням журнального файлу.

Кілька процесів можуть без проблем читати дані з однієї бази даних одночасно. Введення бази даних може бути здійснено лише в тому випадку, якщо наразі не надсилаються інші запити; в іншому випадку спроба запису не вдається, і код помилки повертається. Інша розробка полягає в автоматичному повторенні запису через визначений часовий інтервал.

SQLite використовує динамічні типи для таблиць. Це означає, що ви можете зберігати будь-яке значення в будь-якому стовпчику, незалежно від типу даних.

SQLite дозволяє з'єднанню однієї бази даних отримати доступ до декількох файлів бази даних одночасно. Це приносить багато приємних функцій, таких як об'єднання таблиць в різні бази даних або копіювання даних між базами даних в одну команду.

SQLite здатний створювати бази даних в пам'яті, з якими дуже швидко працювати.

Особливості SQLite:

- Транзакції є атомними, послідовними, ізольованими та довговічними (ACID) навіть після збоїв у системі та відключення електроенергії.
- Нульова конфігурація - не потрібно налаштування чи адміністрування.
- Повнофункціональна реалізація SQL з розширеними можливостями, такими як часткові індекси, індекси виразів, JSON, загальні вирази таблиці та функції вікон.
- Повна база даних зберігається в одному файлі міжплатформних дисків. Відмінно підходить для використання у форматі файлу додатків.
- Підтримує бази даних, що мають розмір терабайт, а також рядки та краплі розміром в гігабайт.
- Невеликий слід коду: менше 600 Кбіт повністю налаштовано або набагато менше з додатковими функціями, опущеними.
- Написано в ANSI-C. TCL прив'язки включені. Прив’язки для десятків інших мов доступні окремо.
- Добре коментований вихідний код зі 100% охопленням гілки.
- Доступний як єдиний файл вихідного коду ANSI-C, який легко компілювати, а тому легко додати до більшого проекту.
- Самодостатня: відсутність зовнішніх залежностей.
- Крос-платформа: Android, * BSD, iOS, Linux, Mac, Solaris, VxWorks та Windows (Win32, WinCE, WinRT) підтримуються поза коробкою.
- Джерела знаходяться у відкритому доступі.

2.5 Розробка методу автоматизованого ведення власних фінансів

Для коректної роботи додатку було розроблено метод, який складається з наступних кроків:

1. Створення аккаунту/авторизація.

На даному кроці користувачеві необхідно придумати свій логін та пароль, який потім буде надіслано до бази даних та збережено там. Після введення даних у поля відбувається перевірка на правильність заповнення полів вводу даних за допомогою конструкції if-else. Якщо поля заповнені неправильно, то з’являється помилка створення аккаунту, якщо все пройшло успішно – реєстрація завершена. По такому ж принципу працює і заповнення полів авторизації.

2. В разі правильного введення даних реєстрації сформувати повідомлення про успішну дію/перехід до основного вікна додатку.

Коли дані користувача введені коректно додаток переходить до основного вікна, де можна побачити пункти витрат, надходжень та балансу.

3. Створення бази витрат та надходжень власних фінансів.

На даному кроці користувач самостійно вводить інформацію, яка йому потрібна: призначення, витрати, надходження.

4. У разі додавання витрати активувати функцію addTransactionToDatabse.

Коли користувач натискає на «+» та додає нову витрату, вписує по черзі суму, призначення та обирає дату, після того, як натискає на кнопку «зберегти деталі» активується функція, яка додає витрати в основне вікно додатку, де користувач зможе їх переглядати.

5. У разі отримання надходження – активувати функцію addIncomeToDatabse.

Коли користувач натискає на «+» та додає нове надходження, вписує суму та обирає дату, після натиснення на кнопку «зберегти деталі» активується функція, яка додає надходження в основне вікно додатку, де користувач зможе їх переглядати.

6. Додати нову інформацію до існуючої.

При додаванні нової інформації вона зберігається в базі даних та висвітлюється у головному вікні додатку.

2.6 Розробка моделей системи автоматизованого ведення власних фінансів

У плані обробки взаємодії між користувальницьким інтерфейсом та його логікою Android дотримується архітектурної схеми «Model-View-ViewModel» (MVVM).

Model-View-ViewModel – це шаблон проектування, що застосовується під час проектування архітектури за стосунків. За допомогою MVVM декларативно визначає користувальницький інтерфейс (у цьому випадку за допомогою XML) і використовує розмітку прив'язки даних для асоціації його з іншими рівнями, що містять дані та команди користувача.

MVVM сприяє відокремленню розробки графічного інтерфейсу від розробки бізнес-логіки, відомої як модель. Модель представлення - це та частина, яка відповідає за перетворення даних для їх постійної підтримки та використання. З цієї точки зору модель представлення більше схожа на модель, ніж на представлення і вона обробляє більшість, якщо не всю, логіку відображення даних.

MVVM впорядковує код, щоб можна було змінювати його частини, не зачіпаючи інших. Це має багато переваг, зокрема:

- можна використовувати ітеративний, довільний стиль написання коду;
- спрощене тестування модулів;
- ефективніше використання інструментів проектування.

При використанні шаблону MVVM додаток Android поділяється на такі частини:

1. Інтерфейс, який розробляється технологіями XML.
2. Логіка інтерфейсу користувача реалізується як компонент ViewModel.
3. Функціональні зв'язки між користувальницьким інтерфейсом та ViewModel реалізуються через прив’язки (bindings).

View – базовий клас для всіх віджетів інтерфейсу. Інтерфейс додатків Android - це дерево нащадків цього класу.

Клас Activity та його підкласи містять логіку, що реалізує користувальницький інтерфейс. Цей клас відповідає ViewModel в архітектурному шаблоні Model-View-ViewModel (MVVM).

У магістерській кваліфікаційній роботі пропонуємо п’ять рівнів моделей представлення підкласів класу Activity(див. рис. 2.1).

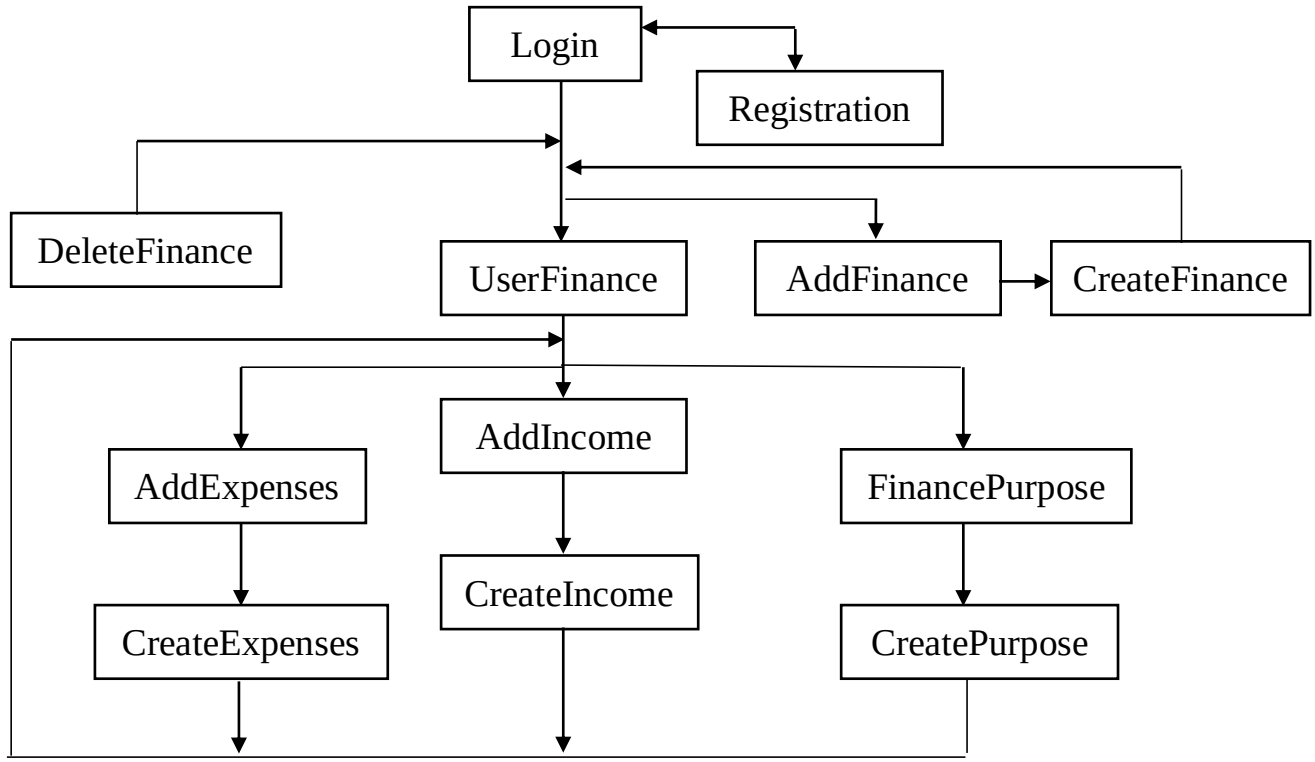


Рисунок 2.1 – Ієрархічна модель додатку

1. Перший рівень – це вікно авторизації, де користувачеві потрібно ввести свій логін і пароль.

2. Якщо користувач не зареєстрований, то спочатку треба перейти на другий рівень – рівень реєстрації, після цього повернутися на перший рівень.

3. Третій рівень – рівень фінансів користувача. В даному вікні можна додавати і видаляти фінанси, а також переглядати свої витрати і надходження.

4. Четвертий – рівень обліку фінансів. Користувач може додавати витрати або доходи.

5. Обравши одну з функцій четвертого рівня, користувач переходить до п’ятого рівня, де можна додати детальну інформацію про витрати чи доходи.

2.7 Розробка інтерфейсу системи.

Найважливішим етапом створення програмного продукту є розробка дизайну і інтерфейсу мобільного додатку. Саме від неї залежить як користувач сприйме додаток, чи захоче й надалі ним користуватись та чи буде додаток зручним в експлуатації, а також чи набере популярності серед користувачів.

Інтерфейс користувача [18] – засіб зручної взаємодії з інформаційною системою. Набір інструментів для обробки та відображення інформації, максимально адаптованої для зручності користувача. Основна мета будь-якої програми – забезпечити зручність та ефективність роботи з інформацією: графікою, базами даних, документами, зображеннями.

Інтерфейси повинні мати такі характеристики:

1. Чіткість. Інтерфейс дозволяє уникнути неоднозначності та чітко пояснює мову, потік, ієрархію та метафори для візуальних елементів.
2. Виразність. Якщо на екрані занадто багато елементів, знайти те, що шукаєте, важко, і через це інтерфейс стає стомлюючим у використанні. Справжнє завдання створення ідеального інтерфейсу - зробити його лаконічним і зрозумілим одночасно.
3. Знайомство. Навіть якщо хтось використовує інтерфейс вперше, певні елементи можуть бути фактично знайомі.
4. Відповідність. Прекрасний інтерфейс не повинен бути млявим. Це означає, що він повинен надавати користувачеві хороший зворотній зв'язок щодо того, що відбувається і чи вдало обробляється введення користувача.
5. Послідовність. Важливо, щоб інтерфейс відповідав вашому додатку, оскільки він дозволяє користувачам розпізнавати схеми використання.
6. Естетика. Дизайн має виглядаи привабливо, щоб користувачеві було приємно користуватись програмою та частіше в неї заходити.
7. Продуктивність. Час - це гроші, а класний інтерфейс повинен зробити користувача ефективним, із ярликами та хорошим дизайном.

При розробці мобільного додатку було дотримано всіх вище наведених правил та розроблено оптимальний на вигляд і для застосування інтуїтивно зрозумілий інтерфейс.

Було вирішено, що в додатку будуть наступні вікна:

1. Вікно реєстрації/авторизації. Реєстрація буде складатись з 3 текстових полів та кнопки зареєструватись. Авторизація буде складатись з двох полів та кнопки вхід.

2. Основне меню. Тут буде відображено такі пункти: надходження, витрати, баланс, а також «+», за допомогою якого можна додати витрату або надходження.

3. Вікно додавання витрат/надходжень. Буде відображено поля: категорія призначення для витрат/для надходжень відсутнє поле, сума, дата, нотатка.

2.8 Розробка структури бази даних

Процес нормалізації є одним з найважливіших, що визначає процедури приведення моделі бази даних до виду, формує ефективну базу даних, що реалізується в коректному поданні і високошвидкісний обробці даних. Запропоновано нову організацію бази даних, яка базується на конвеєрному принципі обробки даних, що дозволить оптимізувати Структуру збереження інформаційних ресурсів.

Щоб виконати нормалізацію моделі бази даних, необхідно представити модель у вигляді сутності з набором атрибутів, які це будуть характеризувати, визначаючи інформаційне опис предметної області. У цій суті будуть виникати поєднання атрибутів, які представляються функціональними залежностями. Нижче наведена база даних системи (рис. 2.2).

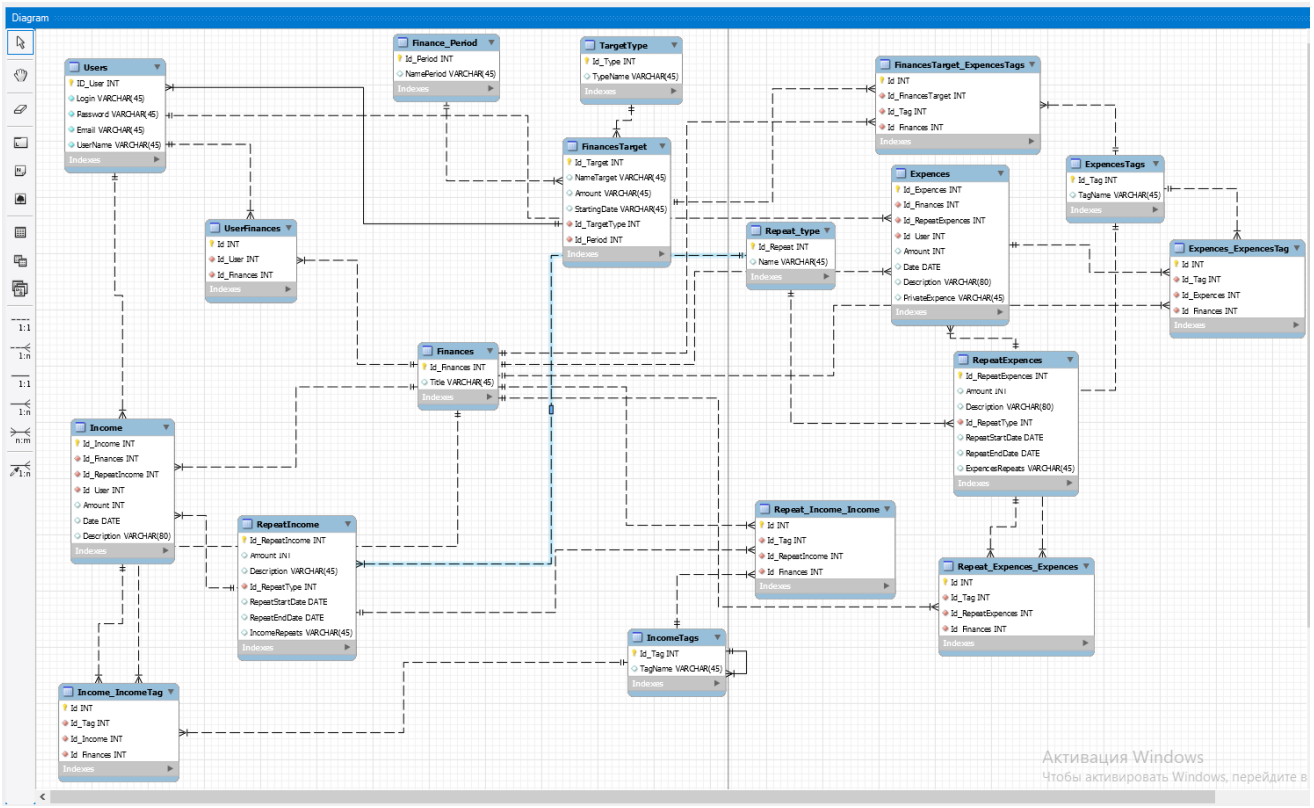


Рисунок 2.2 – Структура бази даних

У роботі база даних зводиться до третьої нормальної форми, тобто вона відповідає вимогам першої та другої нормальних форм [19]:

Перша нормальна форма (1NF, 1NF) формує грунт для структурованої схеми бази даних:

- Кожна таблиця повинна мати первинний ключ: мінімальний набір стовпців, що ідентифікують запис.
- Уникаючи групових повторів (категорії даних, які можуть виникати в різний час у різних записах) правильно визначають не ключові атрибути.
- Атомність: кожен атрибут повинен мати лише одне значення, а не кілька значень.

Друга нормальна форма (2NF, 2NF) вимагає, щоб дані, що зберігаються в таблицях із складеним ключем, не залежали лише від частини ключа:

- Схема бази даних має відповідати вимогам першої нормальної форми.
- Кілька рядків відображаються в окремих таблицях.

Третя нормальна форма (3NF, 3NF) вимагає, щоб дані таблиці залежали виключно від головного ключа:

- Схема бази даних повинна відповідати всім вимогам другої нормальної форми.
- Будь-яке поле, яке залежить від головного ключа та будь-якого іншого поля, повинно бути вказане в окремій таблиці.

2.9 Формування таблиць бази даних системи та моделі системи

Модель ER - це представлення бази даних у вигляді графічних діаграм. Модель ER візуалізує процес, який ідентифікує конкретну предметну область. Діаграма взаємозв'язку сутність - це схема, яка графічно представляє сутність, атрибути та відносини.

Модель ER - це лише концептуальний рівень моделювання. Модель ER не містить детальних відомостей про реалізацію. Деталі його реалізації можуть відрізнятися для тієї ж моделі ER. Модель автоматизованої системи наведено на рис. 2.3.

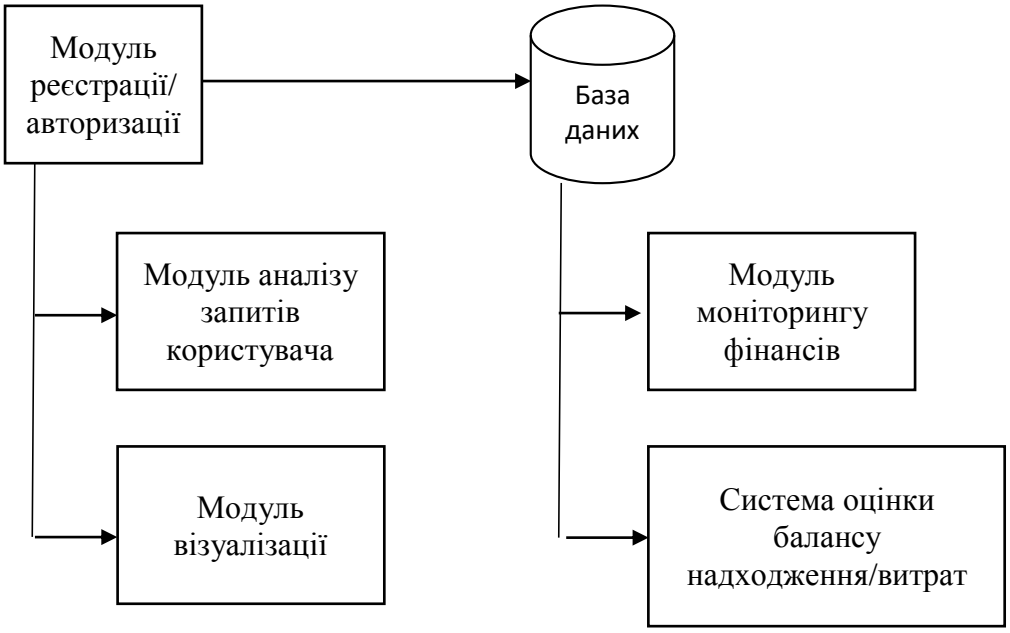


Рисунок 2.3 – Модель автоматизованої системи

Сутність бази даних - це будь-який об’єкт у базі даних, який можна виділити виходячи із суті предметної області, для якого база даних розробляється.

У моделі “сутність”-“зв’язок” розрізняють два різновиди типів сутностей:

- слабкий тип. Цей тип сутності є залежним від сильної сутності;
- сильний тип. Це самостійний тип сутності, який ні від кого не залежить.

ER-модель сутності «User» наведено далі на рис. 2.4.

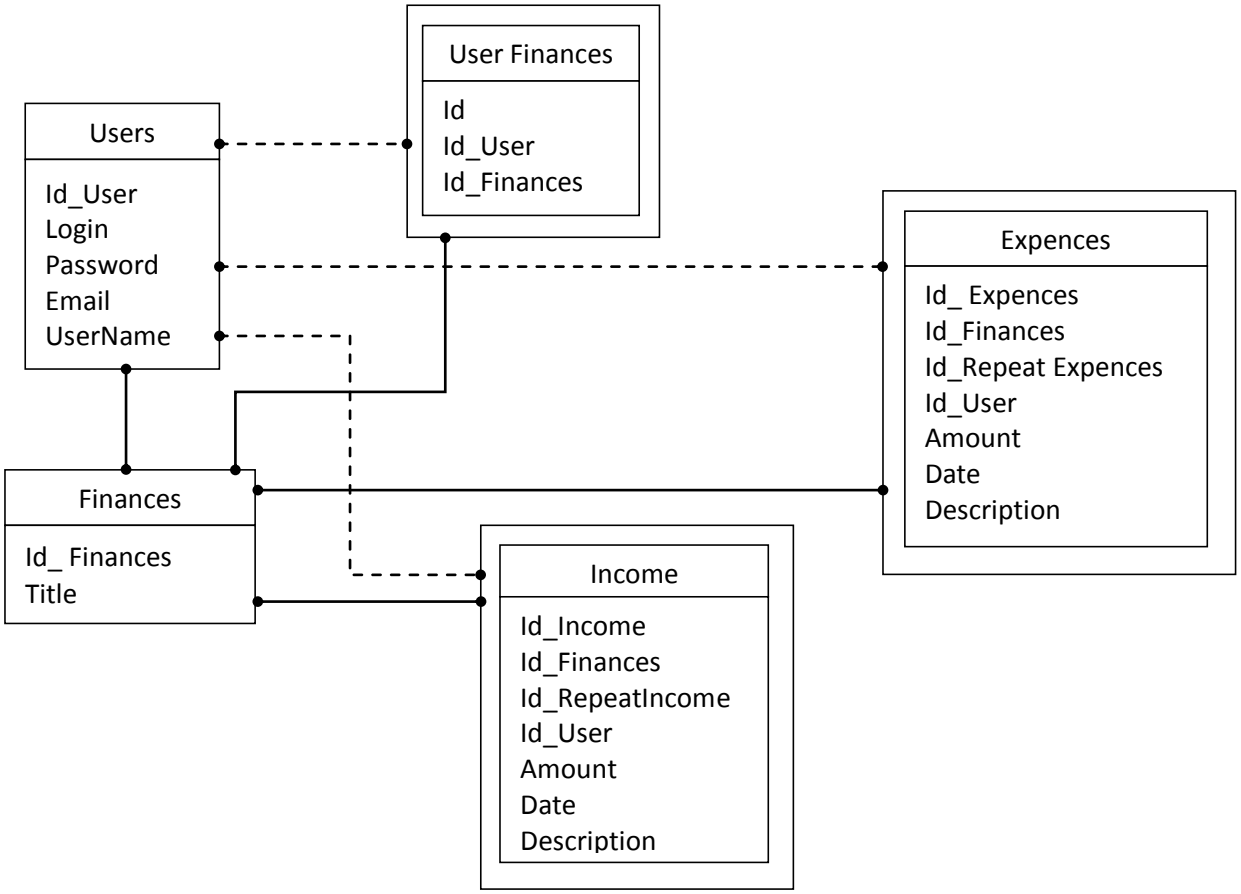


Рисунок 2.4 – Фрагмент ER-моделі для типу сутності «User»

Зв’язано таблиці Users та Finances, а Finances в свою чергу зв’язана з UserFinances, Income та Expences, які належать до фінансів та до самого користувача через зв’язок між таблицями.

2.10 Розробка архітектури та алгоритму роботи мобільного додатку

Архітектура самого мобільного сервісу складається з системи «Клієнт-Сервер» (рис.2.5), що широко використовується в подібних мобільних додатках. Потім будується система зв’язку клієнтського мобільного додатку із серверною базою даних (БД) та набором скриптів до виконання.

Місце «Клієнта» займає мобільний пристрій ОС Android із налаштованим мобільним додатком. Він надсилає, приймає та оброблює запити на сервер, інтерпретує дані таким чином, щоб користувачу було зручно їх переглядати.

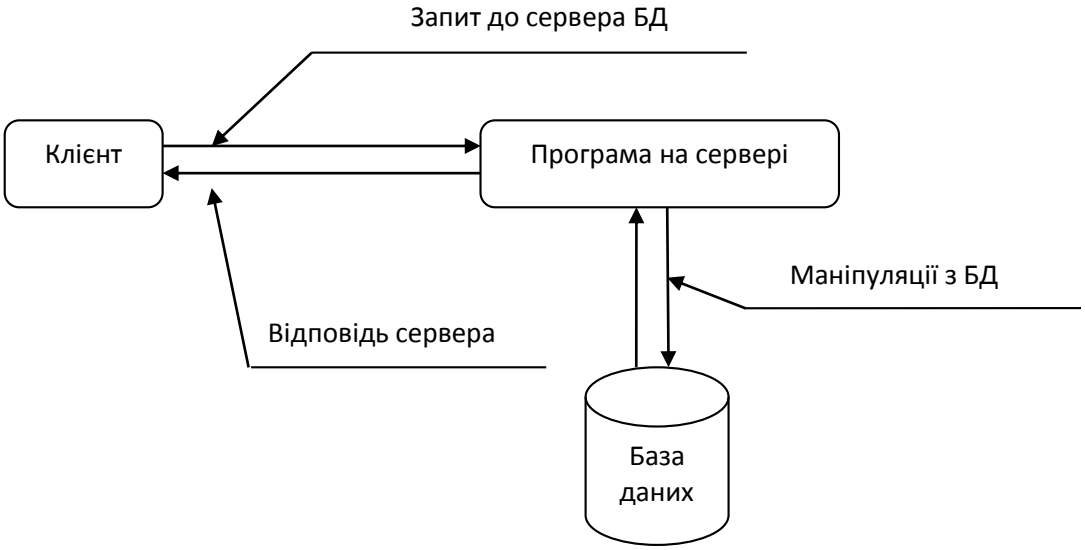


Рисунок 2.5 – Архітектура мобільного сервісу

Алгоритм [20] - це точний і чіткий опис послідовності дій над заданими об’єктами, що дозволяє отримати кінцевий результат.

Властивості алгоритмів:

1. Скінченність. Кожен алгоритм повинен мати кінцеву кількості кроків.

2. Продуктивність. Реалізація алгоритму завжди повинна призводити до певного результату. 3. Формальність. Відповідно до алгоритму, виконавець повинен отримати результат, не заглиблюючись у його суть. Ця властивість має особливе значення для автоматизованого виконання алгоритмів.

4. Визначеність. Будь-який алгоритм повинен бути описаний так, щоб у виконавця не було неоднозначних інструкцій. Тобто різні виконавці алгоритму повинні діяти однаково і отримувати однаковий результат.

5. Масовість. Створений алгоритм може вирішити цілий клас проблем.

6. Зрозумілість. Повинен містити лише знайомі користувачеві інструкції.

7) Результативність – завершення алгоритму певними результатами.

Алгоритм розробленої автоматизованої системи є розгалуженим, оскільки містить декілька умов, в результаті яких здійснюється поділ на кілька паралельних гілок алгоритму (див. рис. 2.6).

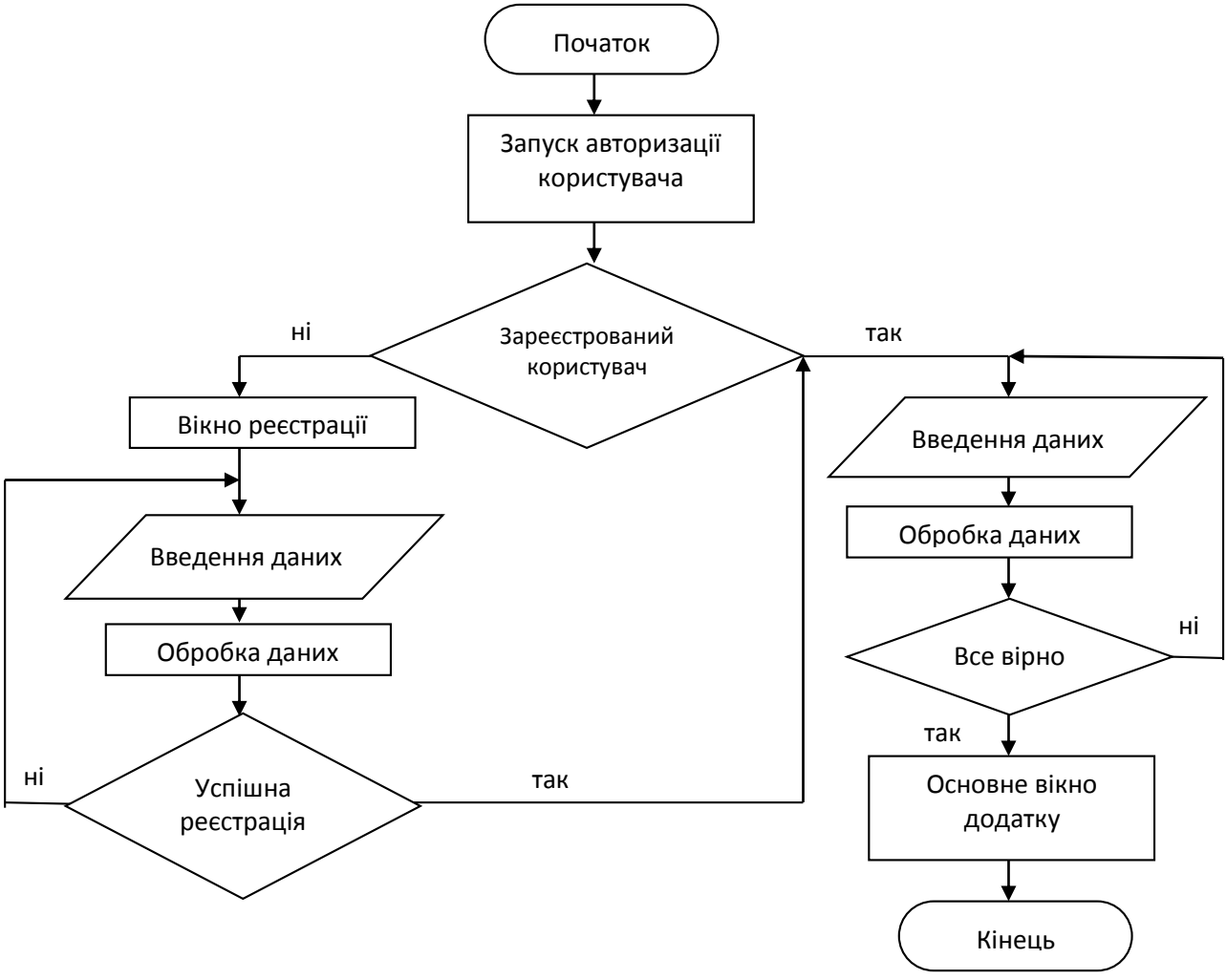


Рисунок 2.6 – Блок-схема алгоритму авторизації користувача

Алгоритми залежно від мети, початкових умов завдання, способів її розв’язання, визначення дій виконавця поділяються так:

- механічні алгоритми - визначають певні дії, маркуючи їх в єдиній та надійній послідовності, забезпечуючи чіткий необхідний/бажаний результат;
- гнучкі алгоритми;
- імовірнісний алгоритм дає рішення задачі кількома шляхами;
- евристичний алгоритм – використовує розумні міркування без строгих обґрунтувань;
- лінійний алгоритм – набір команд, послідовних в часі один за одним;
- розгалужений алгоритм – алгоритм, який містить хоча б одну умову, може здійснюватися поділ на кілька паралельних гілок алгоритму;
- циклічний алгоритм – алгоритм, що передбачає багаторазове повторення однієї і тієї ж дії.

2.11 Висновки

У розділі були наведені вхідні та вихідні дані системи ведення власних фінансів. Розглянуто структури та принципи розробки мобільних додатків. Також проаналізовано особливості розробки баз даних під платформу Android.

Розроблено метод системи автоматизованого ведення власних фінансів. Було розроблено також ієрархічну модель системи автоматизованого ведення власних фінансів, яка складається з п’яти рівнів. Було розроблено інтерфейс системи, в якому було складено чіткий список необхідних вікон з їх складовими.

Здійснено розробку структури бази даних. Було розроблено ER-модель та архітектуру мобільного додатку та модель автоматизованої системи. Наведено блок-схему алгоритму роботи блоку реєстрації та авторизації користувача, який належить до розгалуженого типу.

3 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ МОБІЛЬНОГО ДОДАТКУ

3.1 Варіантний аналіз і обгрунтування вибору засобів реалізації автоматизованої системи ведення фінансів під Android.

Існує багато програм для написання мобільних додатків. Найвідоміші з них: Java, Kotlin, Windows Phone SDK, C/C++, BASIC, Lua (з використанням Corona SDK).

Java [21] — об'єктно-орієнтована мова програмування. У мові є купа особливостей у вигляді конструкторів класів, винятків, що призводять до падіння додатків під час роботи і інших моментів, які завжди необхідно враховувати при розробці. Втім, код на Java легко читається і структурується, особливо при дотриманні прийнятих стандартів його оформлення.

При розробці на Java під Android використовуються не тільки Java-класи, що містять код, але також файли маніфесту на мові XML, що надають системі основну інформацію про програму, і системи автоматичного складання Gradle, Maven або Ant, команди в яких пишуться на мовах Groovy, POM і XML відповідно; за замовчуванням в проектах використовується Gradle, а на початкових етапах навчання розробці на Java правити файли, написані на Groovy, практично не доведеться. Для верстки UI-частини зазвичай також використовується мова XML.

Kotlin був офіційно представлений в травні 2017 року на Google I / O і позиціонується Google як друга офіційна мова програмування під Android після Java, тільки трохи простіша для розуміння. Знання Java необхідні тут, щоб розуміти принципи роботи Kotlin, загальну структуру мови та її особливості [22].

Kotlin сумісний з Java і не викликає зниження продуктивності і збільшення розміру файлів. Відмінність від Java в тому, що він вимагає менше службового, так званого boilerplate-коду, тому більш обтічний і легкий для читання. Його творцям вдалося уникнути nullpointexceptions, а компіляція більше не переривається через дрібниці на зразок забутого знака «;».

Ще одна технологія - це технологія розробки Windows Phone SDK. Код програми, що розробляється, описаний у XAML. Хоча це насправді лише розмітки XML-файлів розмітки XAML. Платформа Windows Phone - це не просто ще одна мобільна платформа. Вона має не тільки технологічну складову, але і всебічно розроблену концепцію дизайну інтерфейсів та взаємодії з користувачем, що називається Metro design або Metro style. Вся розробка під Windows Phone здійснюється у Visual Studio. Для мобільних додатків Windows Phone налагодження та тестування проводиться за допомогою емулятора Windows Phone у програмі Windows Phone [21].

C / C ++ більш низькорівневі мови, які також підтримуються Android Studio з використанням Java NDK. Це дозволяє писати нативні додатки, що може стати в нагоді для створення ігор або інших ресурсоємних програм. Android Studio пропонує підтримку C / C ++ через Android NDK (Native Development Kit).

Це означає, що код буде запускається не через Java Virtual Machine, а безпосередньо через девайс, що дасть вам більше контролю над такими елементами системи, як пам'ять, сенсори, жести і т. Д., а також можливість вижати з Android-пристроїв максимум ресурсів. У свою чергу, він складний в налаштуванні і не дуже зручний, тому рекомендується використовувати його для написання тільки тих модулів програми, де необхідно швидко робити складні операції: обробку і рендеринг графіки, відео і складних 3D-моделей.

Завдяки своїй простоті і доброзичливості BASIC є оптимальною точкою входу для початківців програмістів. Погана новина: він не підтримується Android Studio і не підходить для середовищ Unity і Xamarin. Хороша новина: для BASIC є спеціальне середовище розробки B4A, в якій можна створювати Android-додатки. Оскільки BASIC не підтримується Android Studio, тому не підходить для розробки додатку.

Мовою Lua заснований багатоплатформовий графічний движок Corona. LUA значно простіше Java, а Corona SDK зробить роботу з цією мовою легкої і приємною. Він підтримує всі нативні бібліотеки, дозволяючи тим самим писати під безліч платформ [21]. Щоб писати код, знадобиться Notepad ++, а щоб його

запустити без попередньої компіляції, потрібен емулятор. Якщо APK зібраний і програма готова до розгортання, то запустити додаток можна онлайн.

Без обмежень не обійшлося, і обмежень таких, які унеможливлюють розробляти серйозні речі і утвердитися в статусі професіонала. Якщо в додатку вам потрібна функціональність кшталт внутрішніх покупок, то за можливість її розробити доведеться платити, як і за використання нативного Android API [22].

Проаналізувавши мови для написання мобільних додатків для Android можна зробити висновок, що найдоцільніше використовувати мову Java для написання програмного продукту. Оскільки інші програми поступаються своїми можливостями та функціоналом.

3.2 Розробка бази даних системи

Android має вбудовану підтримку бази даних SQLite. Всі функції SQLite підтримуються, надається API оболонки із сумісним інтерфейсом. API Android SQLite є типовим; розробник повинен реалізувати всю обробку баз даних, включаючи створення, управління версіями, оновлення бази даних та інші налаштування [17].

При підключенні до бази даних у додатку необхідно вказати ім'я бази даних та її версію. Можуть виникнути наступні ситуації:

1) бази даних не існує. Це може бути, якщо це первинна установки програми. У такому випадку додаток має сам створити базу даних та всі таблиці в ній. І далі вже працює з новою базою;

2) база даних існує, але її версія застаріла. Це може бути при оновленні програми. Нова версія може потребувати додаткових полів в старих таблицях або нові таблиці. Тоді додаток повинен оновити існуючі таблиці та створити нові, якщо це необхідно;

3) база даних існує і її версія актуальна. Додаток успішно підключається до бази даних і функціонує.

Для обробки вище наведених ситуацій треба створити клас, який є нащадком для SQLiteOpenHelper (клас для створення бази даних). SQLiteOpenHelper має два абстрактних методи:

- onCreate – задає початкову установку параметрів при ініціалізації активності. Метод буде викликаний, якщо база даних, до якої треба підключитися, не існує;
- onUpgrade – буде викликаний при модифікації бази даних.

Створений нащадковий клас DBHelper повинен реалізувати вищевказані методи, записавши логіку створення бази даних та модифікації.

Клас DBHelper також неявно імітує інтерфейс BaseColumns, який задає константну строку _id, що представляє ім'я поля для ідентифікаторів запису. У створюваних таблицях бази даних поле _id повинно мати тип integer primary key autoincrement. Описувач autoincrement – необов'язковий.

Частина коду створення таблиць бази даних наведено нижче:

```
private static final String DATABASE_NAME = "PFdatabase.db"; private static final int DATABASE_VERSION = 3;
private static final String DATABASE_TABLE_USERS = "Users";
private static final String DATABASE_TABLE_BPERIOD = "Budget_Period"; private static final String DATABASE_TABLE_TTYPE = "TargetType"; private static final String DATABASE_TABLE_BUDGET = "Budget";
static final String LOGIN_COLUMN = "login";
static final String PASSWORD_COLUMN = "password";
static final String EMAIL_COLUMN = "email";
static final String USER_NAME_COLUMN = "user_name"; static final String NAME_PERIOD_COLUMN = "name_period"; static final String NAME_TYPE_COLUMN = "name_type"; static final String TITLE_COLUMN = "title";
private static final String DATABASE_CREATE_SCRIPT_USERS = "create table "
+ DATABASE_TABLE_USERS + " (" + BaseColumns._ID
+ " integer primary key autoincrement, " + LOGIN_COLUMN
+ " text not null, " + PASSWORD_COLUMN +" text not null, " +
EMAIL_COLUMN
+ " text not null," + USER_NAME_COLUMN + "text not null);";
private static final String DATABASE_CREATE_SCRIPT_BPERIOD = "create table "
+ DATABASE_TABLE_BPERIOD + " (" + BaseColumns._ID
+ " integer primary key autoincrement, " + NAME_PERIOD_COLUMN
+ " text not null);";
private static final String DATABASE_CREATE_SCRIPT_TTYPE = "create table "
```

```
+ DATABASE_TABLE_TTYPE + " (" + BaseColumns._ID
+ " integer primary key autoincrement, " + NAME_TYPE_COLUMN
+ " text not null);";
private static final String DATABASE_CREATE_SCRIPT_BUDGET = "create table "
+ DATABASE_TABLE_BUDGET + " (" + BaseColumns._ID
+ " integer primary key autoincrement, " + TITLE_COLUMN
+ " text not null);";
@Override
public void onCreate(SQLiteDatabase db) {
 db.execSQL(DATABASE_CREATE_SCRIPT_USERS);
db.execSQL(DATABASE_CREATE_SCRIPT_BPERIOD);  db.execSQL(DATABASE_CREATE_SCRIPT_TTYPE);
db.execSQL(DATABASE_CREATE_SCRIPT_BUDGET);
}
@Override
public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion) { db.execSQL("DROP TABLE IF
IT EXISTS" + DATABASE_TABLE_USERS); db.execSQL("DROP TABLE IF IT EXISTS" +
DATABASE_TABLE_BPERIOD); db.execSQL("DROP TABLE IF IT EXISTS" + DATABASE_TABLE_TTYPE);
db.execSQL("DROP TABLE IF IT EXISTS" + DATABASE_TABLE_BUDGET); onCreate(db);
}
```

У методі зворотного виклику onCreate() потрібно реалізувати логіку створення таблиць. Метод onUpdate () викликається під час оновлення програми зі зміненою структурою таблиць.

3.3 Розробка дизайну додатку

Основним кольором додатку було обрано зелений, адже в більшості людей гроші асоціюються саме з цим кольором. Загальна гамма відповідає сучасним правилам поєднання кольорів та є ненав’язливою та гарно сприймається людським оком. Текст виконано в різних варіаціях підтонів сірого кольору. На рисунку 3.1 наведено вікно додатку «Авторизація користувача».

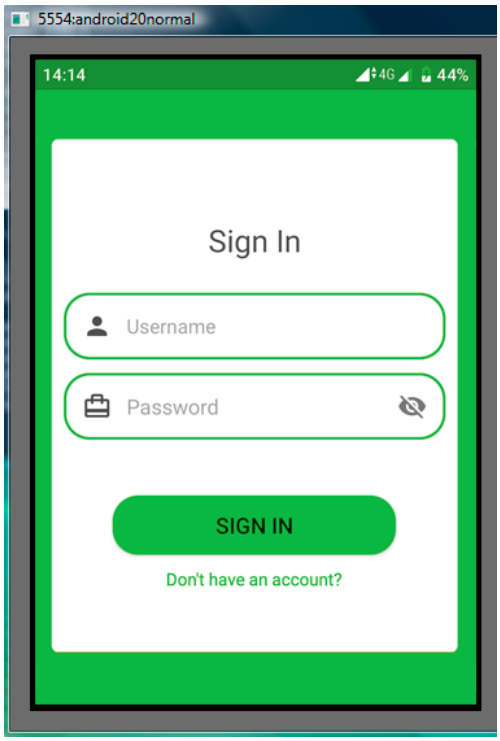


Рисунок 3.1 – Авторизація користувача

Також розроблено інтуїтивно зрозумілий інтерфейс для додавання витрат чи транзакцій (рис. 3.2), де потрібно вибрати категорію, ввести суму, обрати дату та по бажанню додати нотатку.

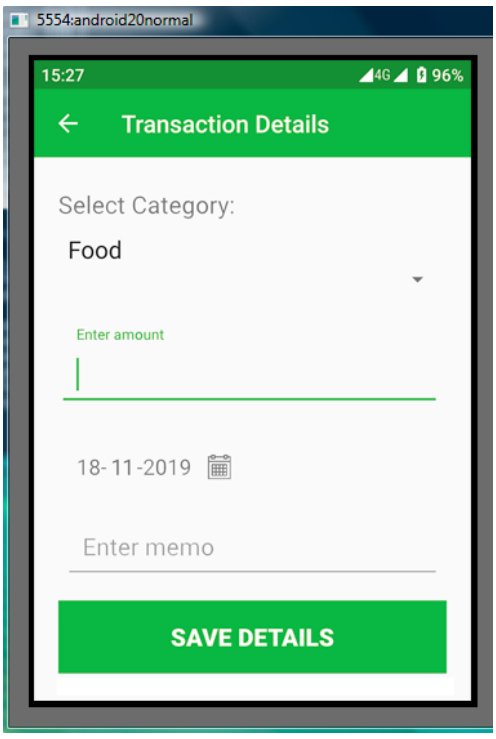


Рисунок 3.2 – Додавання витрат

3.4 Програмна реалізація інтерфейсу

Макет визначає візуальну структуру користувальницького інтерфейсу, наприклад, призначеного для користувача інтерфейсу операції або віджета додатка. Існує два способи оголосити макет:

1. Оголошення елементів користувацького інтерфейсу в XML. В Android є зручний довідник XML-елементів для класів View і їх підкласів, наприклад таких, які використовуються для віджетів і макетів.

2. Створення екземплярів елементів під час виконання. Програма може програмним чином створювати об'єкти View і ViewGroup (а також керувати їх властивостями).

Розробляючи Android-додаток, потрібно виконати візуальну частину вікон. Програмний код розмітки екрану знаходиться у файлах з розширенням .xml у папці проекту під назвою «layout».

Розмітка для Android додатків будується з використанням ієрархії View і ViewGroup об'єкти. View об'єкти це графічні елементи для інтерфейсу користувача, такі як кнопки чи текстові поля, ViewGroup це невидимий вид контейнерів, які визначають розташування в сітці або вертикальному списку.

Існує кілька стандартних типів розміток [23]:

1. FrameLayout - найпростіший тип розмітки. Може містити тільки один дочірній елемент типу View або ViewGroup, який буде вирівнюватися по верхній лівій межі. При додаванні наступних елементів вони будуть перекривати один одного.

2. LinearLayout на відміну від FrameLayout може містити більше одного елемента. Як випливає з назви, LinearLayout є лінійною розміткою, дочірні елементи якої розміщуються один за одним. Може розміщувати елементи як горизонтально, так і вертикально. При вертикальному розміщенні кожен елемент займає окремий рядок, а при горизонтальному - елементи розміщуються в одному рядку, причому висота рядка є висотою найбільш високого дочірнього елемента.

3. TableLayout дозволяє створювати розмітку за подобою таблиці в html. Рядок таблиці, TableRow, може не мати комірок або мати кілька комірок. Допускається також комірки типу ViewGroup. Тобто, комірка може бути іншим типом розмітки, в тому числі і TableLayout.

4. RelativeLayout - елементи цього способу розмітки можуть розташовуватися відносно один одного, не дотримуючись принципів поведінки LinearLayout і TableLayout.

5. GridLayout для будь-якого компонента можна вказати рядок і колонку, і в цьому місці таблиці він буде розміщений. Вказувати елемент для кожного осередку не знадобиться, це потрібно робити тільки для тих осередків, де дійсно повинні бути компоненти. Компоненти можуть розтягуватися на кілька осередків таблиці. Більш того, в одну клітинку можна помістити кілька компонентів.

Код розмітки класу LoginActivity наведений нижче:

```
<RelativeLayout
   xmlns:android="http://schemas.android.com/apk/res/android"
   android:layout_width="match_parent"
   android:layout_height="match_parent"
   android:gravity="center"
   android:orientation="vertical">
   <EditText
      android:id="@+id/login"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_margin="15dp"
      android:hint=" login"
      android:inputType="text login " />
   <EditText
      android:id="@+id/loginpaswd"
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:layout_below="@id/loginl"
      android:layout_margin="15dp"
      android:hint="Password"
      android:inputType="numberPassword"/>
   <Button
```

```
        android:id="@+id/btnLogIn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/loginpaswd"
        android:layout_marginTop="15dp"
        android:layout_marginLeft="15dp"
        android:layout_marginRight="15dp"
        android:text="Login"
        android:textSize="25dp" />
    <TextView
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:id="@+id/OR_button"
        android:layout_below="@id/forgotPass"
        android:layout_marginTop="14dp"
        android:text="OR"
        android:gravity="center"
        android:textColor="@color/colorPrimary"/>
    <TextView
        android:id="@+id/TVSignIn"
        android:layout_width="match_parent"
        android:layout_height="wrap_content"
        android:layout_below="@id/OR_button"
        android:gravity="center_horizontal"
        android:layout_margin="15dp"
        android:text="New User? Register Here"
        android:textSize="20sp" />
    <com.google.android.gms.common.SignInButton
        android:id="@+id/sign_in_button"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_below="@id/TVSignIn"
        android:layout_centerHorizontal="true"
        android:layout_centerVertical="true" />
</RelativeLayout>
```

LoginActivity реалізовано за допомогою розмітки RelativeLayout, так як у даному випадку її найзручніше використовувати.

Оголошення інтерфейсу користувача в файлах XML дозволяє відокремити інтерфейс від коду, тому можна змінювати інтерфейс без зміни і перекомпіляції коду.

3.5 Програмна реалізація модулів

3.5.1 Розробка модуля реєстрації

Запустивши додаток, користувач бачить MainActivity – вікно реєстрації користувача, де йому необхідно ввести свій логін і пароль для подальшої взаємодії з додатком. Частина коду реалізації MainActivity наведена нижче:

```
private void registerAndLogin() {
        firebaseAuthenticationClient.createUser(currentUserLogin, currentUserPassword,
                new SimpleLoginAuthenticatedHandler() {
                        public void authenticated(Error error, User user) {
                                if (error != null) {
                                        // There was an error creating this account.
                                        } else {
                        AppUser newUser = new AppUser(currentUserFullName, currentUserLogin);
                        firebaseRef.child("users").push().setValue(newUser);
                                Toast.makeText(MainActivity.this, "Register complete!",
                                Toast.LENGTH_SHORT).show();
                        attemptLogin();
                                        }
                                }
                        });
}
```

Спочатку відбувається перевірка на правильність заповнення полів вводу даних за допомогою конструкції if-else. Якщо поля заповнені неправильно, то з’являється помилка створення аккаунту, якщо все пройшло успішно – реєстрація завершена.

3.5.2 Розробка модуля авторизації

Якщо ж користувач вже зареєстрований, то йому потрібно просто ввести дані логіну і паролю для входу. Частина коду реалізації наведена нижче:

```
public class AuthenticationManager {
private Firebase firebaseRef = new Firebase(DatabaseManager.firebaseURL);
private SimpleLogin firebaseAuthenticationClient = new SimpleLogin(firebaseRef);
private String errorMessage = "";
```

```
private Boolean queryFinished;
// Returns empty string if success, else returns error message.
public String attemptLoginWithLogin (String login, String password) {
        queryFinished = false;
        firebaseAuthenticationClient.loginWithLoginl(login, password,
                        new SimpleLoginAuthenticatedHandler() {
                                public void authenticated(Error error, User user) {
                                        if (error != null) {
                                                // There was an error logging into this account.
                                                if (error == Error.InvalidPassword) {
                                                        errorMessage = "Password is incorrect!";
                                                } else if (error == Error.InvalidLoginl) {
                                                errorMessage = "Login doen't exist!";
                                                }
                                        } else {
                                                errorMessage = "Login complete!";
                                        }
                                        queryFinished = true;
                                }
                        });
        DatabaseManager.waitForQueryResult(queryFinished);
        return errorMessage;
}
```

Після введення даних в поля логіну і паролю проводиться перевірка на їх правильність. Якщо все введено вірно – відображається основне вікно програми, якщо ні – сповіщення про помилку та повторне введення даних користувача.

3.5.3 Розробка модуля транзакцій

Щоб додати витрату чи транзакцію, потрібно вибрати категорію призначення, дату, суму та написати нотатку(по бажанню). Частина коду реалізації наведена нижче:

```
public class TransactionActivity extends AppCompatActivity {
   Realm realm;
   ArrayList<String> categories;
   ArrayAdapter adapter;
   Spinner cat;
   private String categorySelected;
   private EditText amount;
   private EditText memo;
```

```
private TextView date;
private Button saveButton;
private ProgressBar progressBar;
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_transaction);
    getSupportActionBar().setTitle("Transaction Details");
    getSupportActionBar().setDisplayHomeAsUpEnabled(true);
    realm = Realm.getDefaultInstance();
    cat = (Spinner) findViewById(R.id.spinner_category);
    CategoryActivity helper = new CategoryActivity();
    progressBar = findViewById(R.id.progress_bar);
    categories = helper.getCategoryName();
    adapter = new ArrayAdapter<String>(this, android.R.layout.simple_spinner_item, categories);
    adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
    cat.setAdapter(adapter);
    // spinner onclick
    cat.setOnItemSelectedListener(new AdapterView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> parent, View view, int position, long id) {
            categorySelected = categories.get(position);
        }
        @Override
        public void onNothingSelected(AdapterView<?> parent) {
        }
    });
    amount = findViewById(R.id.trans_amt);
    date = findViewById(R.id.date);
    Date date2 = Calendar.getInstance().getTime();
    SimpleDateFormat simpleDateFormat = new SimpleDateFormat("dd-MMM-yyyy");
    String formattedDate = simpleDateFormat.format(date2);
    date.setText(formattedDate);
    memo = findViewById(R.id.memo);
    saveButton = findViewById(R.id.btn_details);
    saveButton.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            if (!amount.getText().toString().isEmpty()){
                addTransactionToDatabse(categorySelected, amount.getText().toString(), date.getText().toString(),
                        memo.getText().toString());
                finish();
```

```
            }
                //Toast.makeText(this,  "Pl", Toast.LENGTH_SHORT).show()
            }
        });
        // List<String> cat1 = new ArrayList<>();
        // cat1.add("Select Category");
        // retrieve spinner data
    }
        private void addTransactionToDatabse(String category, String amount, String date, String memo){
        progressBar.setVisibility(View.VISIBLE);
        String Id = "TRA" + String.valueOf(Calendar.getInstance().getTimeIn Millis());
        realm.beginTransaction();
        Transaction transaction = realm.createObject(Transaction.class, Id);
        transaction.setAmount(amount);
        transaction.setCategory(category);
        transaction.setDate(date);
        transaction.setMemo(memo);
        realm.commitTransaction();
        progressBar.setVisibility(View.GONE);
    }
}
```

Після того, як всі поля будуть заповнені, потрібно натиснути на кнопку «зберегти деталі» та транзакція буде висвітлена у списку витрат.

3.6 Висновки

У розділі було проведено варіантний аналіз і обгрунтування вибору засобів реалізації автоматизованої системи ведення фінансів під Android. Було обрано мову Java для написання програмного продукту.

Розроблено базу даних додатку за допомогою SQLite. Розроблено інтуїтивно-зрозумілий інтерфейс користувача та обґрунтовано вибір кольорової гами та складових вікон. Розроблено розмітку екрану Android-додатку за допомогою XML та наведено частину коду реалізації розмітки вікна LoginActivity. Наведено частину коду вікна авторизації користувача. Реалізовано програмні модулі додатку.

4 ТЕСТУВАННЯ РОЗРОБЛЕНОГО ПРОГРАМНОГО ПРОДУКТУ

Тестування програмного забезпечення – це процес технічного дослідження, призначений для виявлення інформації про якість продукції щодо контексту, в якому вона повинна використовуватися. Методи тестування також включають процес пошуку помилок чи інших дефектів та тестування програмних компонентів для оцінки [24].

Роль тестування в розробці програмного забезпечення:

1. Тестування дозволяє перевірити, чи правильно реалізовано усі вимоги до ПЗ, що розроблялось.
2. Тестування допомагає виявити дефектів / помилок та забезпечує їх розпізнавання / вирішення до етапу розгортання програмного забезпечення.
3. Тестування пом’якшує наслідки та ризики втрат, якщо програмний продукт все ж випустили по неправильних вимогах. Вимоги в такому випадку намагаються частково виправити та переоцінити, а також покращити програму.
4. Тестування також демонструє, що створене ПЗ працює правильно та відповідає вимогам до продуктивності.
5. Тестування допомагає перевірити належну інтеграцію та взаємодію програми з навколишнім середовищем.

Програмний продукт є якісним, коли [25]:

- під час роботи користувача з програмним продуктом не виникає багато відмов;
- програмний продукт надійний, тобто його використання рідко викликало аварійні відмови чи вихід з програми;
- програмний продукт задовольняє вимоги більшості користувачів.

Тестування програмного коду - процес виконання програмного коду, спрямований на виявлення існуючих в ньому дефектів. Під дефектом тут розуміється ділянка програмного коду, виконання якого за певних умов призводить до несподіваного поведінки системи.

4.1 Аналіз методів тестування

4.1.1 Тестування методом чорного ящика

Тестування методом «чорного ящика» [24], також відоме як тестування, засноване на специфікації або тестування поведінки – техніка тестування, заснована на роботі виключно з зовнішніми інтерфейсами тестованої системи.

Тестування чорного ящика – це:

1. Тестування, як функціональне, так і не функціональне, що не передбачає знання внутрішнього устрою компонента або системи.

2. Тест-дизайн, заснований на техніці чорного ящика – процедура написання або вибору тест-кейсів на основі аналізу функціональної або нефункціональної специфікації компонента або системи без знання її внутрішнього устрою.

Метою цієї техніки є пошук помилок в наступних категоріях:

- неправильно реалізовані або відсутні функції;
- помилки інтерфейсу;
- помилки в структурах даних або організації доступу до зовнішніх БД;
- помилки поведінки або недостатня продуктивність системи.

Техніки тест-дизайну, засновані на використання чорного ящика, включають:

- класи еквівалентності;
- аналіз граничних значень;
- таблиці рішень;
- діаграми зміни стану;
- тестування всіх пар.

Переваги:

- тестування проводиться з позиції кінцевого користувача і може допомогти виявити неточності і протиріччя в специфікації;
- тестувальнику немає необхідності знати мови програмування і заглиблюватися в особливості реалізації програми;

- тестування може проводитися фахівцями, незалежними від відділу розробки, допомагає уникнути упередженого ставлення;
- можна починати писати тест-кейси, як тільки готова специфікація.

Недоліки:

- тестується тільки дуже обмежена кількість шляхів виконання програми;
- без чіткої специфікації досить важко скласти ефективні тест-кейси;
- деякі тести можуть виявитися надмірними, якщо вони вже були проведені розробником на рівні модульного тестування.

4.1.2 Тестування методом білого ящика

Тестування методом білого ящика [24] – метод тестування програмного забезпечення, який передбачає, що внутрішня структура / пристрій / реалізація системи відомі тестувальнику. Ми вибираємо вхідні значення, грунтуючись на знанні коду, який буде їх обробляти. Точно так само ми знаємо, яким повинен бути результат цієї обробки. Знання всіх особливостей програми, яка тестується і її реалізації – обов'язкові для цієї техніки. Тестування білого ящика – поглиблення під внутрішній устрій системи, за межі її зовнішніх інтерфейсів.

Тестування білого ящика – це:

1. Тестування, засноване на аналізі внутрішньої структури компонента або системи.

2. Тест-дизайн, заснований на техніці білого ящика - процедура написання або вибору тест-кейсів на основі аналізу внутрішнього устрою системи або компонента.

Переваги:

- тестування може проводитися на ранніх етапах: немає необхідності чекати створення призначеного для користувача інтерфейсу;
- можна провести більш ретельне тестування з покриттям великої кількості шляхів виконання програми.

Недоліки:

- для виконання тестування білого ящика необхідна велика кількість спеціальних знань;
- при використанні автоматизації тестування на цьому рівні, підтримка тестових скриптів може виявитися вельми накладної, якщо програма часто змінюється.

4.2 Тестування за допомогою використання емулятора та симулятора

Емулятор [25] – це оригінальна заміна пристрою. У вас немає можливості модифікувати програми і додатки, але ви можете їх запускати. Симулятор, в свою чергу, не копіює апаратне забезпечення пристрою, однак у вас є можливість налаштувати аналогічне середовище, таке як в ОС оригінального пристрою.

Емулятори - це простий спосіб протестувати додаток на мобільному телефоні, не використовуючи його фізично. Доступні інструменти для тестування програм операційної системи Androi:

1. Google Android Emulator. Android Емулятор запускається на Windows як окремий застосунок без необхідності повністю завантажувати і встановлювати Android SDK.
2. Офіційний Android SDK Emulator. Включає в себе емулятор мобільного пристрою, який реалізує всі апаратні та програмні особливості типового пристрою.
3. MobiOne. MobiOne Developer — це mobile Web IDE для Windows, допомагає розробнику програмувати, тестувати, налагоджувати, упаковувати і впроваджувати мобільні веб-застосунки на пристрої, такі як iPhone, BlackBerry, пристрої на Android і Palm Pre.

Для платформи Android найпопулярнішими емуляторами є: Bluestacks, Oracle VM VirtualBox, Android SDK Emulator, GenyMotion.

Емулятор BlueStacks створений спеціально для Андроїд під операційну систему Windows. Він дозволяє користувачам запустити на комп'ютері різні мобільні ігри. Основними перевагами програми є:

- можливість роботи в повноекранному режимі;
- установка необмеженої кількості андроїд-додатків в оболонку;
- просте використання і привабливий інтерфейс;
- запуск дуже складних додатків;
- робота відразу з декількома необхідними пристроями;
- можливість управління android-додатками за допомогою клавіатури;
- можливість грати в онлайн-ігри.

Серед мінусів можна виділити:

- можлива проблема при запуску через конфлікт з антивірусом;
- некоректна робота в Windows x64;
- складність роботи з онлайн-додатками.

Oracle VM VirtualBox - потужна безкоштовна система віртуалізації для створення ізольованих віртуальних машин з різними операційними системами, корпоративним і домашнім користувачем. Серед переваг можна виділити такі:

- можливість встановити відразу декілька віртуальних систем Android на ПК;
- великий функціонал;
- стабільна робота на різних версіях Windows.

До мінусів можна віднести:

- велике споживання ресурсів комп'ютера;
- складність в роботі серед звичайних користувачів.

Однією з переваг Android SDK Emulator при тестуванні пристроїв є можливість задавати довільні значення розширення і щільності пікселів екрана. Також емулятор має й інші переваги:

- можливість протестувати додаток різними версіями ОС Android, на пристроях з різними розширеннями дисплея;

- різні налаштування, необхідні для тестування, такі як зміна орієнтації екрану;
- емуляція SD-карти.

З недоліків можна виділити такі:

- досить значний проміжок часу між натисканням кнопки «Run» і запуском додатку на емуляторі;
- робота емулятора дуже повільна.

Genymotion дозволяє запускати програми і ігри від системи Android та тестувати їх на комп'ютері. Утиліта зберігає в собі великий каталог зі смартфонами, список пристроїв включає як нові моделі, так і дещо застарілі. Кожен віртуальний пристрій можна тестувати і налаштовувати. Переваги наступні:

- швидка установка і налаштування середовища VirtualBox;
- присутній режим повноекранного відображення із змінними параметрами дозволу;
- оптимізація з ОС Windows 10 і 8 (в т.ч. на 64 bit), а також сучасними Linux і Mac;
- здатність задавати параметри в RAM налаштуваннях;
- емуляція ADB (засоби налагодження Android Debug Bridge).

Серед недоліків можна зазначити:

- тільки англійський інтерфейс, розбереться не кожен користувач;
- безкоштовно Genymotion позиціонується для домашнього використання, потрібно придбати повну версію програми;
- перед завантаженням софта буде потрібно зареєструвати аккаунт і ввести логін, пароль в рамках офіційного сайту;
- не функціонує з ПК на Windows XP і Vista, немає підтримки процесорів AMD;
- у платній версії більше доступних опцій і розумних налаштувань.

Проаналізувавши переваги та недоліки емуляторів, було вирішено використати Android SDK Emulator, так як він найкраще підходить для тестування додатків на ОС Windows x64 [25].

4.3 Тестування розробленого мобільного додатку

Для тестування автоматизованої системи ведення власних фінансів використано функціональний вид тестування на Android SDK Emulator. При даному виборі текст програми недоступний, і програма розглядається як «чорний ящик». Мета тестування – визначення коректності виконання функцій.

Після запуску мобільного додатку з'являється вікно авторизації, в якому користувачу потрібно ввести логін і пароль. Якщо користувач не зареєстрований, йому необхідно перейти на вікно реєстрації (рис. 4.1). Якщо реєстрація пройшла успішно, то на екрані повинно з'явитися повідомлення про це (рис. 4.2).

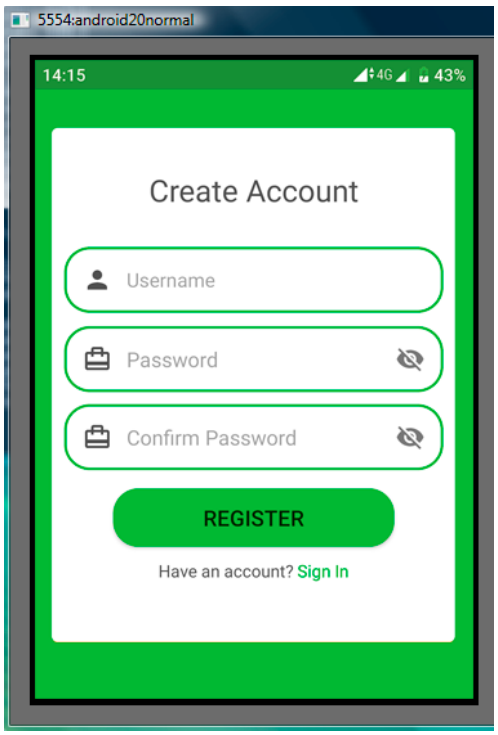


Рисунок 4.1 – Вікно реєстрації

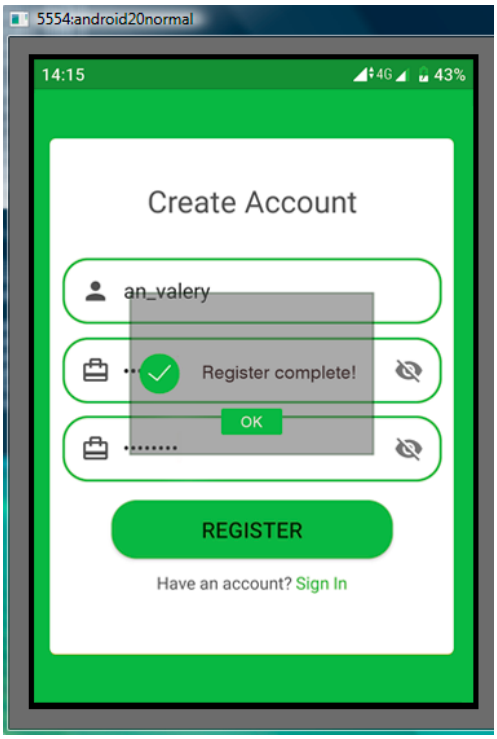


Рисунок 4.2 – Реєстрацію завершено

При неправильному введенні даних зареєстрованого користувача, буде червоним текстом видано помилку про некоректність (рис. 4.3).

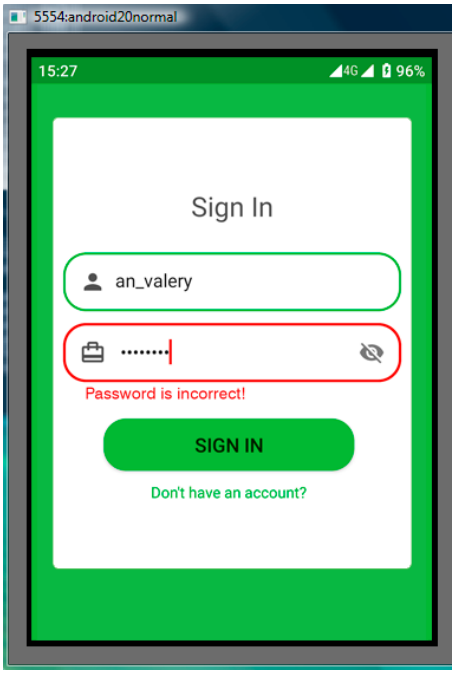


Рисунок 4.3 – Авторизація користувача

Якщо ж дані введені вірно, то далі програма переходить до основного вікна додатку, де відображені надходження, витрати та баланс, а також «+» для того, щоб додати витрату/ надходження і «х» для вилучення операції (рис. 4.4).

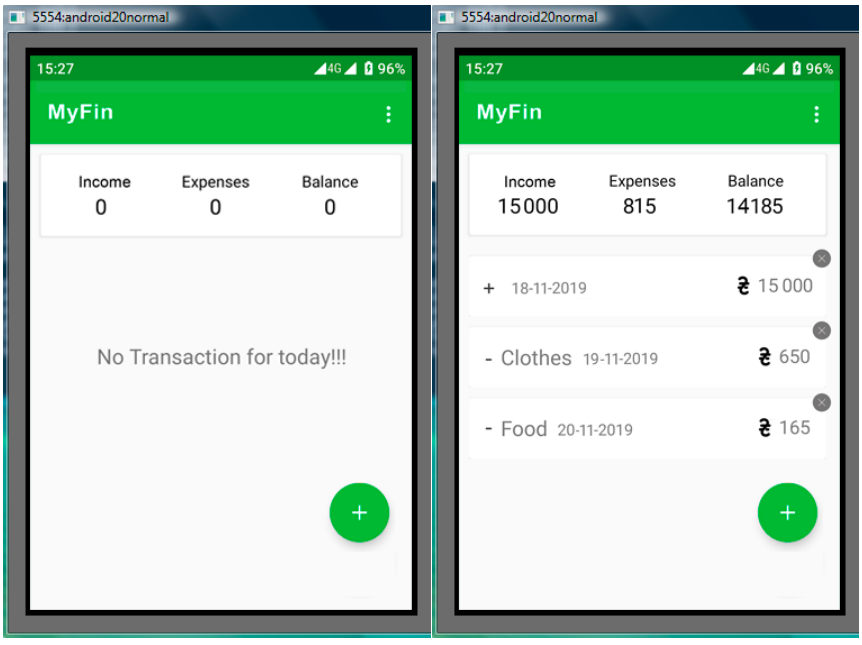


Рисунок 4.4 – Головне вікно додатку

Щоб додати надходження, потрібно ввести суму, обрати дату та за бажанням додати надходження (рис. 4.5).

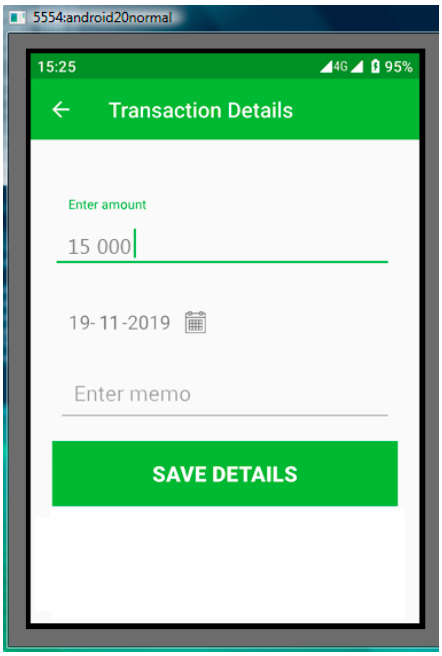


Рисунок 4.5 – Додавання надходжень

Результати тестування підтвердили коректність роботи мобільної мистеми ведення власниз фінансів.

4.4 Висновки

У розділі проаналізовано існуючі методи тестування мобільних додатків, проведено порівняльний аналіз популярних емуляторів для тестування Android-додатків, обрано Android SDK Emulator у якості тестуючого пристрою. Тестування автоматизованої системи проведено функціональним методом, тобто текст програми не доступний, вона розглядається як «чорний ящик».

Тестування підтвердило ефективність та правильність функціонування розробленого автоматизованої системи ведення власних фінансів. Програмний продукт простий у використанні, має привабливий та інтуїтивно зрозумілий інтерфейс.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть Войтко В. В. та Крилик Л.В.

Здійснюємо оцінювання комерційного потенціалу розробки за 12-ма критеріями за 5-ти бальною шкалою.

Результати оцінювання комерційного потенціалу розробки наведено в табл. 5.1.

Таблиця 5.1 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Войтко В. В.	2. Крилик Л.В
	Бали, виставлені експертами:	
1	4	4
2	4	3
3	3	4
4	4	3
5	3	4
6	4	4
7	3	3
8	4	4
9	4	4
10	4	3
11	3	4
12	3	4
Сума балів	$СБ_1 = 44$	$СБ_2 = 44$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 44$	

Отже, з отриманих даних таблиці 5.1 видно, що нова розробка має високий рівень комерційного потенціалу.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні такі витрати.

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \quad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р$ = 22 дні;

t - число днів роботи розробника, t = 45 днів.

Розрахунки заробітних плат для керівника і програміста наведені в табл.5.2.

Таблиця 5.2 – Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за робочий день, грн.	Число днів роботи, t	Витрати на оплату праці, грн.
Науковий керівник	5500	250	6	1500
Інженер-програміст	4000	181,81	45	8181,45
Всього:				9681,45

Розрахуємо додаткову заробітну плату:

$$Здод = 0{,}1 \cdot 9681{,}45 = 968{,}14 \text{ (грн.)}$$

Нарахування на заробітну плату операторів НЗП розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати (5.2):

$$Нзп = (Зо + Зр) \cdot \frac{\beta}{100}, \quad (5.2)$$

$$Нзп = (9681{,}45 + 968{,}14) \cdot \frac{36{,}3}{100} = 3865{,}80 \text{ (грн.)}.$$

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою (5.3):

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12}, \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Розрахунки амортизаційних відрахувань наведені в табл.5. 3.

Таблиця 5.3 – Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Персональний комп'ютер	10000	25	3	625
Всього:				625

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою (5.4):

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо Кі = 1,1).

Витрати на комплектуючі, що були використані для розробки ПЗ, зведено в табл. 5.4.

Таблиця 5.4 - Витрати на комплектуючі, що були використані для розробки ПЗ.

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Флешка	шт.	180	1	180
Пачка паперу	уп.	115	1	115
Ручка	шт.	5	1	5
Всього з урахуванням транспортних витрат				330

Витрати на силову електроенергію розраховуються за формулою (5.5):

$$В_е = В \cdot П \cdot Ф \cdot К_п ; \quad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=1,7 грн/кВт);

П – установлена потужність комп'ютера (П=0,6кВт);

Ф – фактична кількість годин роботи комп'ютера (Ф=200 год.);

$К_п$ – коефіцієнт використання потужності ($К_п < 1$, $К_п = 0,7$).

$$Ве = 1,7 \cdot 0,6 \cdot 200 \cdot 0,7 = 142,8 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто (5.6):

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 * (9681,45 + 968,14) = 10649,59 \text{ (грн).}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + І_в$$

В = 9681,45+968,14+3865,80+625+330+142,8+10649,59= 26262,78 (грн.)

Розрахуємо загальну вартість наукової роботи $В_{заг}$ за формулою (5.7):

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{26262,78}{1} = 26262,78$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою (5.8):

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{26262,78}{0,9} = 29180,86 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою (5.9):

$$\Delta П_i = \sum_1^n (\Delta П_я \cdot N + П_я \Delta N)_i \quad (5.9)$$

де $\Delta П_я$ – покращення основного якісного показника від впровадження результатів розробки у даному році;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

Δ N – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_я$ – основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

У результаті впровадження результатів наукової розробки витрати на виготовлення програмного продукту зменшаться на 20 грн (що автоматично спричинить збільшення чистого прибутку підприємства на 20 грн), а кількість користувачів, які будуть користуватись збільшиться: протягом першого року – на 200 користувачів, протягом другого року – на 175 користувачів, протягом третього року – 125 користувачів. Реалізація програмного продукту до впровадження результатів наукової розробки складала 600 користувачів, а прибуток, що отримував розробник до впровадження результатів наукової розробки – 300 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ протягом першого року складатиме:

$$\Delta П_1 = 20 \cdot 600 + (300 + 20) \cdot 200 = 76000 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = 20 \cdot 600 + (300 + 20) \cdot (200 + 175) = 132000 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = 20 \cdot 600 + (300 + 20) \cdot (200 + 175 + 125) = 172000 \text{ грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $E_{абс}$ вкладених інвестицій розраховується за формулою (5.10):

$$E_{абс} = (ПП - PV), \qquad (5.10)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд рис. 5.1.

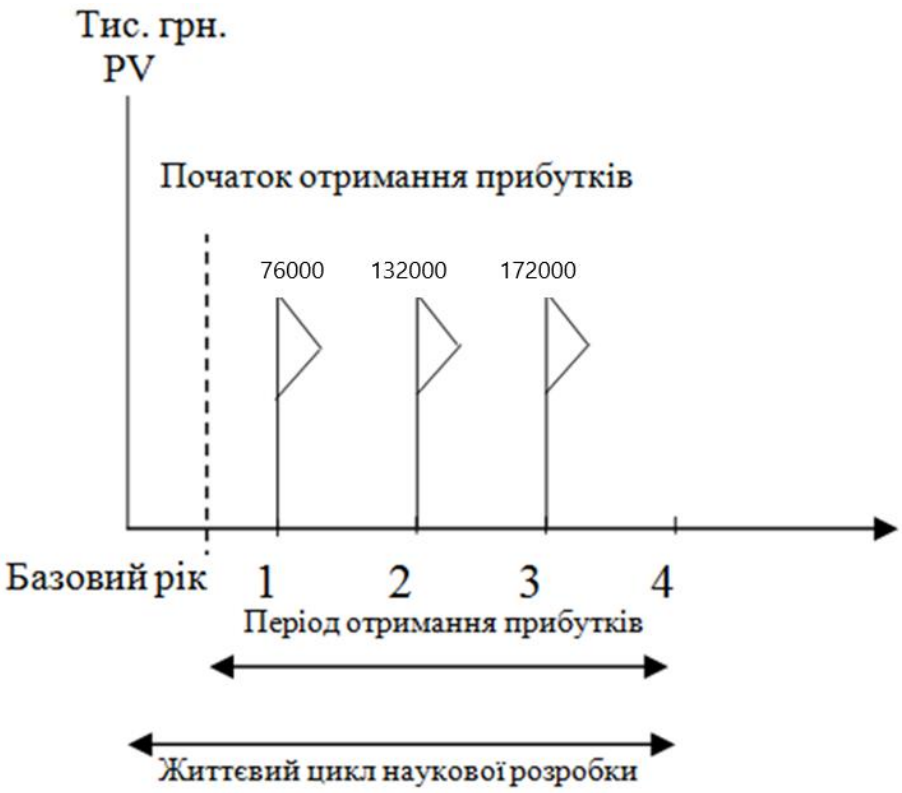


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою (5.11):

$$ПП = \sum_1^m \frac{\Delta П_i}{(1+\tau)^t} \quad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{29180,86}{(1+0,1)^0} + \frac{76000}{(1+0,1)^2} + \frac{132000}{(1+0,1)^3} + \frac{172000}{(1+0,1)^4} = 308642,63 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 308642,63 - 29180,86 = 279461,77 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою (5.12):

$$Е_в = \sqrt[Т]{1 + \frac{Е_{абс}}{PV}} - 1 \quad (5.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{279466,71}{29180,86}} - 1 = 1,19 \text{ або } 119\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f,$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0,2 + 0,1 = 0,3$$

Оскільки $E_в$ = 119% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в}$$

$$T_{ок} = \frac{1}{1,19} = 0,84 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки

У п’ятому розділі було проведено оцінювання комерційного потенціалу розробки, виконано технологічний аудит для оцінювання комерційного потенціалу розробки. Залучено 2-х незалежних експертів. Спрогнозовано витрати на виконання науково-дослідної роботи та конструкторсько-технологічної роботи.

На вісі часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР показано життєвий цикл наукової розробки, початок отримання прибутків та його період.

Вартість чистого прибутку складає 308642,63 (грн.). Абсолютна ефективність вкладених інвестицій : 279461,77 грн. Відносна ефективність вкладених інвестицій : 1 , 19 або 119 %. Термін окупності: 0,84 року . Обрахувавши вартість чистого прибутку, абсолютну і відносну ефективність вкладених інвестицій та термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

ВИСНОВКИ

У магістерскій кваліфікаційній роботі удосконаленно метод автоматизованого ведення власних фінансів для коректної роботи мобільного додатку. Розроблено модель автоматизованої системи ведення власного бюджету, яка використовує технології моніторингу фінансового стану, орієнтованих на нові алгоритми обліку фінансів.

На основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано нову організацію бази даних, яка базується на конвеєрному принципі обробки даних.

Досліджено актуальність даної розробки. Було проаналізовано стан даної проблеми на сьогоднішній день. Розглянуто основні аналоги, визначено їх особливості та недоліки і розроблено порівняння з власним програмним продуктом та методом.

Проведено варіантний аналіз засобів реалізації автоматизованої системи і обґрунтовано вибір середовища програмування Android Studio та мови програмування Java.

Були проаналізовані принципи розробки мобільних додатків і обрано платформу Android для розробки автоматизованої системи, а також був проведений аналіз особливостей розробки бази даних.

Розширено можливості моніторингу та аналізу власного бюджету за рахунок використання розробленої автоматизованої системи ведення власних фінансів під Android систему, що дозволяє підвищити економічну самодисципліну та сприяє заощадженню коштів

У процесі досліджень було використано теорію алгоритмів для побудови алгоритму авторизації користувача, теорії баз даних для створення бази даних автоматизованої системи, методи проектування програмного продукту для розробки мобільного додатку, методи комп’ютерного моделювання для аналізу і перевірки достовірності отриманих теоретичних результатів.

Подальшого розвитку набули моделі автоматизованої системи ведення власного бюджету, які відрізняються від існуючих використанням технологій моніторингу фінансового стану, орієнтованих на нові алгоритми обліку фінансів, що забезпечує підвищення ефективності процесів контролю і управління особистим бюджетом.

Запропоновано нову організацію бази даних, яка, на відміну від існуючих, базується на конвеєрному принципі обробки даних, що підвищує продуктивність програмного продукту.

Проведене тестування довело повну ефективність і правильність функціонування розробленої автоматизованої системи ведення власних фінансів. Проведено оцінювання комерційного потенціалу розробки та проведено розрахунки, які підтверджують економічну доцільність розробки програмного продукту.

Продукт вирішує проблему ведення обліку власних витрат і доходів. Програма полегшує процес контролю за фінансовим станом користувача, так як в будь-який момент можна зайти і перевірити свій баланс та витрати.

Розроблений додаток сприятиме ефективному веденню фінансового обліку власних фінансів.

За результатами дослідження було зроблено дві наукові публікації. Результати доповідались на двох конференціяї та здобули перемогу в 10 міжнародних конкурсах.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Особисті фінанси - інструменти управління. [Електронний ресурс]. – Режим доступу: http://vseprogroshi.com.ua/osobisti-finansi-yak-upravlyati-i-yaki-instrumenti-vikoristovuvati.html. - Назва з екрану.

2. Войтко В.В. Мобільний додаток для атоматизованого ведення власних фінансів / В.В. Войтко, С. В. Бевз, С.М. Бурбело, А.В. Денисюк, А.В. Волошина // Електронні інформаційні ресурси: створення, використання, доступ: Збірник матеріалів Міжнародної науково-практичної Інтернет-конференції. – Вінниця: ВНТУ, листопад 2019 – С. 48-53.

3. Волошина А. В. Розробка засобів реалізації web-сайту навчального закладу/XLVII Міжнародна науково-технічна конференція ВНТУ – 2018 [електронний ресурс] // Режим доступу: https://conf.vntu.edu.ua/index.php/all-fitki/all-fitki-2018/paper/view/5196 – Назва з екрану.

4. Як правильно вести особисті фінанси. [Електронний ресурс]. – Режим доступу: https://marketer.ua/ua/how-to-keep-personal-finances/. - Назва з екрану.

5. Мобільний додаток. [Електронний ресурс]. – Режим доступу: https://www.quality-assurance-group.com/mobilnyj-dodatok-vnosymo-rozuminnya-u-znachennya-terminu/ - Назва з екрану.

6. Мобільні операційні системи. [Електронний ресурс]. – Режим доступу: https://www.quality-assurance-group.com/mobilni-os/. - Назва з екрану.

7. Нараян Прасти. Блокчейн. Разработка приложений. – СПб. : БХВ-Петербург, 2016. – 256 с.

8. Monefy: удобный учет расходов. – [Електронний ресурс]. – Режим доступу: https://rskrf.ru/goods/monefy-udobnyy-uchet-raskhodov/. – Назва з екрану.

9. Дзен-мани. – [Електронний ресурс]. – Режим доступу: https://zenmoney.ru/ – Назва з екрану.

10. Android Studio. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Android_Studio – Назва з екрану.

11. IntelliJ IDEA. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/IntelliJ_IDEA – Назва з екрану.

12. Eclipse. [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Eclipse – Назва з екрану.

13. Intel XDK. [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Intel_XDK – Назва з екрану.

14. Типы мобильных приложений. [Електронний ресурс]. – Режим доступу: https://training.qatestlab.com/blog/technical-articles/types-of-mobile-applications/ – Назва з екрану.

15. Блинов, Романчик. Java. Методы программирования.– СПб. : Минск, 2013. – 897 с.

16. Ефективні методи розробки додатків. [Електронний ресурс]. – Режим доступу: http://android.mobile-review.com/articles/22580/ – Назва з екрану.

17. Features Of SQLite. [Електронний ресурс]. – Режим доступу: https://www.sqlite.org/features.html – Назва з екрану.

18. Інтерфейс користувача [Електронний ресурс]. – Режим доступу: https://uk.wikipedia.org/wiki/Інтерфейс_користувача – Назва з екрану.

19. Нормалізація і нормальні форми [Електронний ресурс]. – Режим доступу: https://stud.com.ua/77226/informatika/normalizatsiya_normalni_formi – Назва з екрану.

20. Алгоритми. [Електронний ресурс]. – Режим доступу: http://ru.wikipedia.org/wiki/Алгоритм – Назва з екрану.

21. Брайан Харди, Билл Филлипс,Крис Стюарт, Кристин Марсикано. Программирование под Android. 2-е издание (2016).

22. П. Дейтел, Х. Дейтел, А. Уолд. Android для разработчиков. 3-е издание(2016).

23. Layout. [Електронний ресурс]. – Режим loступу: http://developer.alexanderklimov.ru/android/theory/layout.php – Назва з екрану.

24. «Искусство тестирования программ» Гленфорд Майерс\ «The Art of Software Testing», Glenford J. Myers – 272 с.

25. Сэм Канер, Джек Фолк “Тестирование программного обеспечения”. Cem Kaner, Jack L. Falk «Testing computer software» - 538 с.

ДОДАТКИ

Додаток А. Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"____ "_____2019 р.

Технічне завдання
на магістерську кваліфікаційну роботу
«Розробка методу і програмних засобів обробки й проекціюнування веб-контенту з використанням доповненої реальності»
за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ к.т.н., доцент кафедри ПЗ, В.В. Войтко
"____"_____2019 р.
Виконав:
_____студент гр. 1ПІ-18м, А.В. Волошина
"____"_____2019 р.

Вінниця ВНТУ 2019

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методу та програмних засобів автоматизованого ведення власних фінансів під Android систему».

Галузь застосування - системи комп'ютерної графіки.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Мета дослідження – розширення можливостей моніторингу та аналізу власного бюджету за рахунок використання розробленої автоматизованої системи ведення власних фінансів під Android систему, що дозволяє підвищити економічну самодисципліну та сприяє заощадженню коштів.

Призначення роботи – розробка методу та системи автоматизованого ведення власних фінансів для мобільних пристроїв із операційною системою Android та програмних засобів для реалізації мобільного додатку.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Брайан Харди, Билл Филлипс,Крис Стюарт, Кристин Марсикано. Программирование под Android. 2-е издание (2016).
2. П. Дейтел, Х. Дейтел, А. Уолд. Android для разработчиков. 3-е издание(2016).
3. Блинов, Романчик. Java. Методы программирования.– СПб. : Минск, 2013. – 897 с.

5. Технічні вимоги

Мова програмування: Java

Технологія розробки: Android Studio

Браузери (або ОС): Android

6. Конструктивні вимоги.

Мобільний додаток повинен відповідати всім вимогам, повинен бути зручним та зрозумілим у використанні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Техніко-економічне обґрунтування доцільності розробки мобільного додатку для ведення власних фінансів	07.10.2019 – 27.10-2019
2	Розробка модулів автоматизованого ведення власних фінансів під Android	28.10.2019 – 8.11.2019
3	Програмна реалізація мобільного додатку	9.11.2019 – 20.11.2019
4	Тестування роботи мобільного додатку	21.11.2019 – 3.12.2019
5	Економічне обґрунтування розробки програмного продукту	4.12.2019 – 7.12.2019

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком

Додаток Б. Програмний код додатку

Login.java

```
public class LoginFragment extends Fragment {
private EditText etEmail;
private EditText etPassword;
private View formView;
private MainActivity parentActivity;
@Override
public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
	container.removeAllViews();
	super.onCreate(savedInstanceState);
	formView = inflater.inflate(R.layout.activity_login, null);
	parentActivity = (MainActivity) getActivity();
	etEmail = (EditText) formView.findViewById(R.id.email);
	etEmail.setText(parentActivity.getCurrentUserEmail());
	etPassword = (EditText) formView.findViewById(R.id.password);
	etPassword.setOnEditorActionListener(new TextView.OnEditorActionListener() {
		@Override
		public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {
			if (id == EditorInfo.IME_NULL) {
				startLogin();
				return true;
			}
			return false;
		}
	});
	etEmail.setText("  ");
	etPassword.setText("  ");
	formView.findViewById(R.id.sign_in_button).setOnClickListener(new View.OnClickListener() {
		@Override
		public void onClick(View view) {
			startLogin();
		}
	});
	// If there was an error logging in.
	if (getArguments() != null) {
		if (getArguments().getInt("error") == Error.InvalidPassword.ordinal()) {
			etPassword.setError(getString(R.string.error_incorrect_password));
			etPassword.requestFocus();
		} else if (getArguments().getInt("error") == Error.InvalidEmail.ordinal()) {
			etEmail.setError(getString(R.string.error_invalid_email));
			etEmail.requestFocus();
		}
	}
	return formView;
}
private void startLogin() {
	boolean cancel = false;
	View focusView = null;

	// Store values at the time of the login attempt.
	String mEmail = etEmail.getText().toString();
	String mPassword = etPassword.getText().toString();

	// Check for a valid password.
	if (TextUtils.isEmpty(mPassword)) {
		etPassword.setError(getString(R.string.error_field_required));
		focusView = etPassword;
```

```
                cancel = true;
            } else if (mPassword.length() < 4) {
                etPassword.setError(getString(R.string.error_invalid_password));
                focusView = etPassword;
                cancel = true;
            }
            // Check for a valid email address.
            if (TextUtils.isEmpty(mEmail)) {
                etEmail.setError(getString(R.string.error_field_required));
                focusView = etEmail;
                cancel = true;
            } else if (!mEmail.contains("@")) {
                etEmail.setError(getString(R.string.error_invalid_email));
                focusView = etEmail;
                cancel = true;
            }
            if (cancel) {
                // There was an error; don't attempt login and focus the first
                // form field with an error.
                focusView.requestFocus();
            } else {
                parentActivity.setCurrentUserEmail(mEmail);
                parentActivity.setCurrentUserPassword(mPassword);
                parentActivity.attemptLogin();
            }
        }
}
```

Register.java

```
public class RegisterFragment extends Fragment {
    private EditText etEmail;
    private EditText etPassword;
    private EditText etFullName;
    private String email;
    private String password;
    private String fullName;
    private View formView;
    private MainActivity parentActivity;

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container, Bundle savedInstanceState) {
        container.removeAllViews();
        super.onCreate(savedInstanceState);
        formView = inflater.inflate(R.layout.activity_register, null);
        parentActivity = (MainActivity) getActivity();
        etFullName = (EditText) formView.findViewById(R.id.reg_fullname);
        etEmail = (EditText) formView.findViewById(R.id.reg_email);
        etEmail.setText(parentActivity.getCurrentUserEmail());
        etPassword = (EditText) formView.findViewById(R.id.reg_password);
        etPassword.setOnEditorActionListener(new TextView.OnEditorActionListener() {
            @Override
            public boolean onEditorAction(TextView textView, int id, KeyEvent keyEvent) {
                if (id == EditorInfo.IME_NULL) {
                    attemptRegister();
                    return true;
                }
                return false;
            }
        });
        formView.findViewById(R.id.btnRegister).setOnClickListener(new View.OnClickListener() {
            @Override
```

```
                public void onClick(View view) {
                    attemptRegister();
                }
            });
            if (getArguments() != null) {
                etPassword.setError(getString(R.string.error_incorrect_password));
                etPassword.requestFocus();
            }
            return formView;
        }
        private void attemptRegister() {
            boolean cancel = false;
            View focusView = null;
            // Reset errors.
            etEmail.setError(null);
            etPassword.setError(null);
            etFullName.setError(null);
            // Store values at the time of the login attempt.
            email = etEmail.getText().toString();
            password = etPassword.getText().toString();
            fullName = etFullName.getText().toString();
            // Check for a valid password.
            if (TextUtils.isEmpty(password)) {
                etPassword.setError(getString(R.string.error_field_required));
                focusView = etPassword;
                cancel = true;
            } else if (password.length() < 4) {
                etPassword.setError(getString(R.string.error_invalid_password));
                focusView = etPassword;
                cancel = true;
            }
            // Check for a valid email address.
            if (TextUtils.isEmpty(email)) {
                etEmail.setError(getString(R.string.error_field_required));
                focusView = etEmail;
                cancel = true;
            } else if (!email.contains("@")) {
                etEmail.setError(getString(R.string.error_invalid_email));
                focusView = etEmail;
                cancel = true;
            } else if (parentActivity.getUserList().containsKey(email)) {
                etEmail.setError(getString(R.string.error_existing_email));
                focusView = etEmail;
                cancel = true;
            }
            if (TextUtils.isEmpty(fullName)) {
                etEmail.setError(getString(R.string.error_field_required));
                focusView = etFullName;
                cancel = true;
            }
            if (cancel) {
                focusView.requestFocus();
            } else {
                parentActivity.setCurrentUserEmail(email);
                parentActivity.setCurrentUserPassword(password);
                parentActivity.setCurrentUserFullName(fullName);
                parentActivity.attempRegister();
            }
        }
}
```

PersonalFinance.sql

```
IF EXISTS(select * from sys.databases where name='PersonalFinance')
DROP DATABASE PersonalFinance;
CREATE DATABASE PersonalFinance;
USE PersonalFinance;
CREATE TABLE Users
(
ID_User INT IDENTITY(1,1) PRIMARY KEY,
[Login] VARCHAR(32) NULL,
[Password] VARCHAR(32) NUll,

Email VARCHAR(32) NOT NULL,
UserName NVARCHAR(32) NOT NULL
);
CREATE TABLE LFinance_Period
(
ID_Period INT IDENTITY(1,1) PRIMARY KEY,
NamePeriod NVARCHAR(32) NOT NULL
);
CREATE TABLE TargetType
(
ID_TargetType int IDENTITY(1,1) PRIMARY KEY, Name_Type NVARCHAR(32) NOT NULL
);
CREATE TABLE LFinanceTarget
(
ID_Target int IDENTITY(1,1) PRIMARY KEY,
Name_Target NVARCHAR(32) NOT NULL,
Target_amount MONEY,
Startingdate DATE,
ID_TargetType INT NOT NULL,   CONSTRAINT TargetType_ID_TargetType_fk
FOREIGN KEY (ID_TargetType)   REFERENCES TargetType (ID_TargetType),
ID_Period INT NOT NULL, CONSTRAINT LFinance_Period_ID_Period_fk FOREIGN KEY
(ID_Period) REFERENCES LFinance_Period (ID_Period)
);
CREATE TABLE LFinance
(
ID_LFinance INT IDENTITY(1,1) PRIMARY KEY,
Title NVARCHAR(32),
);
CREATE TABLE ExpensesTags
(
ID_Tag INT IDENTITY(1,1) PRIMARY KEY,
TagName NVARCHAR(32) NOT NULL
);
CREATE TABLE IncomeTags
(
ID_Tag INT IDENTITY(1,1) PRIMARY KEY,
TagName NVARCHAR(32) NOT NULL
);
CREATE TABLE Repeat_Type
(
ID_Repeat INT IDENTITY(1,1) PRIMARY KEY,
Name NVARCHAR(32) NOT NULL
);
CREATE TABLE LFinanceTarget_ExpensesTags
(
ID INT IDENTITY(1,1) PRIMARY KEY,
ID_Target INT NULL,      CONSTRAINT LFinanceTarget_ExpensesTags_ID_Target_fk
FOREIGN KEY (ID_Target) REFERENCES LFinanceTarget (ID_Target),
ID_Tag INT NULL, CONSTRAINT LFinanceTarget_ExpensesTags_ID_Tag_fk  FOREIGN
KEY (ID_Tag) REFERENCES ExpensesTags (ID_Tag),
```

```
ID_LFinance INT NULL, CONSTRAINT LFinanceTarget_ExpensesTags_ID_LFinance_fk
FOREIGN KEY (ID_LFinance) REFERENCES LFinance (ID_LFinance)
);
CREATE TABLE User_LFinance
(
ID INT IDENTITY(1,1) PRIMARY KEY,
ID_User INT NOT NULL, CONSTRAINT User_LFinance_ID_User_fk FOREIGN KEY (ID_User) REFERENCES
Users (ID_User),
ID_LFinance INT NOT NULL, CONSTRAINT User_LFinance_ID_LFinance_fk FOREIGN KEY (ID_LFinance)
REFERENCES LFinance (ID_LFinance) );

CREATE TABLE Repeat_Income
(
ID_RepeatIncome INT IDENTITY(1,1) PRIMARY KEY,
ID_User INT NOT NULL, CONSTRAINT Repeat_Income_ID_User_fk FOREIGN KEY (ID_User) REFERENCES
Users (ID_User),
Amount MONEY NOT NULL,
[Description] NVARCHAR(255) NULL,
ID_Repeat INT NOT NULL, CONSTRAINT Repeat_Income_ID_Repeat_fk FOREIGN KEY (ID_Repeat)
REFERENCES Repeat_Type (ID_Repeat),
RepeatStartDate DATE,
EndRepeatDate DATE,
IncomeRepeats INT NULL
);
CREATE TABLE Repeat_Expenses
(
ID_RepeatExpenses INT IDENTITY(1,1) PRIMARY KEY,
 ID_User INT NOT NULL, CONSTRAINT Repeat_Expenses_ID_User_fk
FOREIGN
KEY (ID_User) REFERENCES Users (ID_User),
Amount MONEY NOT NULL,
[Description] NVARCHAR(255) NULL,
 ID_Repeat INT NOT NULL, CONSTRAINT Repeat_Expenses_ID_Repeat_fk
 FOREIGN
KEY (ID_Repeat)            REFERENCES Repeat_Type (ID_Repeat),
RepeatStartDate DATE,
EndRepeatDate DATE,
ExpensesRepeats INT NULL
);
CREATE TABLE Income
(
 ID_Income INT IDENTITY(1,1) PRIMARY KEY,
ID_LFinance INT NOT NULL, CONSTRAINT Income_ID_LFinance_fk
(ID_LFinance) REFERENCES LFinance (ID_LFinance),
 FOREIGN KEY
 ID_RepeatIncome INT NULL, CONSTRAINT Income_ID_RepeatIncome_fk FOREIGN KEY (ID_RepeatIncome)
REFERENCES Repeat_Income (ID_RepeatIncome),
ID_User INT NOT NULL, CONSTRAINT Income_ID_User_fk FOREIGN KEY (ID_User) REFERENCES Users
(ID_User),
Amount MONEY NOT NULL,
[Date] DATE,
[Description] NVARCHAR(255) NULL,
);
CREATE TABLE Expenses
(
ID_Expenses INT IDENTITY(1,1) PRIMARY KEY,
ID_LFinance INT NOT NULL, CONSTRAINT Expenses_ID_LFinance_fk FOREIGN KEY (ID_LFinance)
REFERENCES LFinance (ID_LFinance),
ID_RepeatExpenses INT NULL,     CONSTRAINT Expenses_ID_RepeatIncome_fk
FOREIGN KEY (ID_RepeatExpenses) REFERENCES Repeat_Expenses (ID_RepeatExpenses),
ID_User INT NOT NULL, CONSTRAINT Expenses_ID_User_fk FOREIGN KEY (ID_User) REFERENCES Users
(ID_User),
```

```
Amount MONEY NOT NULL,
[Date] DATE,
[Description] NVARCHAR(255) NULL,
Private_expenses BIT
);
CREATE TABLE Repeat_Income_IncomeTag
(
ID INT IDENTITY(1,1) PRIMARY KEY,
ID_Tag INT NOT NULL, CONSTRAINT Repeat_Income_IncomeTag_ID_Tag_fk FOREIGN
 KEY (ID_Tag) REFERENCES IncomeTags (ID_Tag),
ID_RepeatIncome INT NOT NULL, CONSTRAINT
Repeat_Income_IncomeTag_ID_RepeatIncome_fk
REFERENCES Repeat_Income (ID_RepeatIncome),
FOREIGN KEY (ID_RepeatIncome)
ID_LFinance INT NOT NULL, CONSTRAINT Repeat_Income_IncomeTag_ID_LFinance_fk FOREIGN KEY
(ID_LFinance) REFERENCES LFinance (ID_LFinance)
);
CREATE TABLE Repeat_Expenses_ExpensesTag
(
ID INT IDENTITY(1,1) PRIMARY KEY,
ID_Tag INT NOT NULL, CONSTRAINT Repeat_Expenses_ExpensesTag_ID_Tag_fk FOREIGN KEY (ID_Tag)
REFERENCES ExpensesTags (ID_Tag),
ID_RepeatExpenses INT NOT NULL, CONSTRAINT
Repeat_Expenses_ExpensesTag_ID_RepeatExpenses_fk         FOREIGN KEY
(ID_RepeatExpenses)       REFERENCES Repeat_Expenses (ID_RepeatExpenses),
ID_LFinance INT NOT NULL, CONSTRAINT
Repeat_Expenses_ExpensesTag_ID_LFinance_fk FOREIGN KEY (ID_LFinance) REFERENCES
LFinance (ID_LFinance)
);
CREATE TABLE Expenses_ExpensesTag
(
ID INT IDENTITY(1,1) PRIMARY KEY,
ID_Tag INT NOT NULL, CONSTRAINT Expenses_ExpensesTag_ID_Tag_fk FOREIGN
KEY (ID_Tag)    REFERENCES ExpensesTags (ID_Tag),
ID_Expenses INT NOT NULL, CONSTRAINT Expenses_ExpensesTag_ID_Expenses_fk FOREIGN KEY
(ID_Expenses) REFERENCES Expenses (ID_Expenses),
ID_LFinance INT NOT NULL, CONSTRAINT Expenses_ExpensesTag_ID_LFinance_fk FOREIGN KEY
(ID_LFinance) REFERENCES LFinance (ID_LFinance)
);
CREATE TABLE Income_IncomeTag
(
ID INT IDENTITY(1,1) PRIMARY KEY,
ID_Tag INT NOT NULL, CONSTRAINT Income_IncomeTag_ID_Tag_fk FOREIGN KEY (ID_Tag) REFERENCES
IncomeTags (ID_Tag),
ID_Income INT NOT NULL, CONSTRAINT Income_IncomeTag_ID_Income_fk FOREIGN KEY (ID_Income)
REFERENCES Income (ID_Income),
ID_LFinance INT NOT NULL, CONSTRAINT Income_IncomeTag_ID_LFinance_fk FOREIGN KEY (ID_LFinance)
REFERENCES LFinance (ID_LFinance)
);
```

Додаток В. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доцент кафедри ПЗ_____ В. В. Войтко

Рецензент, к.т.н, доцент кафедри КН_____ Л. В. Крилик

Нормоконтроль, д. т. н., проф. кафедри ПЗ _____ В. В. Войтко

Виконавець, студент групи 1ПІ-18м _____ А. В. Волошина

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Розробка методу та програмних засобів автоматизованого ведення власних фінансів під Android систему

Студент 1ПІ-18м: Волошина А.В.
Науковий керівник: к.т.н., доцент: Войтко В.В.

Мета, об'єкт та предмет дослідження

- **Метою роботи** є розширення можливостей моніторингу та аналізу власного бюджету за рахунок використання розробленої автоматизованої системи ведення власних фінансів під Android систему, що дозволяє підвищити економічну самодисципліну та сприяє заощадженню коштів.
- **Об'єкт дослідження** – процес ведення власних фінансів в середовищі автоматизованої системи мобільного пристрою.
- **Предметом дослідження** є засоби реалізації мобільних додатків призначених для ведення власних фінансів.

НАУКОВА НОВИЗНА

- Подальшого розвитку набули моделі автоматизованої системи ведення власного бюджету, які відрізняються від існуючих використанням технологій моніторингу фінансового стану, орієнтованих на нові алгоритми обліку фінансів, що забезпечує підвищення ефективності процесів контролю і управління особистим бюджетом.
- 2. Запропоновано нову організацію бази даних, яка на відміну від існуючих базується на конвеєрному принципі обробки даних, що підвищує продуктивність програмного продукту.

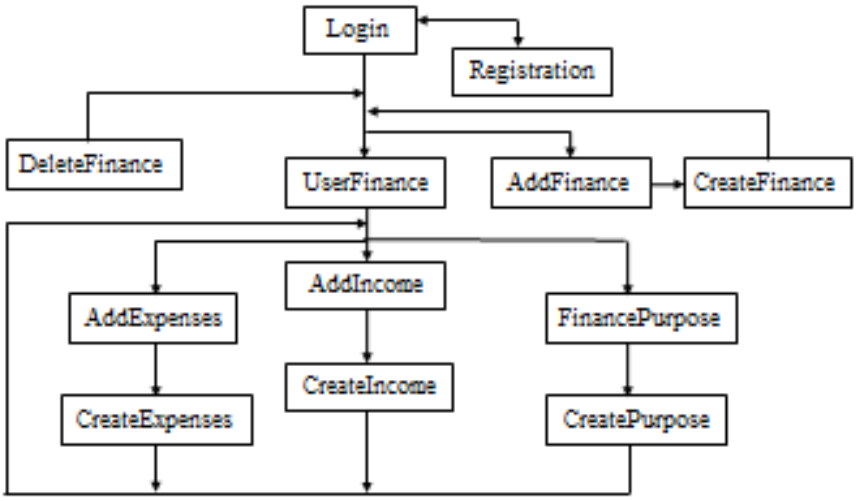
ПОРІВНЯЛЬНИЙ АНАЛІЗ АНАЛОГІВ

Критерій	Monefy: зручний облік витрат	ДзенМані	MyFin
Зручний інтерфейс	+	-	+
Не вимагає доступ до карток чи паролів	+	-	+
Високий рівень безпеки	-	+	+
Повний безкоштовний функціонал	-	-	+

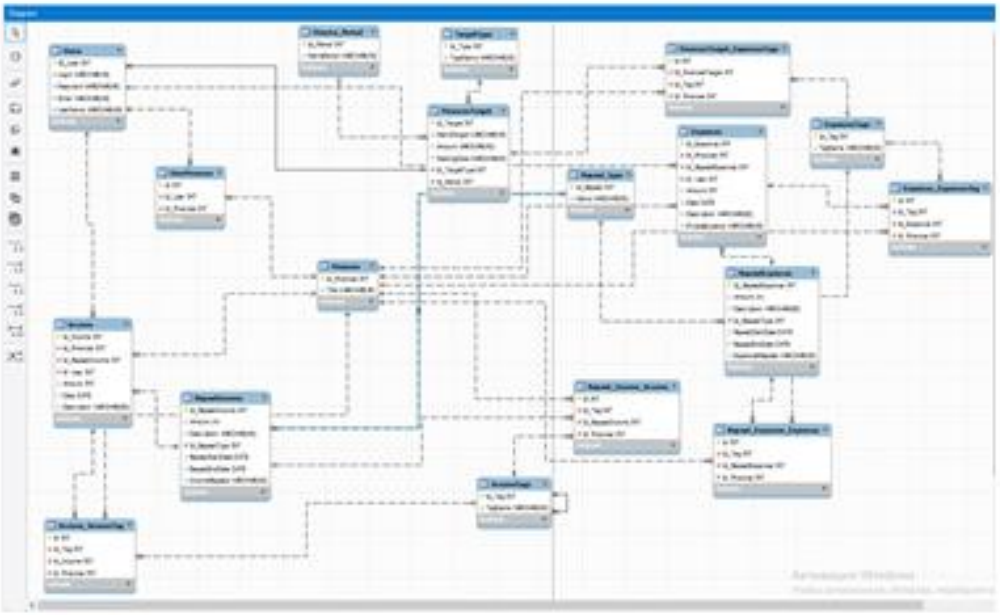
Метод системи автоматизованого ведення власних фінансів

- Створення аккаунту/авторизація.
- В разі правильного введення даних реєстрації сформувати повідомлення про успішну дію/перехід до основного вікна додатку.
- Створення бази витрат та надходжень власних фінансів.
- У разі додавання витрати активувати функцію addTransactionToDatabse.
- У разі отримання надходження активувати функцію addIncomeToDatabse.
- Додати нову інформацію до існуючої.

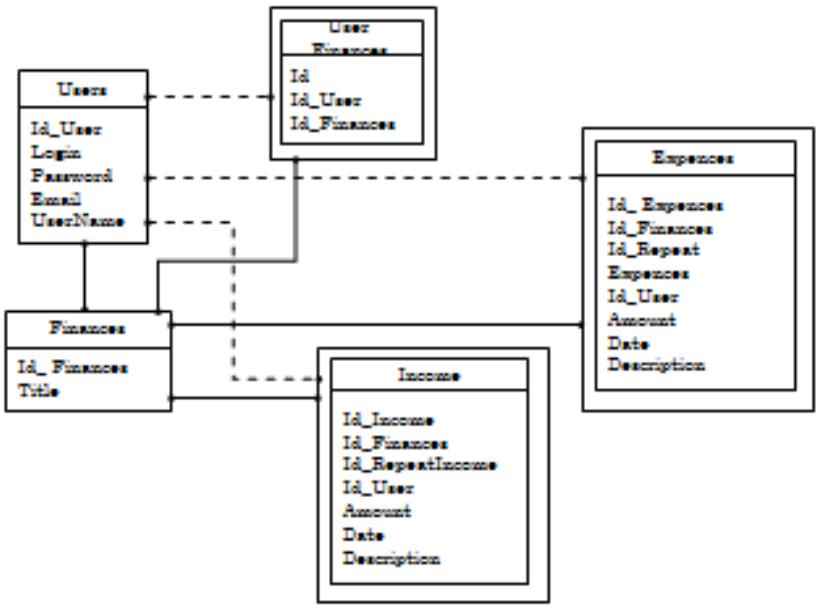
Ієрархічна модель додатку



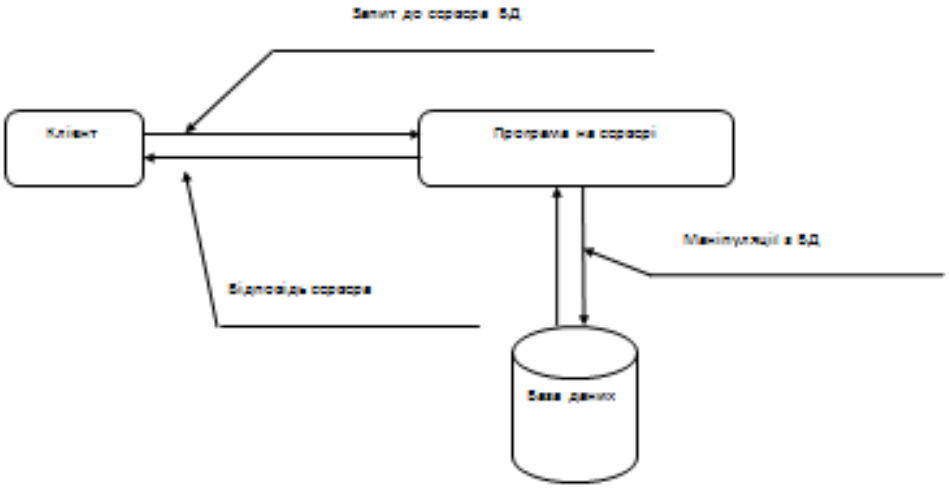
Структура бази даних



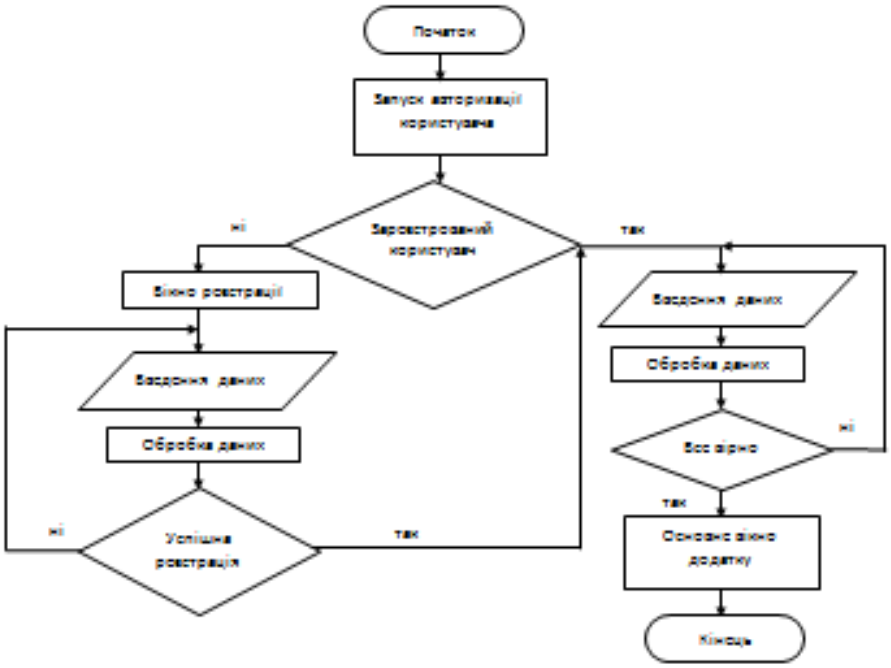
Фрагмент ER-моделі для типу сутності «User»



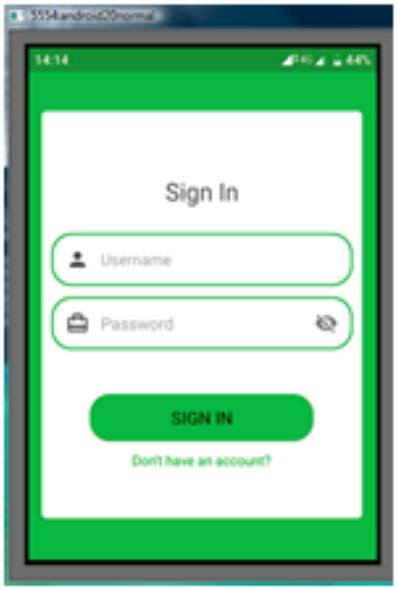
Архітектура мобільного сервісу

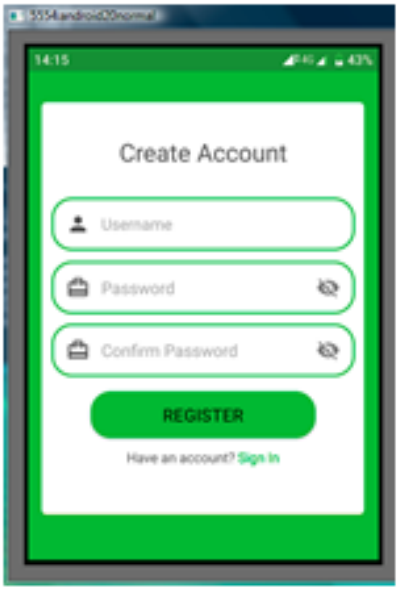


Алгоритм авторизації користувача

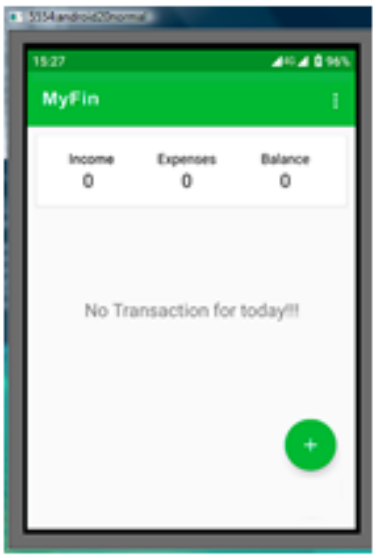


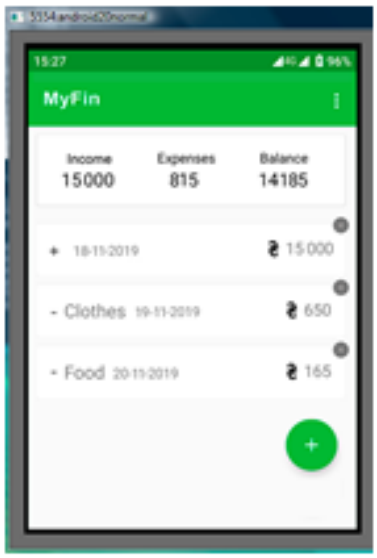
Авторизація/реєстрація користувача





Головне вікно додатку





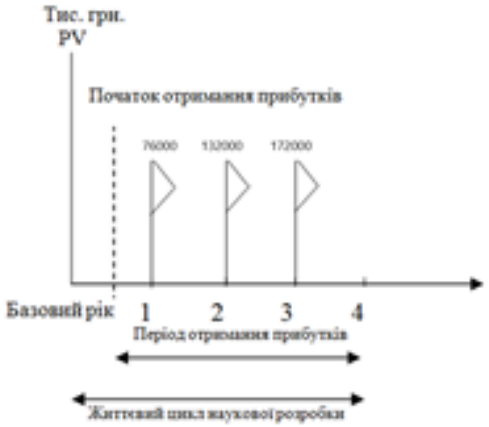
Розрахунок ефективності вкладених інвестицій та період їх окупності

Вартість чистого прибутку: 308642,63 (грн.)

Абсолютна ефективність вкладених інвестицій: 279461,77 грн.

Відносна ефективність вкладених інвестицій: 119 %

Термін окупності: 0,84 року



Висновки

- У магістерській кваліфікаційній роботі удосконаленно метод автоматизованого ведення власних фінансів для коректної роботи мобільного додатку. Розроблено модель автоматизованої системи ведення власного бюджету, яка використовує технології моніторингу фінансового стану, орієнтованих на нові алгоритми обліку фінансів.
- На основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано нову організацію бази даних, яка базується на конвеєрному принципі обробки даних.
- Досліджено актуальність даної розробки. Було проаналізовано стан даної проблеми на сьогоднішній день. Розглянуто основні аналоги, визначено їх особливості та недоліки і розроблено порівняння з власним програмним продуктом та методом.
- Проведено варіантний аналіз засобів реалізації автоматизованої системи і обґрунтовано вибір середовища програмування Android Studio та мови програмування Java.
- Були проаналізовані принципи розробки мобільних додатків і обрано платформу Android для розробки автоматизованої системи, а також був проведений аналіз особливостей розробки бази даних.

- Розширено можливості моніторингу та аналізу власного бюджету за рахунок використання розробленої автоматизованої системи ведення власних фінансів під Android систему, що дозволяє підвищити економічну самодисципліну та сприяє заощадженню коштів
- У процесі досліджень було використано теорію алгоритмів для побудови алгоритму авторизації користувача, теорії баз даних для створення бази даних автоматизованої системи, методи проектування програмного продукту для розробки мобільного додатку, методи комп'ютерного моделювання для аналізу і перевірки достовірності отриманих теоретичних результатів.
- Проведене тестування довело повну ефективність і правильність функціонування розробленої автоматизованої системи ведення власних фінансів. Проведено оцінювання комерційного потенціалу розробки та проведено розрахунки, які підтверджують економічну доцільність розробки програмного продукту.
- За результатами дослідження було зроблено дві наукові публікації. Результати доповідались на двох конференціяї та здобули перемогу в 10 міжнародних конкурсах.

Апробація результатів роботи

- **Апробація матеріалів магістерської кваліфікаційної роботи.** Збірник матеріалів Міжнародної науково-практичної Інтернет-конференції. – Вінниця: ВНТУ, листопад 2019. XLVII Міжнародна науково-технічна конференція 2018.
- **Публікації.** Результати роботи опубліковані в 2 наукових публікаціях: Мобільний додаток для атоматизованого ведення власних фінансів та Розробка засобів реалізації web-сайту навчального закладу.
- Графічні зображення, використані в роботі, подавалися на Міжнародний конкурс з комп'ютерної графіки «Творчість без меж» (Болгарія, м. Хасково) 2019р., де отримано 2 місце (срібну медаль).