ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи

на ступінь вищої освіти бакалавр

на тему: « **РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ З МОНІТОРИНУ ПОВСЯКДЕННИХ ФІНАНСОВИХ ОПЕРАЦІЙ МОВОЮ C#** »

Виконав: студент 4 курсу, групи ПД-44

спеціальності

121 Інженерія програмного забезпечення

(шифр і назва спеціальності/спеціалізації)

Грибінчак Є.В.

(прізвище та ініціали)

Керівник Гребенюк В.В.

Рецензент

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Ступінь вищої освіти – «Бакалавр»

Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного забезпечення
_____Негоденко О. В.
"___" _____2023 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА
ГРИБІНЧАКА ЄВГЕНІЯ ВАЛЕРІЙОВИЧА

(прізвище, ім'я, по батькові)

1. Тема роботи: « Розробка програмного забезпечення з моніторингу повсякденних фінансових операцій мовою C#»

Керівник роботи: Гребенюк В.В. доктор філософії
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «24» лютого 2023 року №26

2. Строк подання студентом роботи « 1 » червня 2023 року

3. Вихідні дані до роботи:

3.1. Аналіз мобільних додатків на ринку;

3.2. Інструментарій та платформа для розробки додатків;

3.3. Створення архітектури проекту;

3.4. Розробка тест-кейсів та плану проекту;

3.5 Технічна та наукова література.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):

4.1. Сегмент мобільних додатків на ринку;

4.2. Аналіз програмного забезпечення для створення мобільного додатку;

4.3. Розробка проекту;

4.4. Реалізація додатку та його подальше тестування;

4.5 Висновки.

5. Перелік демонстраційного матеріалу.

5.1. Основні параметри роботи;

5.2. Актуальність задачі;

5.3. Яка користь від використання додатків для моніторингу особистих фінансів;

5.4. Слабкі сторони наявних додатків;

5.5. Мобільна операційна система Android;

5.6. Xamarin;

5.7. Можливість перенесення коду;

5.8. Метод Activity;

5.9. Патерн MVVM;

5.10. Model;

5.11. View;

5.12. ViewModel;

5.13. Зберігання даних;

5.14. Висновки.

6. Дата видачі завдання: “25” лютого 2023 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1.	Підбір науково-технічної літератури	25.02.23-27.02.23	Виконано
2.	Формулювання вимог до додатку	28.02.23-02.03.23	Виконано
3.	Ознайомлення з фреймворком Xamarin.	3.03-7.03.23	Виконано
4.	Розробка макету додатку.	8.03.23-18.03.23	Виконано
5.	Розробка інтерфейсу додатку.	19.03.23-31.03.23	Виконано
6.	Розробка функціональної складової додатку.	1.04.23-1.05.23	Виконано
7.	Вступ, висновки, реферат	2.05.23-10.05.23	Виконано
8.	Розробка необхідних демонстраційних матеріалів.	11.05.23-20.05.23	Виконано
9.	Попередній захист роботи	21.05.23-31.05.23	Виконано
10.	Здача роботи	1.06.23	Виконано

Студент _____ Грибінчак Є.В.

(підпис) прізвище та ініціали

Керівник роботи _____Гребенюк В.В.

(підпис) прізвище та ініціали

РЕФЕРАТ

Текстова частина бакалаврської роботи 46с., 26 рис., 10 джерел.

Ключові слова: Visual Studio, Xamarin, ОС Android, мобільний додаток.

Об’єкт дослідження - моніторинг повсякденних фінансових операцій

Предмет дослідження – програмне забезпечення з моніторингу повсякденних фінансових операції.

Мета роботи - підтримка процесів моніторингу повсякденних фінансових витрат за рахунок використання програмного забезпечення мовою C#.

Проаналізувавши особливості створення мобільних додатків для Android, а також сучасні можливості реалізації процесів управління фінансами дозволяють нам створити та сформулювати наступні вимоги до програмного продукту.

Мобільний додаток для управління фінансами повинен відповідати наступним вимогам:

- має працювати на всіх версіях ОС Android;
- буде коректно відображатися на екрані будь якого пристрою;
- актуальна інформація про доходи та витрати користувача;
- коректно відображати дані, що зберігаються в базі даних;
- перевіряти наявність нових даних при вході в додаток або оновлені сторінки;
- динамічно відображати дані за категоріями відоповідно до фільтрів встановлених користувачем;
- мати простий та інтуїтивно зрозумілий інтерфейс;

Галузь використання – додаток, який можна використовувати на телефоні або планшеті на базі Android, для користувачів які хочуть відстежувати свої фінансові витрати.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 9
ВСТУП 10
1. СЕГМЕНТ МООБІЛЬНИХ ДОДАТКІВ НА РИНКУ 12
1.1 Розвиток ринку мобільних додатків з часом 12
1.2 Які користі можуть мати користувачі від мобільних додатків для відстеження своїх фінансів 13
1.4 Висновки про аналіз ринку та постановка завдання. 18
2 ІНСТРУМЕНТАРІЙ ТА ПЛАТФОРМА ДЛЯ РОЗРОБКИ ДОДАТКІВ 19
2.1. Мобільна операційна система Android. 19
2.2. Xamarin 20
2.3 Можливість перенесення коду 21
2.4 Activity 24
2.5 Фрагменти 25
2.6 MVVM 27
2.7 Зберігання даних 30
3 ПРОЕКТУВАННЯ ПРОЕКТУ 32
3.1 Розробка діаграм 32
3.2 Макет активностей додатку 36
3.2.1 Головне вікно додатку. Розділ ‘‘Доходи’’. Категорії дозодів. 36
3.2.2 Секція ‘‘Всі доходи’’. Діаграма доходів. Список доходів по категоріям 37
3.2.3 Розділ ‘‘Витрати’’. Категорії витрат. Баланс 38
3.2.4. Всі витрати. Діаграма витрат. Список витрат по категорій 39
4.ФАЗА ТЕСТУВАННЯ ТА РЕАЛІЗАЦІЇ ДОДАТКУ 41
4.1 Тестування додатку 41
4.2 Структура додатку 41
4.3 SQLite 44
4.4 Реалізація інтерфейсу 46
4.4.1 Головне меню 46
4.4.2 Нова операція 47
4.4.3 Випадаюче меню 50
4.4.4 Реалізація діаграм 51
ВИСНОВКИ 53
ДОДАТОК А 57

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

UI – user interface

ОС – операційна система

IDE – Integrated Driver Electronics

БД – база даних

SDK – software development kit

API – Application Programming Interface

ВСТУП

Актуальність темі - У наш час мобільні телефони стали невід'ємною частиною життя сучасної людини. Розвиток мобільних технологій відбувається швидкими темпами, як у сфері функціональності, так і в дизайні. Смартфони використовуються для виконання різних службових завдань, пошуку інформації в Інтернеті та розваг, таких як перегляд фільмів, перегляд фотографій і прослуховування музики. Зв'язок з комп'ютерами став практично безперервним, але це також вносить певні ризики, пов'язані з доступом до небезпечної інформації. Тому розробка мобільних додатків має велику актуальність у сучасному світі.

У світі смартфонів найпопулярнішими операційними системами є Android, iOS і Windows Phone / Windows 10 Mobile. Ця статистика впливає на те, що деякі мобільні додатки розробляються для обох платформ - Android і iOS. Якщо ви вирішите створювати програми окремо для кожної платформи, ви зіткнетеся з низкою викликів, наприклад:

- одаток потрібно адаптувати до вигляду і взаємодії, характерних для конкретної платформи;
- різні способи реалізації певних функцій;
- різне середовище розробки;

Сучасні смартфони та планшети широко використовуються для керування особистими та робочими фінансовими справами. Фінанси є невід'ємною складовою нашого життя, а розумне фінансове управління полегшує наше щоденне існування. Зростаюча тенденція до автоматизації усього, що можна автоматизувати, також охопила цю сферу. Тепер нам не потрібно носити з собою фізичний блокнот, який можна пом'яти, промокнути або загубити, і який легко можуть вкрасти. Мати фінансовий додаток з додатковими функціями планування, сповіщень та статистики в смартфоні - це чудова альтернатива паперовим

записам. Крім того, при здійсненні невеликих витрат, таких як оплата, легко забути про них і не внести в записи в блокнот, тоді як наш телефон завжди під рукою і не потребує окремих зусиль для пошуку ручок та підкладок під блокнот.

Розвиток мобільної індустрії має на меті полегшити наше життя, а смартфони особливо цінні тим, що завжди доступні під рукою. Смартфони відрізняються від комп'ютерів тим, що ми завжди маємо їх під рукою, готові до використання в будь-який момент. Завдяки цьому, ми можемо швидко здійснювати різноманітні завдання, використовувати корисні додатки і мати доступ до інформації в будь-який час і в будь-якому місці. Смартфони стали невід'ємною частиною нашого повсякденного життя, допомагаючи нам управляти фінансами, спілкуватися, організовувати розклад, отримувати новини та виконувати багато інших завдань зручним і швидким способом.

1. СЕГМЕНТ МООБІЛЬНИХ ДОДАТКІВ НА РИНКУ

1.1 Розвиток ринку мобільних додатків з часом

Мобільний додаток - це програма для використання на мобільних пристроях, таких як телефони, смартфони та комунікатори, яка призначена для виконання різних завдань. На ранніх мобільних пристроях було доступно обмежене число додатків, які встановлювалися разом з програмним забезпеченням пристрою. Серед перших мобільних додатків можна виділити телефонний довідник, який був частиною програмного забезпечення апарату і впорядковував контакти користувача.

У 1997 році Nokia 6110 включила вбудовану версію гри "Змійка", яку багато хто вважає першим мобільним додатком. Перший iPod також поставлявся з вбудованими іграми, такими як Пасьянс та Тетріс.

У 1983 році Стів Джобс вже мріяв про App Store або подібний до нього магазин, де програмне забезпечення можна було купити за допомогою телефонної лінії.[1]

До початку 2000-х років мобільний контент загалом і мобільні додатки зокрема почали розвиватися швидкими темпами. З'явлення нових технологій передачі даних через мобільний зв'язок (GPRS, EDGE) знизило вартість мобільного інтернет-трафіку, що послужило спонуканням до зростання ринку мобільних технологій.

У 2000-х роках смартфони та комунікатори почали поступово витісняти звичайні мобільні телефони на ринку мобільних пристроїв стільникового зв'язку. Оскільки вони мали більш широкі можливості та продуктивність, вони відрізнялися від звичайних мобільних телефонів наявністю розробленої операційної системи (Windows Mobile, Symbian OS, RIM, Android, Mac OS), яка була відкритою для розробки програмного забезпечення сторонніми розробниками.[2]

З початку 2000-х років спостерігається швидкий розвиток мобільних пристроїв та додатків, що відбувається паралельно з технологічними досягненнями в цій галузі.

У 2007 році відбулися два важливі події для розвитку кишенькових комп'ютерів і смартфонів. По-перше, на виставці Macworld Conference & Expo був представлений iPhone, який перевернув уявлення користувачів про смартфони. По-друге, в цей же рік було оголошено про Android, що, очевидно, відбулося під впливом стрімко набираючої популярності продукту від Apple. На той момент Android був доступний лише у вигляді бета-версії комплекту для розробників з емулятором.

Рік 2009 можна вважати піком розвитку мобільної операційної системи Android. Виробники мобільної техніки активно зацікавлювалися цією ОС і анонсували свої перші пристрої на її основі, тоді як Google поспішала допрацьовувати її, заповнюючи численні прогалини в дизайні та функціональності.

Можливості розвитку мобільних додатків значно розширилися завдяки розвитку операційних систем для мобільних пристроїв. Розробники отримали доступ до все більшого функціоналу, який надавали операційні системи. Це призвело до зростання попиту на мобільну розробку, що стала популярною серед широкої аудиторії нарівні з комп'ютерною розробкою.

AppAnnie - велика платформа мобільного аналізу підвела підсумки розвитку мобільного ринку у 2015 році. Відзначилось, що у майбутньому веб-сайти не зможуть конкурувати з додатками. Люди з кожним днем все більше витрачають часу та грошей на додатки, що використовують у різних сферах життя. Впровадження таких пристроїв та платформ, як Apple TV, Apple Watch та аналогічні розробки на базі Android, в майбутньому сприятиме тому, що користувачі стануть ще більш "мобільними".

1.2 Які користі можуть мати користувачі від мобільних додатків для відстеження своїх фінансів

Фінанси є невід'ємною частиною нашого життя, а фінансова грамотність допомагає полегшити життя людей. Якісне програмне забезпечення для управління

особистими фінансами може допомогти керувати вашими грошима. Статистика Федеральної резервної системи США свідчить про те, що майже половина дорослого населення США не має можливості здійснювати незаплановані витрати на розмір всього лише чотирьох сотень доларів. Дослідження показує, що майже 25% дорослих, які працюють, покривають свої щомісячні витрати. Ситуація з фінансами не набагато краща і для тих, хто живе в інших частинах світу.[3] Наприклад, згідно з даними національної статистики у Великій Британії середня сума боргів домашніх господарств перевищує 3000 фунтів. Дослідження також показує, що майже кожен п'ятий мешканець Великої Британії вважає свої борги значним навантаженням.

Керування власними фінансами - це серйозна справа, і великі борги можуть мати серйозні наслідки, такі як зіпсований кредитний рейтинг, втрата майна або особисте банкрутство. Однак, вести облік своїх витрат може бути складно, оскільки легко витратити трохи грошей тут і там. І, перш ніж ви зрозумієте, всі ці дрібні витрати складуться в значну суму.

Складання бюджету - це важлива річ, що допоможе вам контролювати витрати і запланувати їх ефективніше. Інструменти планування бюджету можуть також допомогти вам розробити стратегію заощадження та інвестування грошей. Використання інших інструментів фінансового управління також може сприяти збереженню і розумному використанню ваших фінансових ресурсів.

1.3 Дослідження ринку мобільних додатків для керування особистими фінансами

На ринку мобільних додатків для контролю власних кошт наявні конкурентні продукти, такі як "CoinKeeper", "Spendee", "Finkee" та "MoneyWiz". Тому ми розглянемо кожен з цих додатків окремо, щоб визначити їх функції та оцінити їх рівень надання послуг.

Додаток "CoinKeeper" є ідеальним вибором для тих, хто цінує детальний облік своїх витрат та доходів. Програма розділяє гроші на дві основні категорії - готівку та безготівкові кошти. Крім того, вона дозволяє планувати витрати та розподіляти їх за категоріями. Сума запланованих витрат відображається у графі "У планах". Проте, у додатку є й деякі недоліки, такі як неповний переклад, деякі платні функції (наприклад, сімейний бюджет та детальна статистика), обмеження періоду ведення обліку, а також наявність реклами та платних пакетів послуг всередині додатку. Крім того, в "CoinKeeper" немає можливості додавати коментарі до витрат.

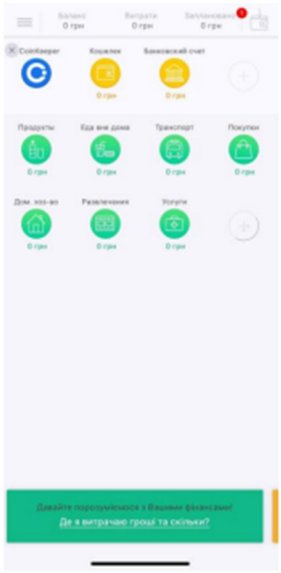


Рисунок 1.1 - Інтерфейс додатку «CoinKeeper»

Додаток "Spendee" має незвичайний дизайн, оскільки всі транзакції відображаються у вигляді стрічки, схожої на соціальні мережі. Крім того, користувач може не просто записувати свої витрати, але також вказувати конкретне місце, де вони були здійснені, включаючи назву супермаркету чи ресторану, а також додати фото цього місця. "Spendee" надає детальну статистику, що подібна до тієї, що є у додатку "CoinKeeper", але ця функція доступна у безкоштовній версії. Однак, додаток має свої недоліки, такі як обмежена кількість категорій, фінансових рахунків і бюджетів у безкоштовній версії, а також відсутність функцій

нагадування про платежі та постановки цілей. Крім того, як і у додатку "CoinKeeper", переклад додатку є неповним.

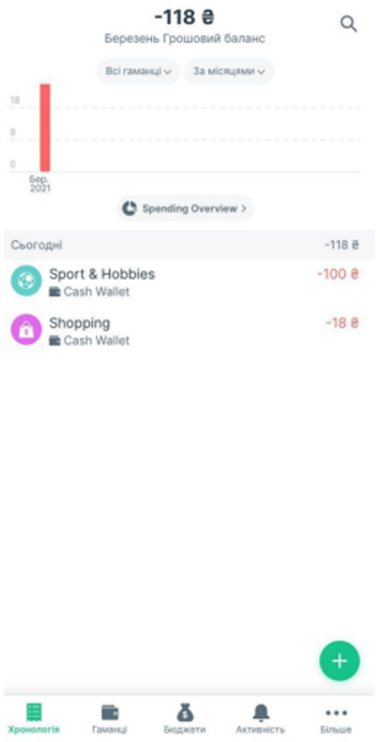


Рисунок 1.2 – Інтерфейс додатку «Spendee»

Додаток "Finkee", розроблений українськими розробниками, недавно з'явився на ринку, пропонуючи широкий спектр функцій, одна з яких не є загальновживаною в подібних програмах - можливість ведення обліку криптовалют. Незважаючи на це, додаток ще не знайшов свою аудиторію. Однак, його головний недолік полягає у відсутності інших мов, окрім англійської. Крім того, введення транзакцій в додаток менш зручне, ніж у "CoinKeeper", і інтерфейс важко адаптувати, оскільки повний баланс відображено маленькими літерами зверху, що може бути складно помітити з першого разу.

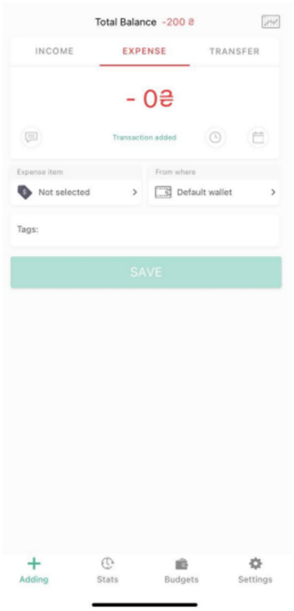


Рисунок 1.3 – Інтерфейс додатку «Finkee»

Додаток "MoneyWiz" має можливість використання на 20 мовах, підтримує всі світові валюти та дозволяє багаторівневе настроювання категорій. У загальній складності додаток містить понад 600 функцій, що робить його одним з найбільш універсальних у даному вибірковому наборі. Єдиний недолік полягає у необхідності підписки для отримання доступу до додатку.

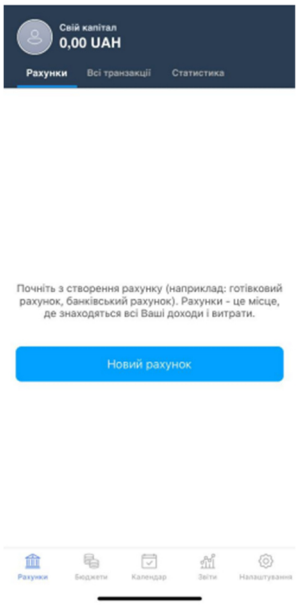


Рисунок 1.4 – Інтерфейс додатку «MoneyWiz»

1.4 Висновки про аналіз ринку та постановка завдання.

Під час аналізу ринку мобільних додатків, що стосуються даної тематики, було виявлено можливості модернізації деяких наявних функцій та додавання нових інструментів для більш гнучкої настройки роботи програми.

Однією з головних концепцій розробки є зручне та легке відображення контенту. Користувач повинен мати можливість швидко та легко знайти необхідні функції з декількох простих рухів пальцями по екрану свого пристрою.

Отже, були визначені певні критерії для основних функцій програми:

- зручний та інтуїтивно зрозумілий інтерфейс;
- передбачається можливість створення записів про витрати та доходи з використанням різних атрибутів, таких як дата, категорія, сума, валюта, коментар і т.д.;
- змінення або вилучення фінансових операцій, які були додані раніше;
- можливість створення, видалення та редагування категорій витрат і доходів;
- можливість перегляду доходів та витрат з вибором діапазону часу;
- забезпечення можливості вибору валюти для фінансових операцій;
- можливість перегляду статистики витрат і доходів у вигляді кругових діаграм.

2 ІНСТРУМЕНТАРІЙ ТА ПЛАТФОРМА ДЛЯ РОЗРОБКИ ДОДАТКІВ

2.1. Мобільна операційна система Android.

Операційна система Android призначена для використання на різноманітних пристроях, включаючи смартфони та планшети. Вона базується на ядрі Linux та використовує багато його основних функцій, таких як віртуальна пам'ять, файлові системи, процеси, ідентифікатори користувачів та планування. Було внесено деякі нові концепції, проте їх кількість є досить обмеженою.[4] Операційна система Android була розроблена альянсом Open Handset Alliance (OHA), який зараз активно займається її підтримкою та розвитком. Довгі роки серед розробників Android популярною мовою програмування була Java, проте в 2017 році Google оголосили Kotlin основною мовою для створення додатків на платформі. Крім цього, розробники мають можливість використовувати інші мови програмування, такі як Delphi, C# тощо. Операційна система Android була встановлена на більш як 83% смартфонів, проданих у 2018 році.

Android відрізняється від інших операційних систем своїм підходом до уніфікації, використанням єдиної схеми управління апаратними компонентами та взаємодії з користувачем. Будучи володарем кількох пристроїв на Android, можна мати на них однаковий набір програм, контактів та календарів без жодних зусиль для користувача.

Однією з ключових переваг Android є його відкритість. Оскільки платформа є вільною для використання, будь-який розробник або виробник мобільних пристроїв може використовувати Android для створення своїх пристроїв, додатків та навіть власної операційної системи. Це завжди є головною перевагою платформи.

Одна з головних переваг операційної системи Android для користувача полягає в можливості вибору з великого списку доступних телефонів. Оскільки ця ОС є відкритою і безкоштовною, конкуренція на ринку мобільних пристроїв є дуже

високою. Тому на Android можна знайти як бюджетні смартфони, так і пристрої елітного класу. Це є надзвичайно важливим для користувачів, які можуть вибирати з широкого асортименту виробників телефонів.

2.2. Xamarin

Xamarin є платформою для створення мобільних додатків, яка дозволяє розробникам створювати нативні додатки для iOS, Android і Windows, використовуючи загальний код C# або .NET. Ця платформа дозволяє багаторазово використовувати від 75% до майже 100% коду між платформами.[5]

Програми, розроблені з використанням Xamarin і C#, мають повний доступ до інтерфейсів API підтримуваних платформ і можливість створення нативних інтерфейсів користувача. Крім того, код компілюється в машинний, що має незначний вплив на продуктивність виконання.

Цей фреймворк складається з наступних компонентів:

- Xamarin.IOS - це бібліотека класів на C#, що дозволяє отримати доступ до iOS SDK;
- Xamarin.Android - це бібліотека класів на C#, яка надає доступ до Android SDK;
- Компілятори для обох ОС;
- Середовища розробки - рідна IDE Xamarin Studio і плагін для Visual Studio.

Xamarin заснований на реалізації платформи .NET або Mono з відкритим вихідним кодом.

Ця технологія містить в собі власний компілятор C#, розробницьке середовище та основний пакет бібліотек .NET. Завдяки цьому, програми, що написані на C#, можуть бути запущені на операційних системах, відмінних від Windows, таких як Unix-системи, Mac OS і інші.

Одна з ключових відмінностей між iOS і Android полягає в способі попередньої компіляції програм. В Android використовується Dalvik, який інтерпретує байт-код,

згенерований в процесі компіляції нативних додатків на мові Java, в команди процесора в момент виконання програми (так звана Just-in-time компіляція). У iOS застосовується модель Ahead-of-Time, де програми компілюються перед виконанням. Xamarin розуміє особливості компіляції на обох платформах і надає окремі компілятори для кожної з них, що дозволяє отримати нативні додатки.

На відміну від Android з його Davlik, для iOS не застосовується проміжний код, а отже, програми мають бути заздалегідь скомпільовані в машинний код. Для цієї мети використовується AOT компілятор Mono.

2.3 Можливість перенесення коду

Xamarin - це інструмент для крос-платформної розробки, що дозволяє запускати додатки, написані один раз, на різних мобільних платформах. Однак, варто зазначити, що для кожної платформи потрібно окремо писати шар UI, незважаючи на загальну логіку.

Якщо розглядати додаток як набір окремих шарів, то можна виокремити наступну структуру:

- Business Layer (BL) - це логіка додатка, яка визначає, як додаток повинен взаємодіяти з даними та виконувати бізнес-правила.

  Service Access Layer (SAL) - це шар, який відповідає за взаємодію з віддаленими сервісами (наприклад, через протокол Json).
- Data Layer (DL) - це шар, який відповідає за збереження даних у сховищі (наприклад, SQLite).
- Data Access Layer (DAL) - це обгортка навколо сховища даних, яка забезпечує можливість виконувати CRUD-операції.
- Application Layer (AL) - це шар, який містить код, залежний від платформи (наприклад, від бібліотеки monotouch.dll або monodroid.dll).
- User Interface Layer (UI) - це шар, який містить інтерфейс користувача.

Всі верстви, за винятком Application Layer і User Interface Layer, є крос-платформовими. Бібліотека Xamarin.Mobile призначена для підвищення рівня платформонезалежності. Вона надає єдиний API для роботи з камерою і геолокацією на різних платформах.[6]

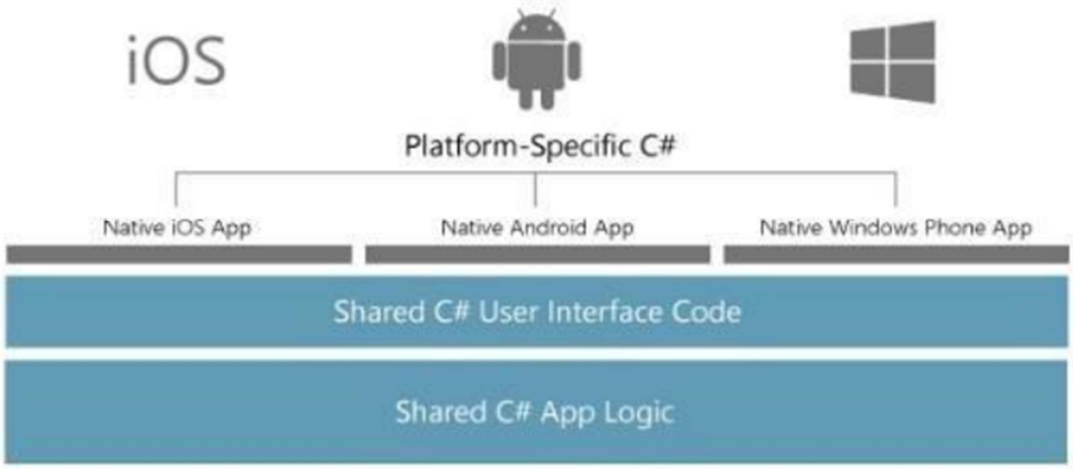


Рисунок 2.1 - Схема роботи кросплатформених додатків на платформі Xamarin

Xamarin Components Store

У середовищі Xamarin присутній магазин компонентів, відомий як Xamarin Components Store. Цей магазин інтегрований в середовище розробки і дозволяє легко та швидко додавати до проекту компоненти від розробників Xamarin і сторонніх розробників, які можуть бути безкоштовними або платними. Компоненти можуть бути двох видів: елементами користувацького інтерфейсу та бібліотеками класів, такими як Json.NET. Деякі з компонентів не є крос-платформенними і пов'язані з певною конкретною платформою. Для зв'язування з нативними бібліотеками Xamarin використовує механізм біндингу, що дозволяє розробникам переносити на C# будь-які нативні бібліотеки класів. Для Xamarin.iOS є спеціальна утиліта, яка автоматично генерує механізми зв'язування з нативними бібліотеками, що дозволяє розробникам швидко використовувати нові функції, такі як Dropbox API.[7]

IDE

Можна скористатися або власною IDE Xamarin Studio, або Visual Studio як середовищем розробки для Xamarin.

Xamarin Studio є крос-платформовою інтегрованою середовище розробки, яка підтримує як Mac OS X, так і Windows. Вона має простий і дружній інтерфейс, але за зовнішнім виглядом приховує потужний інструмент з безліччю функцій, які знайомі користувачам Visual Studio і Resharper.

Xamarin дозволяє розроблювати за допомогою Visual Studio, якщо встановити спеціальний плагін. При роботі з iOS потрібно підключитися до Mac, який буде використовуватися для компіляції проекту і запуску на пристрій або емулятор. Однак, розстановка брейкпоінтів та налагодження відбуваються безпосередньо в Visual Studio. Xamarin Studio також є крос-платформеною IDE, яка працює на Mac OS X та Windows і включає в себе безліч функцій, знайомих з Visual Studio та Resharper.

Існують кілька способів працювати в Visual Studio, зокрема: перший - використання віртуальної машини всередині Mac OS (наприклад, за допомогою Parallels), другий - використання двох різних фізичних машин. Однак використання одного Mac-пристрою для декількох PC-розробників є важким завданням, оскільки процес налагодження вимагає маніпуляцій з симулятором. Щодо третього способу - використання віртуальної машини з Mac OS X (hackintosh), то це непоганий варіант, але має свої обмеження. Наприклад, переміщення по Storyboard в Xcode можливе тільки за допомогою смуг прокручування, оскільки миша з Windows і миша з Mac - це різні речі.

Розробники, які мають досвід роботи з C#, .NET та Visual Studio, можуть скористатися такими ж можливостями та продуктивністю при створенні мобільних додатків з використанням Xamarin. Це включає віддалену налагодження на Android, iOS та Windows пристроях, не потребуючи вивчення нативних мов, таких як Swift або Kotlin. Дивно, але багато високопродуктивних додатків з красивими

користувацькими інтерфейсами, такі як NASCAR, Aviva та MixRadio, були створені з використанням Xamarin.

2.4 Activity

Activity (активність) є однією з ключових компонентів при створенні візуального інтерфейсу мобільного додатку в Xamarin.

Зачастую активність порівнюють з окремим екраном або вікном додатка, тому перемикання між вікнами можна описати як переміщення від однієї активності до іншої. У додатку для Android може бути від однієї до кількох активностей. Усі об'єкти activity є екземплярами класу android.app.Activity, який надає базову функціональність для всіх activity. Для головної активності, яка зазвичай має назву MainActivity, наслідування від класу AppCompatActivity є стандартом, а цей клас, в свою чергу, успадковується від базового класу Activity.

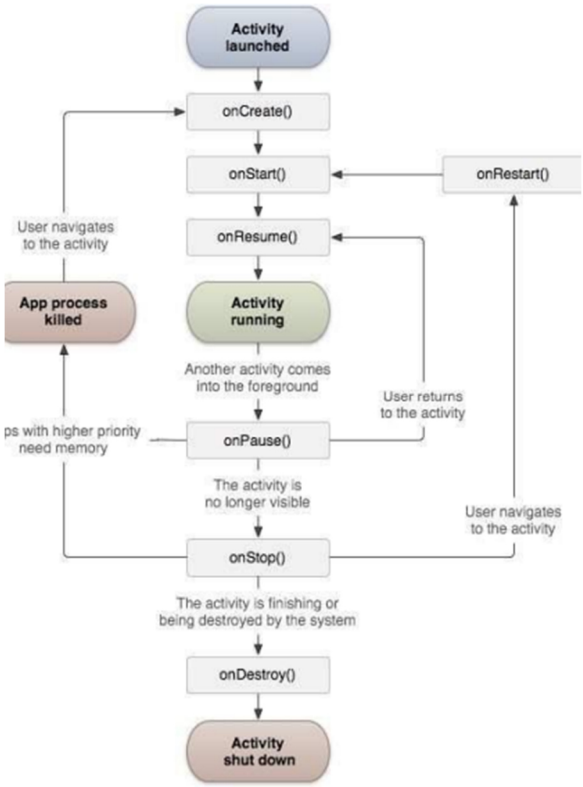


Рисунок 2.2 - Клас Activity та його життєвий цикл

2.5 Фрагменти

Не завжди є оптимальним мати додаток, який складається з кількох активностей. Світ Android має значну фрагментацію і включає в себе різноманітні пристрої з різними технічними характеристиками. Хоча використання декількох активностей може бути ефективним для мобільних пристроїв з середніми екранами, на планшетах та телевізорах зовнішній вигляд інтерфейсу буде помітно поступатися вигляду менших пристроїв. Концепція фрагментів була винайдена саме через цю причину.

Життєвий цикл фрагмента пов'язаний з життєвим циклом активності, і фрагмент не може існувати поза контекстом активності. Кожна активність може містити декілька фрагментів.

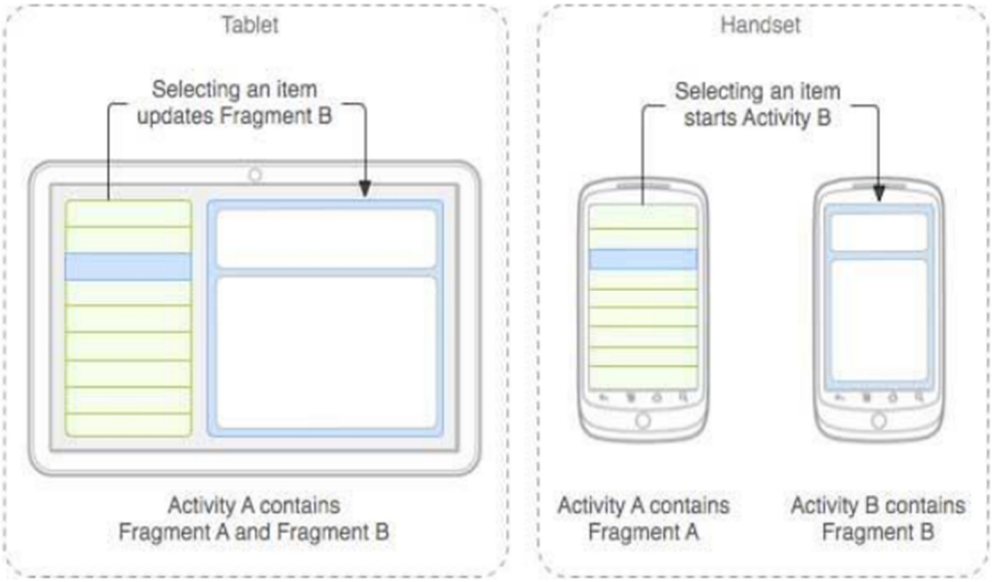


Рисунок 2.3 - Життєвий цикл фрагментів

Класи фрагментів походять від android.app.Fragment. Є кілька підкласів фрагментів, таких як ListFragment, DialogFragment, PreferenceFragment, WebViewFragment та інші.

Для взаємодії між фрагментами використовується android.app.FragmentManager - спеціальний менеджер фрагментів.

Клас android.app.FragmentManager не виконує свою роботу самостійно, а використовує класи-допоміжники. Наприклад, для здійснення транзакцій (додавання, видалення, заміна) використовується клас android.app.FragmentTransaction.

Нижче перераховані деякі назви класів з бібліотеки сумісності:

- android.support.v4.app.FragmentActivity
- android.support.v4.app.Fragment
- android.support.v4.app.FragmentManager
- android.support.v4.app.FragmentTransaction

Для того, щоб система коректно працювала з фрагментами, потрібно спочатку вказати використання класу FragmentActivity (замість стандартного Activity).

Один додаток може використовувати як нові фрагменти, так і фрагменти з бібліотеки сумісності.

Основний порядок взаємодії з фрагментами такий:

Для кожного фрагмента необхідно мати відповідний клас, який успадковується від класу Fragment або його аналогів. Це подібно до створення нової активності або іншого компонента.

Аналогічно до роботи з activity, створюються різні методи, наприклад, onCreate() та інші. Якщо фрагмент має розмітку, то використовується метод onCreateView(), який можна порівняти з методом setContentView() для activity. У методі onCreateView() повертається об'єкт View, який є кореневим елементом розмітки фрагмента.

Можна створити розмітку для фрагмента як програмно, так і декларативно з використанням XML.

Створення розмітки для фрагмента ідентичне створенню розмітки для активності.

Клас, який представляє фрагмент, повинен бути успадкований від класу Fragment.

Для створення візуальної частини фрагменту необхідно перевизначити метод onCreateView() батьківського класу, який приймає три аргументи:

- для створення інтерфейсу використовується об'єкт LayoutInflater для надання ресурсу розмітки;
- параметр container типу ViewGroup визначає контейнер для інтерфейсу;
- передача параметра Bundle savedInstanceState дозволяє відновити попередній стан фрагмента.

Для створення інтерфейсу фрагмента використовується метод inflate() об'єкта LayoutInflater. Цей метод приймає ресурс розмітки layout для поточного фрагмента, контейнер, в який буде додано інтерфейс, та третій булевий параметр, який вказує, чи потрібно прикріплювати розмітку до контейнера з другого параметра.

Так само, як і в activity, тут ми можемо призначати обробники для елементів керування, взаємодіяти з їх подіями. У даному випадку, в текстовому полі виводиться поточна дата.

2.6 MVVM

Переставлення логіки додатка від візуальної частини (подання) можливо за допомогою паттерну MVVM (Model-View-ViewModel). Цей паттерн є архітектурним, оскільки він визначає загальну структуру програми.[9]

Архітектура MVVM включає три складові компоненти: модель (Model), модель представлення (ViewModel) та представлення (View).

Рисунок 2.4 - Компоненти MVVM та їх зв'язок

Model

Модель у MVVM визначає дані, що використовуються в додатку, і може містити логіку, що безпосередньо стосується цих даних. Наприклад, модель може містити логіку для валідації властивостей. Проте модель не повинна містити будь-якої

логіки, що стосується відображення даних чи взаємодії з візуальними елементами управління. Іноді модель може реалізовувати інтерфейси INotifyPropertyChanged або INotifyCollectionChanged, що дозволяють повідомляти систему про зміни властивостей моделі. Це спрощує прив'язку до подання, але взаємодії між моделлю та уявленням немає.

View

View, або представлення, є візуальною складовою, через яку користувач взаємодіє з додатком. У Xamarin.Android представлення визначається в коді XAML та включає в себе елементи, такі як кнопки, текстові поля та інші візуальні елементи.

Хоча клас Window в WPF може містити як інтерфейс в xaml, так і пов'язаний з ним код на C#, проте зазвичай бажано, щоб код на C# містив лише конструктор, який викликає метод InitializeComponent та виконує початкову ініціалізацію вікна. Основна логіка додатку повинна бути відокремлена від інтерфейсу та знаходитись в ViewModel компоненті.

Іноді може зустрічатися ситуація, коли в файлі, пов'язаному з візуальним елементом, може бути вкладена певна логіка, яка не завжди може бути легко реалізована у ViewModel в рамках патерну MVVM.

За винятком рідких випадків, View не обробляє жодних подій, а виконує свої дії за допомогою команд.

ViewModel

ViewModel (або модель уявлення) забезпечує зв'язок між моделлю і відображенням через механізм прив'язки даних. Якщо значення властивостей моделі змінюються, і якщо модель реалізує інтерфейс INotifyPropertyChanged, то дані автоматично оновлюються в відображенні. Це стає можливим завдяки прив'язці даних, хоча модель і відображення не пов'язані безпосередньо.

В моделі уявлення також міститься код для отримання даних з моделі і їх подальшого передавання в уявлення. Крім того, ViewModel відповідає за логіку оновлення даних в моделі.

Так як візуальні компоненти, такі як кнопки, не використовують подій, то взаємодія між View та ViewModel відбувається за допомогою команд.

Припустімо, що користувач бажає зберегти дані, введені в текстове поле. Коли він натискає на відповідну кнопку, команда передається в ViewModel, яка обробляє отримані дані і відповідно до них оновлює модель.

Використання патерну MVVM призводить до розподілу програми на три компоненти, що дозволяє зробити їх розробку та тестування більш простими. Також це сприяє більш ефективній модифікації та підтримці в майбутньому.

При розробці на основі технологій Microsoft легко побачити, що MVVM забезпечує повторне використання.

Для розробки додатків для iOS потрібно враховувати певні особливості. Незважаючи на те, що Xamarin Studio для Mac має всі необхідні інструменти для розробки під iOS, користувачам Visual Studio на ПК все ще потрібно встановлювати Mac з інструментарієм Xamarin. Це дозволяє компілювати програми через мережу та тестувати їх на iOS Simulator або на самому iOS-пристрої.

Xamarin дозволяє розробляти програми для iOS та Android за допомогою C# або F# та шаблону Model-View-Controller. Однак, для поліпшення тестування, супроводу та портування може знадобитися перенесення шаблону MVVM на ці платформи. В цьому випадку використовується фреймворк MvvmCross.

MvvmCross для додатків Xamarin

MvvmCross є кросплатформеною інфраструктурою з відкритим вихідним кодом, яка підтримує додатки для Windows Phone, Windows 8, iOS, Android і WPF. Вона дозволяє використовувати шаблон MVVM на платформах, де він раніше не був підтриманий, таких як iOS і Android.

MvvmCross також дозволяє зв'язувати дані з представленням. Це міцний функціонал, що дозволяє розділяти обов'язки (separation of concerns). Для забезпечення необхідної поведінки в додатку View використовує різні ViewModel. Навіть ViewModel можна легко знайти в окремому проекті MvvmCross, щоб можна було легко посилатися на них та повторно використовувати в інших додатках.

У MvvmCross найбільш важливою особливістю є можливість знаходження різних ViewModel в Portable Class Library (PCL), які можна додавати як посилання в будь-які інші проекти. Звичайно, це не єдиний цікавий аспект MvvmCross. Ця інфраструктура також базується на архітектурі на основі плагінів, підтримує вбудовування залежностей (dependency injection, DI) та інші функції.

Можна використовувати MvvmCross для розробки додатків на платформах Android та iOS.

MvvmCross можна легко використовувати, оскільки вона складається з кількох NuGet-пакетів, які додаються до проекту. Потім необхідно виконати кілька простих дій перед запуском програми. Хоча операції в iOS та Android відрізняються, але все ж вони досить схожі.

2.7 Зберігання даних

SQLite - це відкрите ядро баз даних без серверу, яке дозволяє створювати та операційно обробляти локальні бази даних. Вся інформація зберігається в таблицях, та операції над даними можна виконувати за допомогою коду на мові C# та LINQ-запитів.[10]

SQLite ідеально підходить для крос-платформної розробки, оскільки воно є портативним ядром баз даних. Ядро SQLite встановлюється заздалегідь як в iOS, так і в Android, і може легко розгортатися в Windows, що робить його чудовим компонентом для створення крос-платформових мобільних додатків, що працюють з локальними базами даних, зокрема з використанням Xamarin.Forms.

Для роботи з базою даних SQLite в проекті необхідно використовувати управління доступу до неї.

Використання SQLitePCL дозволяє створювати локальні бази даних в додатках для різних платформ, таких як Windows, Windows Store, Windows Phone, Android (Xamarin) та iOS (Xamarin). Ця бібліотека є безкоштовною та має відкритий код, доступний для всіх бажаючих.

Існують різні підходи та бібліотеки для роботи з SQLite, але найбільш рекомендованою є SQLite.NET. Ця бібліотека надає просте ORM-рішення (Object Relational Mapping) для розробки з використанням Xamarin.

SQLite.NET дозволяє використовувати базу даних як сховище об'єктів, що дозволяє маніпулювати даними у вигляді стандартних класів C# без необхідності використовувати SQL-вирази.

При роботі з SQLite в Xamarin існує одна проблема - повний шлях до бази даних буде відрізнятися на кожній окремій платформі. Щоб вирішити цю проблему, рекомендується використовувати механізм впровадження залежностей (dependency injection).

3 ПРОЕКТУВАННЯ ПРОЕКТУ

3.1 Розробка діаграм

Діаграма прецедентів (варіантів використання) відображає зв'язки між акторами та прецедентами системи, дозволяючи описати систему на концептуальному рівні. Прецедент - це можливість моделювання системи, яка представляє частину її функціональності та дозволяє актору отримати конкретний результат, який він потребує. Прецедент відповідає окремому сервісу системи та описує типовий спосіб взаємодії користувача з системою. Діаграма прецедентів зазвичай використовується для специфікації зовнішніх вимог до системи.

Під час проектування було визначено такий актор.

Користувач є особою, яка має можливість використовувати всі функції додатка.

Діаграма прецедентів була створена на основі функціональних вимог до мобільного додатку.

Рисунок 3.1 - Діаграма прецедентів

Функціональність "Додати новий дохід" полягає в внесенні нових доходів у світовій валюті.

При виборі категорії визначається, з якої категорії отримано дохід: зарплата, депозит, збереження тощо.

Функціональність "Додати нову витрату" полягає в внесенні нових витрат з таких категорій, як Їжа, Житло, Здоров'я, Одяг, Транспорт, Подарунки.

При функціоналі "Додати опис" можна додати будь-який опис витрати.

При функціоналі "Переглянути витрати за певний період" можна побачити статистику своїх витрат за обраний період часу: день, тиждень, місяць, рік, весь час.

Функціонал "Переглянути витрати за певною категорією" дозволяє переглядати статистику витрат за певними категоріями.

Діаграма послідовності (sequence diagram) є однією з UML-діаграм, яка відображає взаємодію між об'єктами та послідовність їх подій. На рисунку 10 наведено приклад діаграми послідовності, що описує процес додавання доходів та витрат у відповідності з вимогами до мобільного додатку.

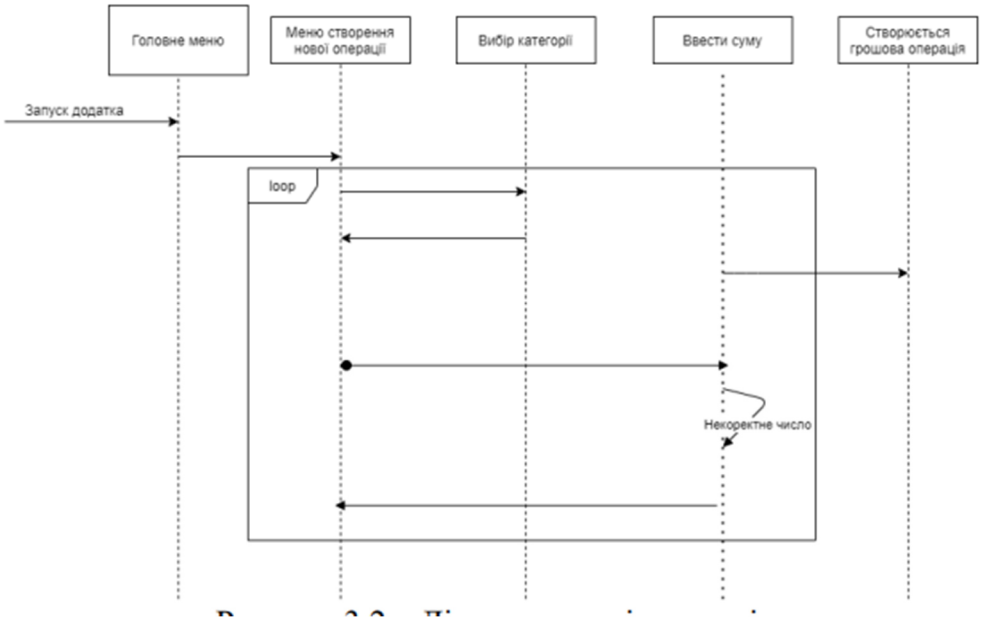


Рисунок 3.2 – Діаграма послідовності

Також була створена діаграма діяльності, яка дозволяє повніше розуміти процес користування додатком користувачем. У даному процесі користувач додає нові витрати або доходи у поле "Додати", обирає категорію, та після додавання всіх необхідних даних, натискає кнопку "Додати дохід/витрату".

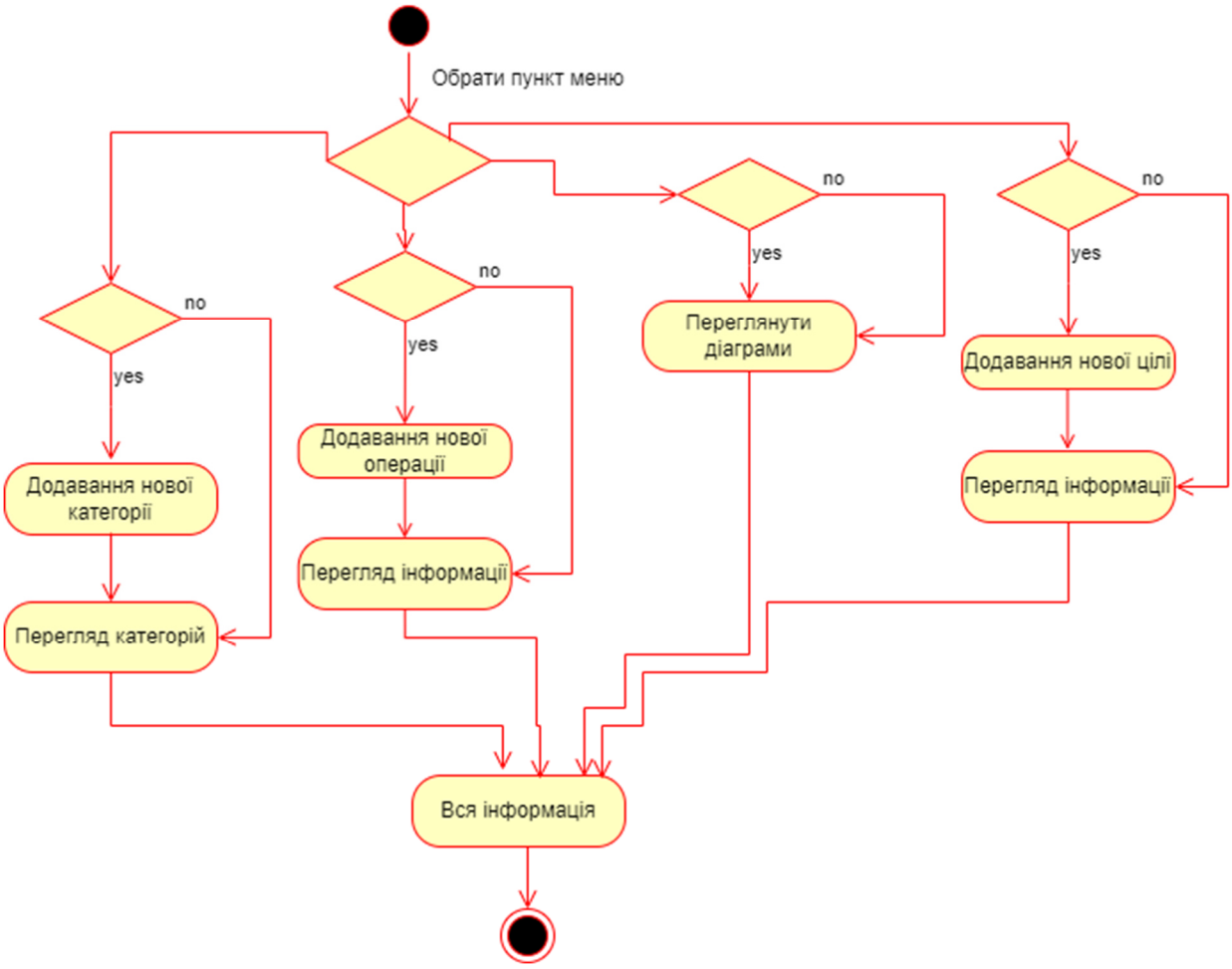


Рисунок 3.3 – Діаграма діяльності

Для створення високоякісного додатку необхідно розуміти взаємозв'язки між об'єктами. Для цього важливо знати, які об'єкти входять до предметної області проекту та які логічні зв'язки між ними існують. Для формування такого розуміння використовують діаграми предметної області.

Логічна модель має на меті зобразити логічну структуру предметної галузі у графічному вигляді.

Логічна модель предметної галузі відображає основні концепти та їх взаємозв'язки в предметній галузі.

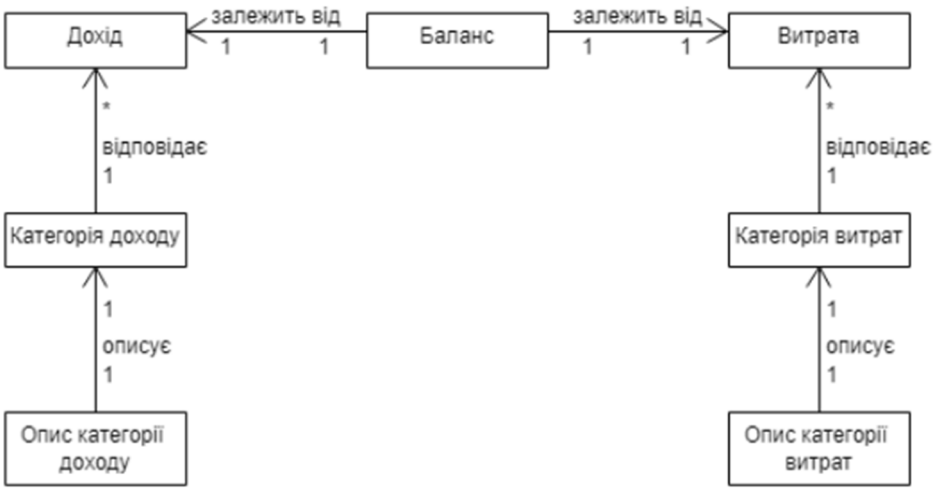


Рисунок 3.4 – Діаграма предметної галузі

Була створена діаграма артефактів, яка демонструє логіку інтерфейсу користувача.

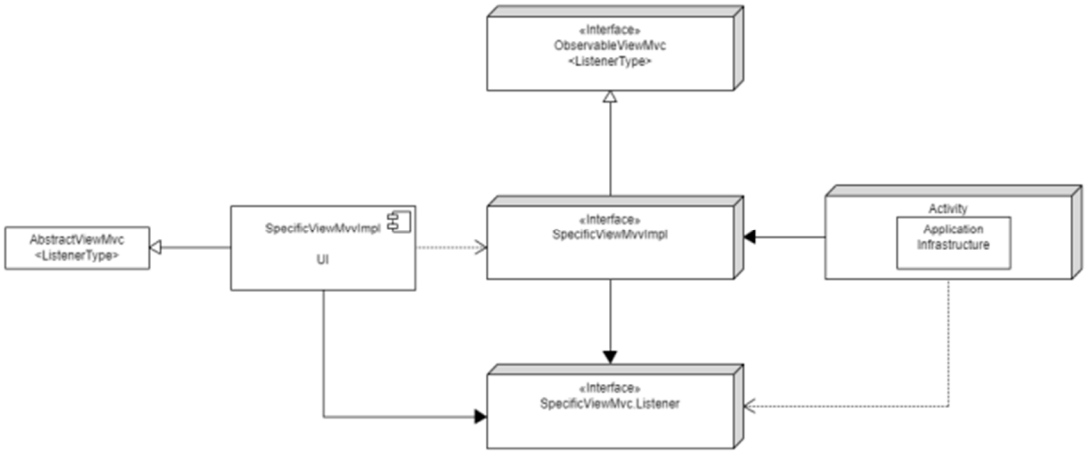


Рисунок 3.5 - Діаграма артефактів

Була розроблена діаграма пакетів, яка допомагає в проектуванні додатку. Ця діаграма дозволяє виділити залежності між компонентами системи, що допомагає знизити їх кількість і, відповідно, зв'язаність компонентів. Хоча пакети не

відповідають на питання про зменшення кількості залежностей в системі, вони забезпечують краще розуміння залежностей та взаємозв'язків між компонентами.

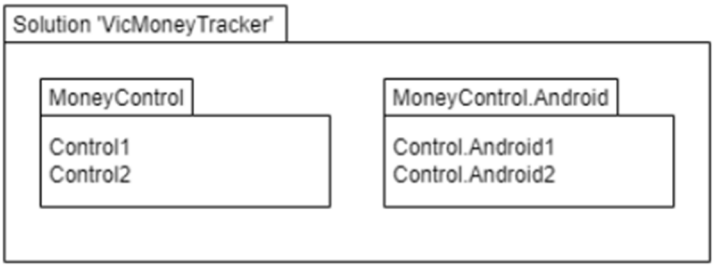


Рисунок 3.6 – Діаграма пакетів

3.2 Макет активностей додатку

Було створено макет проекту, використовуючи діаграми прецедентів та послідовності як основу та враховуючи вимоги до функціоналу додатку. Графічно показано, як буде виглядати додаток, включаючи його вікна та компоненти. На цій моделі було описано розташування елементів інтерфейсу програми, таких як кнопки, списки, поля введення, іконки тощо.

3.2.1 Головне вікно додатку. Розділ ‘‘Доходи’’. Категорії дозодів.

Головне вікно додатку показує поточну інформацію про доходи, витрати та загальний баланс користувача. Крім того, воно дозволяє здійснювати навігацію до основних функцій програми.

При вході в розділ "Доходи" відкривається вікно, де можна додавати інформацію про доходи.

При додаванні доходів можна обрати категорію, натиснувши на відповідний елемент інтерфейсу. Це відкриє вікно «Категорії доходів», де відображається список наявних категорій доходів. Тут можна також створити нову категорію або змінити наявну.

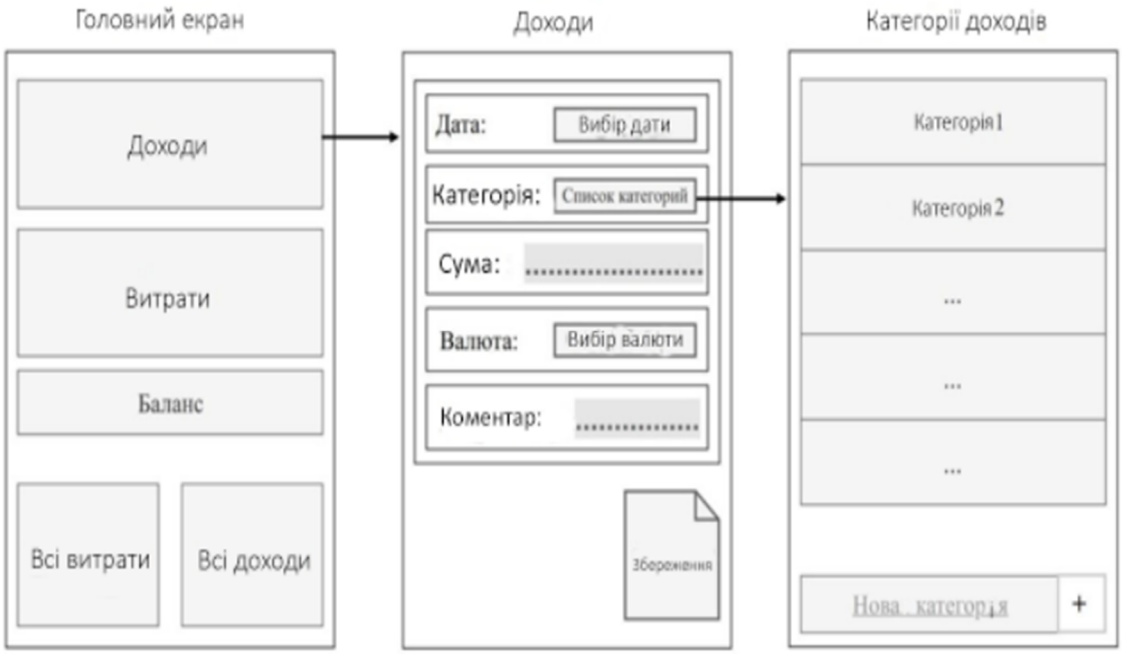


Рисунок 3.7 - Макети головного вікна, доходів і категорій доходів

3.2.2 Секція ‘‘Всі доходи’’. Діаграма доходів. Список доходів по категоріям

Після натискання кнопки "Всі доходи" на головному вікні, з'явиться окреме вікно під назвою "Всі доходи", яке міститиме інформацію про загальну суму доходів за обраний період часу. Крім того, натискання кнопки "Діаграма" у цьому вікні відкриє нове вікно з діаграмою, яка візуально представлятиме відсоткове співвідношення суми доходів за різними категоріями.

Якщо натиснути на певну категорію в розділі "Всі доходи", буде відображений список записів з цієї категорії. У вікні списку записів є можливість відредагувати або видалити записи за бажанням.

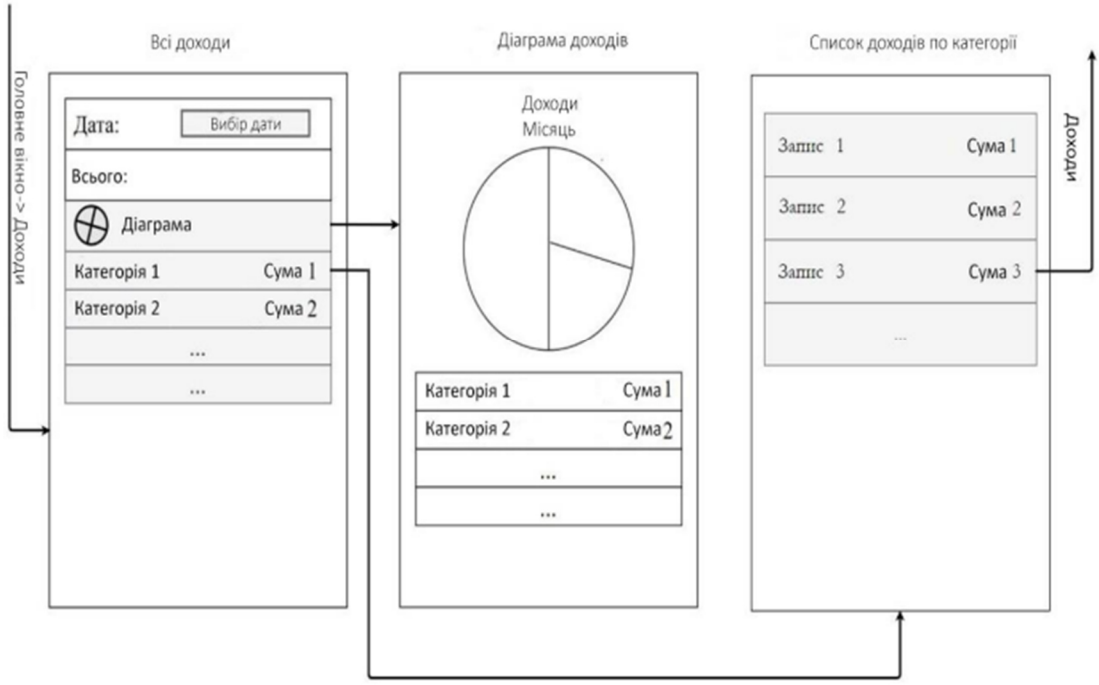


Рисунок 3.8 - Макет всіх доходів, діаграми доходів і списку доходів по категорі

3.2.3 Розділ ‘‘Витрати’’. Категорії витрат. Баланс

При переході до розділу "Витрати" відкривається активність, де можна додавати витрати. Натискання на "Категорію" відкриває вікно, де можна вибрати категорію витрат.

У розділі "Категорії витрат" відображається список доступних категорій, де можна вибрати певну категорію або створити нову. Також є можливість відредагувати або видалити наявні категорії.

При виборі розділу "Баланс" з головного вікна, відкривається сторінка зі статистикою по місяцях, що включає інформацію про витрати та доходи.

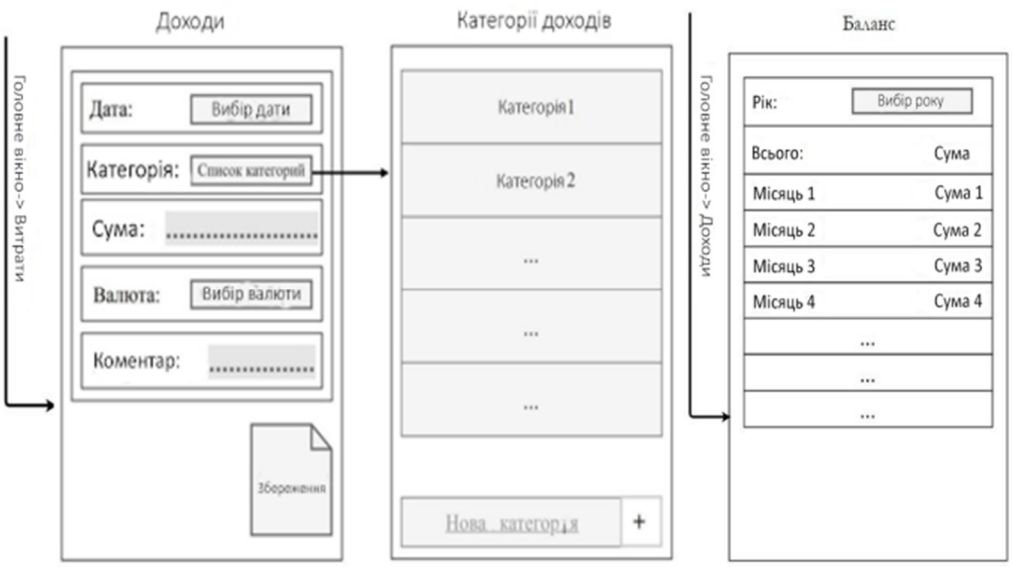


Рисунок 3.9 - Макет витрат, категорій витрат і балансу

3.2.4. Всі витрати. Діаграма витрат. Список витрат по категорій

При переході в розділ "Усі витрати" з головного вікна, користувач отримує зведену інформацію про витрати за категоріями протягом вказаного періоду. При натисканні кнопки "Діаграма" відбувається відкриття активності "Діаграма витрат", на якій представлена кругова діаграма, що відображає відсоткове співвідношення суми витрат за категоріями.

При виборі категорії в вікні "Усі витрати", у новому вікні будуть відображені витрати, пов'язані з цією категорією. Користувач має можливість внести зміни до записів, пов'язаних з цією категорією.

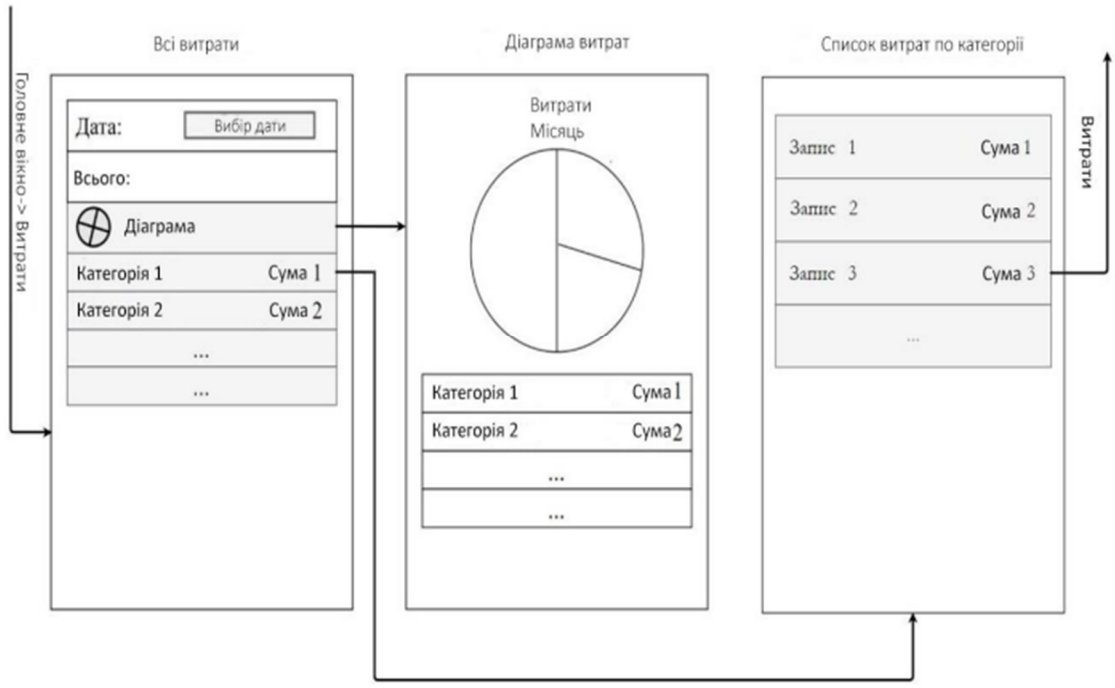


Рисунок 3.10 - Макет всіх витрат, діаграми витрат і списку витрат по категорії

4.ФАЗА ТЕСТУВАННЯ ТА РЕАЛІЗАЦІЇ ДОДАТКУ

4.1 Тестування додатку

Процес розробки програми включав етапи поетапного тестування. Для цього були створені емулятори смартфонів та планшетів з різними розмірами екрану та версіями Android. Програмний продукт був послідовно запусканий на цих емуляторах, його поведінка аналізувалася, а при потребі вносилися зміни в код на основі результатів аналізу.

Були виконані наступні тести:

1. Було проведено тестування поля вводу з неприпустимими даними для перевірки поведінки програми під час обробки цих невірних даних.
2. Додаток був протестований на пристроях, які працюють під різними версіями Android, з метою виявлення особливостей його роботи в різних операційних системах.
3. Після завершення розробки, програмний продукт пройшов тестування на реальних пристроях, зокрема Xiaomi Mi A1, Xiaomi Redmi 8 Pro.

4.2 Структура додатку

За допомогою архітектурного шаблону MVVM (Model-View-ViewModel), мій додаток має наступну структуру.

Таблица 1 - Структура додатку

<table>
<tr><td colspan="3">Application</td></tr>
<tr><td colspan="2">Core</td><td>User Interface</td></tr>
<tr><td>Models</td><td>View Models</td><td>View</td></tr>
</table>

Ядро (або Core) - це компонент програми, який містить в собі всю логіку додатку, тоді як інтерфейс користувача (User Interface) - це складова, що описує зовнішній вигляд інтерфейсу програми.

У розділі Ядра визначаються класи для моделей та ViewModel, тоді як у розділі UI описуються класи, що відповідають за візуальну частину програми.

У моєму проекті структура MVVM виглядає наступним чином:

Нижче наведено короткий опис послідовності дій, що виконуються для реалізації цього принципу у моєму проекті:

Ядро(Core)

1. Створення PCL-проекту (Portable Class Library) здійснюється з метою забезпечення переносимості коду та можливості його подальшої перебудови під різні операційні системи.
2. Встановлення пакета MvvmCross здійснюється шляхом додавання його до редактора пакетів NuGet.
3. Створення папки "Model", де будуть розміщені дві моделі: "Transaction.cs" і "Wallet.cs".
4. Створення класу "WalletViewModel.cs", в якому буде реалізована основна логіка програми. Цей клас буде успадковуватися від "MvxViewModel".

У цьому класі будуть визначатися властивості для доходів, витрат та поточного балансу. Також будуть описані команди, що дозволять зв'язати елементи управління з логікою програми.

5. Включення класу App.cs, який успадкований від MvxApplication, дозволить зареєструвати стартову точку і забезпечить необхідні налаштування для запуску програми.

UI

1. Установка пакета MvvmCross.
2. Уявлення (візуальна частина) програми будуть зберігатися у папці Resources/layout. У цьому проекті інтерфейс реалізовується за допомогою

XML-розмітки, оскільки використовується технологія, відмінна від Xamarin.Forms.

Файли XML-розмітки визначатимуть структуру сторінки шляхом опису всіх кнопок, елементів TextBox та інших елементів. Також у цих файлах відбудеться зв'язування між елементами керування та їх логікою. Наприклад, опис кнопки буде мати такий вигляд:

<Button android: text = "Додати" android: layout_width = "match_parent" android: layout_height = "wrap_content" local: MvxBind = "Click AddConsumptionCommand" />

1. Створення папки "Views", де розташовуватиметься Activity. Головна Activity має включати метод "OnCreate()", в якому буде визначатися, яке вікно відкривається при запуску програми.
2. Створення папки "Views", де розташовуватиметься Activity. Головна Activity має включати метод "OnCreate()", в якому буде визначатися, яке вікно відкривається при запуску програми.

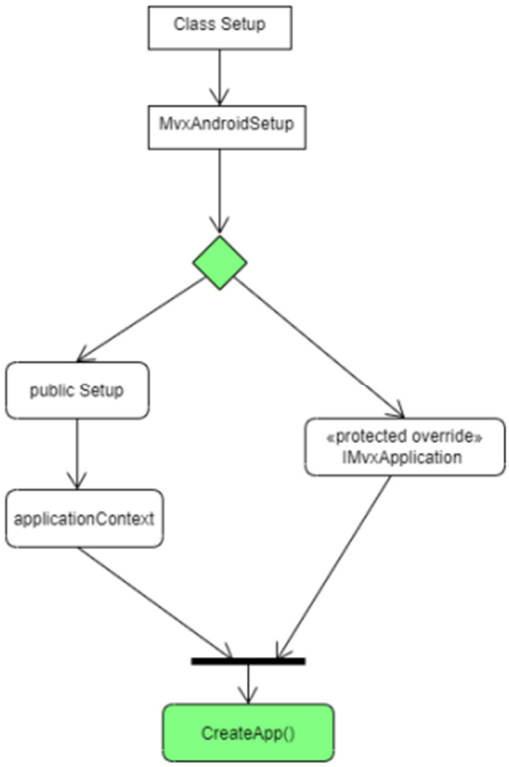


Рисунок 4.1 - Створення класу Setup

4.3 SQLite

SQLite є компактною вбудованою реляційною базою даних. Вираз "вбудована" означає, що SQLite не використовує клієнт-серверну архітектуру, де движок SQLite працює як окремий процес, з яким програма взаємодіє. Замість цього, SQLite надає бібліотеку, яка вбудовується в програму і стає її складовою частиною. SQLite має кілька переваг, серед яких:

- SQLite відрізняється високою надійністю. Під час випуску кожної версії вона пройшла низку ретельних автоматичних тестів, включаючи близько 2 мільйонів тестів. Крім того, код SQLite має 100% покриття тестами, що забезпечує високу якість і надійність бази даних.
- SQLite відзначається простотою та зручністю вбудовування. Він доступний на будь-якому Android-пристрої і не потребує окремої установки. Це означає, що ви можете легко використовувати SQLite в своєму додатку без необхідності встановлення додаткового програмного забезпечення.
- SQLite відзначається високою продуктивністю. Додатки, що використовують SQLite, зазвичай працюють швидше, ніж ті, що використовують MySQL (у 2-3 рази швидше) та PostgreSQL (у 10-15 разів швидше) для більшості типових завдань.

SQLite ідеально підходить для Android-додатків, особливо якщо більшість запитів становлять запити на читання. Він також чудово підходить для невеликих проектів. Оскільки SQLite використовує стандартні SQL запити, що використовуються в серверних базах даних, це дозволяє легко перейти на іншу базу даних, не змінюючи суттєво логіку програми.

Використання структури таблиць:

1. Існує багато бібліотек для роботи з базами даних SQLite у Microsoft .NET Framework, але для наших потреб нам потрібна спеціальна портована бібліотека, яка підтримує також Xamarin-додатки. Ця бібліотека називається SQLite-net і є легковаго виконанням з відкритим вихідним кодом для .NET,

Mono і Xamarin-додатків. Вона доступна як пакет NuGet з назвою "sqlite-net-pcl". Ми додамо цей пакет до нашого проекту повністю - як до ядра, так і до інтерфейсу користувача.

2. Для доступу до бази даних SQLite код використовує рядок підключення. Оскільки SQLite база даних є файлом, розташованим в локальній папці, для побудови рядка підключення потрібно вказати шлях до цієї бази даних.

Створення рядка підключення вимагає платформоспецифічного коду. Після цього ви можете використовувати вбудовування залежностей (dependency injection), щоб отримати доступ до рядка підключення.

```
public interface IDatabaseConnection
{
SQLite.SQLiteConnection DbConnection();
}
```

Цей інтерфейс містить метод DbConnection, який буде реалізований в кожному проекті, специфічному для платформи, і повертатиме відповідний рядок підключення.

3. Ми створюємо окремі класи для кожної таблиці. Кожен клас повинен реалізувати інтерфейс INotifyPropertyChanged, який дозволяє сповіщати про зміни в даному класі, які зберігаються.
4. Ми здійснюємо виконання операцій CRUD (створення, читання, оновлення, видалення) з даними.

Операції створення, читання, оновлення та видалення (CRUD) є надзвичайно важливими. Об'єкт SQLiteConnection надає методи Insert, InsertAll, Update і UpdateAll, які дозволяють вставляти або оновлювати об'єкти в базі даних.

Методи InsertAll і UpdateAll виконують операцію вставки або оновлення набору об'єктів, який реалізує інтерфейс IEnumerable<T>, переданий як аргумент.

Операція вставки або оновлення виконується пакетним способом, і обидва методи також підтримують виконання операцій в межах транзакції.

4.4.1 Реалізація інтерфейсу

Activity і View є двома ключовими поняттями в інтерфейсі Android, важливими для розробки додатків.

Activity - це компонент програми, який взаємодіє з користувачем. Його можна порівняти з "вікном" у традиційних операційних системах. Усередині Activity розміщуються різноманітні елементи інтерфейсу, що взаємодіють з користувачем.

View - це елемент інтерфейсу, який включає в себе різноманітні компоненти, такі як кнопка, поле для введення тексту, контейнер для зображень і т.д. Вони відображаються на екрані пристрою і дозволяють користувачеві взаємодіяти з програмою.

Так само значущим елементом є ViewGroup. Фактично, це модифікований варіант View, призначений для використання як контейнер для інших елементів View. Він дозволяє групувати і організовувати інші елементи View в структуровані групи, що спрощує організацію і відображення компонентів інтерфейсу.

Layout - це загальна назва для декількох класів, які успадковуються від ViewGroup. Вони використовуються як контейнери для елементів View і створені з метою зручного розташування кнопок, полів для введення тексту та інших елементів інтерфейсу. Layout дозволяють організовувати і керувати розміщенням та розмірами елементів інтерфейсу на екрані, забезпечуючи належне відображення додатку.

Основні layouts:

- LinearLayout
- FrameLayout
- RelativeLayout

4.4.2 Головне меню

Головний екран включає меню, що містить інформацію про доходи та витрати, які відображаються в стовпчиковому вигляді, відсортованому за датою додавання.

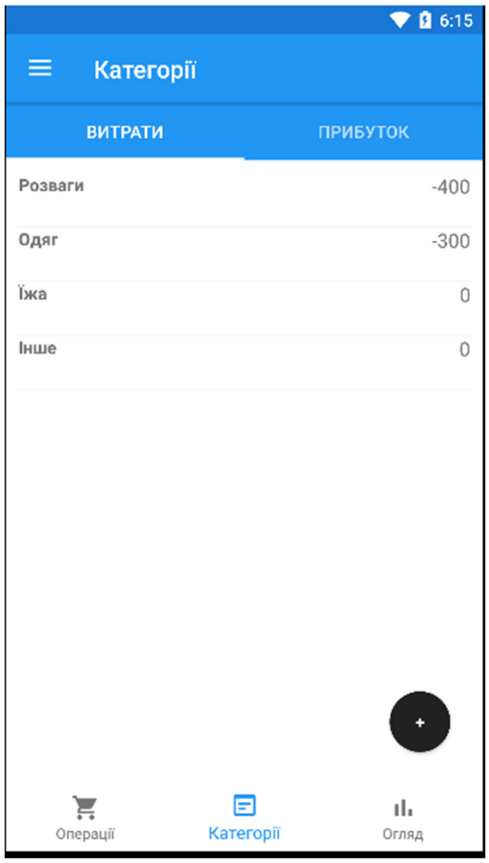


Рисунок 4.2 - Головне меню

4.4.3 Нова операція

Щоб перейти на вкладку "Нова операція", потрібно натиснути кнопку з символом "+".

Вкладка "Нова операція" містить поля для заповнення такою інформацією: опис, категорія, дата, час, рахунок.

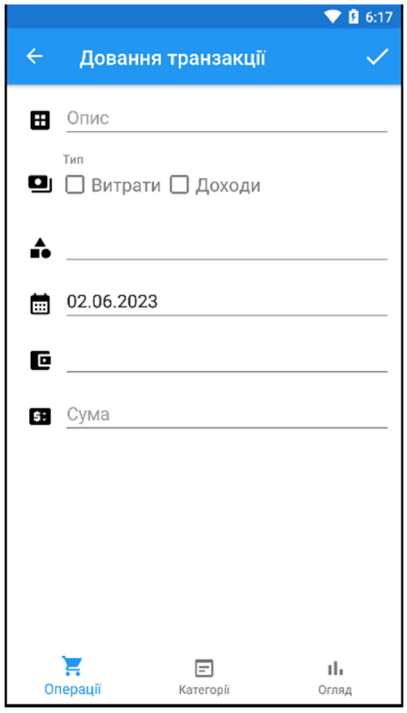


Рисунок 4.3 - Вкладка "Нова операція"

За замовчуванням, дата та час установлюються на момент переходу до цієї активності. Зміна дати реалізована за допомогою компонента CalendarView, який відображає календар із розділу Widgets. Для зміни дати достатньо натиснути кнопку з зображенням календаря.

Для правильного введення суми використовується атрибут android:inputType="NumberDecimal", що обмежує введення літерних значень та дозволяє вводити лише десяткові числа.

Опис є необов'язковим полем при додаванні, за замовчуванням воно залишається порожнім. При введенні опису кількість символів не обмежена.

Категорія є обов'язковим атрибутом для додавання доходу або витрати. Натискаючи на іконку, відбувається перехід в активність Категорії де потрібно вибрати, що хочете ви додати з вкладок дохід чи витрату.

У верхній частині вікна категорій є можливість вибрати додавання доходу або витрати. У вкладці "Дохід" представлені наступні категорії: зарплата, стипендія.

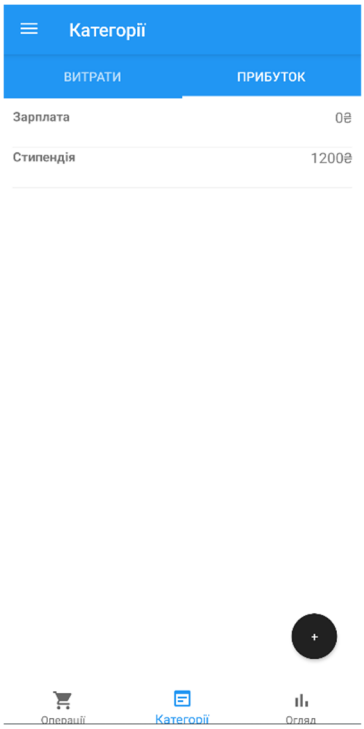


Рисунок 4.4 - Вибір категорії доходу

Вкладка "Витрати" містить наступні категорії: розваги, одяг, їжа, інше. Для вибору категорії необхідно натиснути на потрібну категорію.

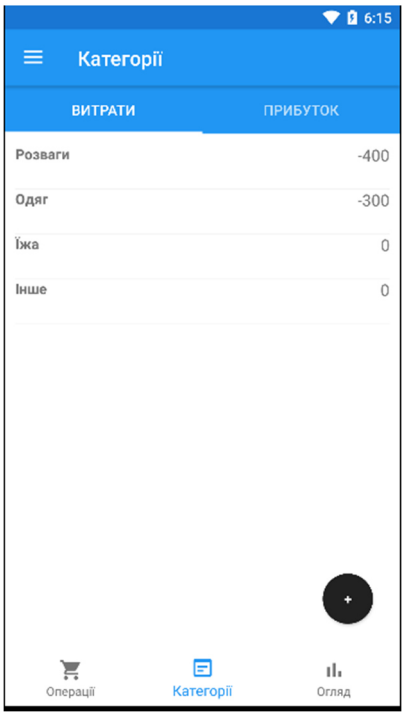


Рисунок 4.5 - Категорія "Витрати"

Для зручності користувачів надається можливість додавати власні категорії. Для цього необхідно натиснути кнопку "+".

У цій активності здійснюється додавання в базу даних назв категорій, які користувач вводить у поле введення.

Після збереження, введене значення відображається на екрані за допомогою компонента RecyclerView, що спрощує роботу зі списком.

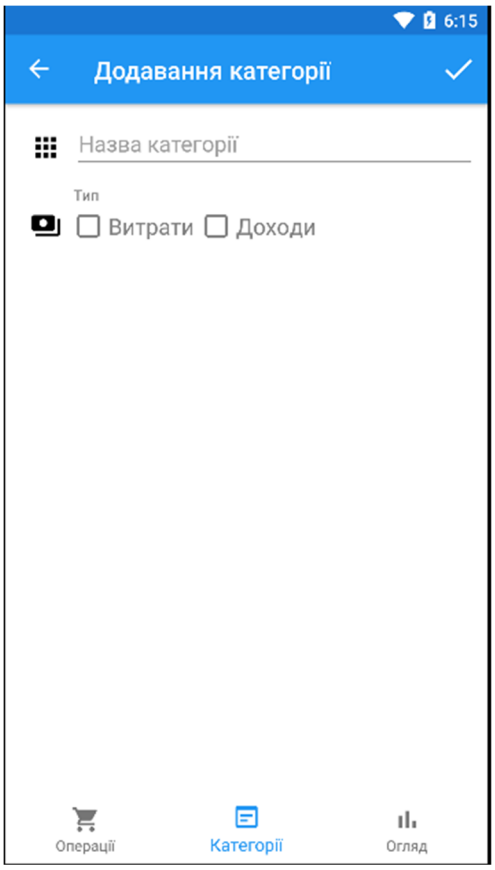


Рисунок 4.6 - Вікно "Нова категорія"

4.4.4 Випадаюче меню

Для створення меню використано Android Design Support Library. Випадаюче меню містить наступні пункти: операції, категорії, огляд і накопичення.

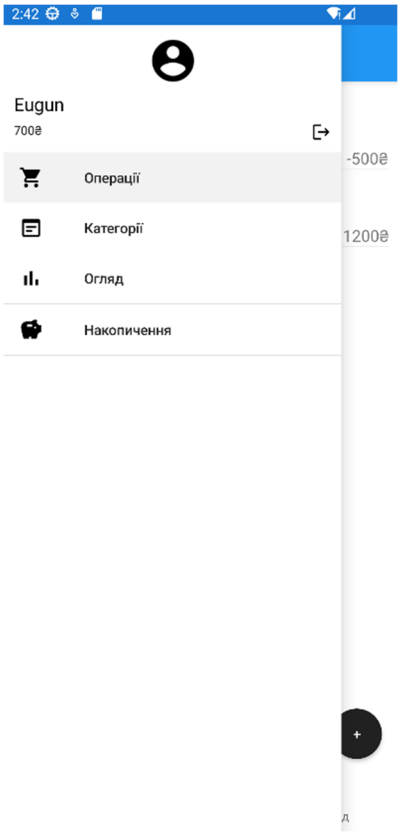


Рисунок 4.7 - Випадаюче меню

4.4.5 Реалізація діаграм

Подивитися статистику за категоріями можна, натиснувши на певний проміжок часу, за який ви бажаєте переглянути статистику. Це відкриє нову активність під назвою "Діаграма". Для створення кругової діаграми використовується бібліотека OxyPlot, яка надає можливість малювати графіки різної складності. Діаграма відображає статистику, що формується на основі інформації з бази даних за обраний часовий період. У цьому меню можна окремо переглянути діаграму доходів, діаграму витрат або спільну діаграму, яка об'єднує обидва види.

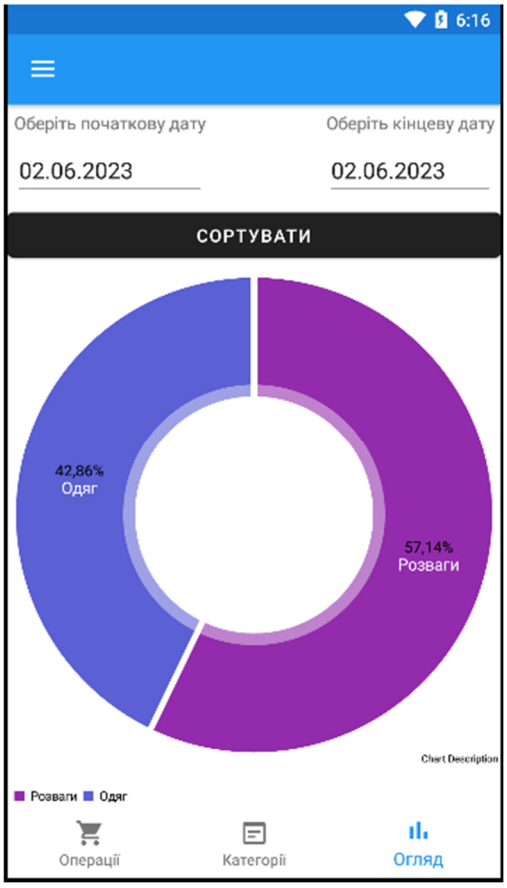


Рисунок 4.8 - Кругова діаграма

ВИСНОВКИ

1. У результаті цієї роботи було створено мобільний додаток для платформи Android, який дозволяє управляти особистими доходами і витратами. Розроблений додаток відповідає поставленим цілям і вимогам проекту.

   Проведено дослідження історичних даних та хронології розвитку мобільних додатків, а також порівняння з конкурентами. Також були зібрані статистичні дані щодо популярності розробки додатків для мобільних телефонів, що підтверджують актуальність даної теми.

   Була надана інформація про використовувані системи в процесі роботи. Проведено дослідження основних складових мобільного додатка. Вибраний та застосований необхідний інструментарій для розробки додатку, а також перераховані ключові бібліотеки та технології, що використовувалися.
2. Було проведено проектування проекту, в ході якого були змоделювані різні компоненти, зокрема прецеденти, послідовності, діяльності, предметна галузь, артефакти та пакети. Крім того, були розроблені макети меню для додатка.
3. У заключному етапі розробки було проведене тестування системи в цілому, яке включало в себе перевірку всіх компонентів. Цей процес пройшов без особливих проблем, оскільки проміжні тестування, здійснені протягом розробки, допомогли уникнути можливих порушень.
4. Додаток працює надійно і виконує свою основну функцію - дозволяє користувачам вести і керувати своїм бюджетом. За допомогою додатка можна додавати, редагувати і видаляти фінансові операції. Також доступна статистика за категоріями доходів і витрат, яка відображає інформацію про суму витрат у кожній категорії. Використання додатку не обмежено віковими рамками.
5. Під час виконання дипломного проекту були вивчені наступні аспекти:

- ОС Android;
- Платформа для кроссплатформенної розробки Xamarin;
- Реляційна база SQLite;
- Принципи проектування додатків;

6. У майбутньому планується продовжувати розвиток та модернізацію цього додатку шляхом впровадження нового функціоналу:

- поліпшення інтерфейсу;
- створення особистого кабінету;
- курс валют онлайн;
- віджети;
- синхронізація на декількох пристроях;
- реалізувати функціонал зчитування SMS-повідомлень від банків - це дозволить автоматично заповнювати безготівкові витрати та доходи у додатку. Однак, поточна версія розробленого додатку вже готова для публікації на Google Play і доступна для безкоштовного завантаження та використання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. The History of Mobile Apps [Електронний ресурс]. – Режим доступу: https://inventionland.com/inventing/the-history-of-mobile-apps/#:~:text=In%201997%2C%20the%20Nokia%206110,very%20basic%20version%

2. Європейці скористаються всіма перевагами єдиного цифрового ринку [Електронний ресурс] – Режим доступу: https://psm7.com/evropejcy-vospolzuyutsya-vsemipreimushhestvami-edinogo-cifrovogo-rynka.html

3. Nearly 25% of Americans have no emergency savings [Електронний ресурс] – Режим доступу: https://www.marketwatch.com/story/nearly-25-of-americans-have-no-emergency-savings-and-lost-income-due-to-coronavirus-is-piling-on-even-more-debt-2020-06-03

4. Що таке Android [Електронний ресурс] – Режим доступу: http://ipkey.com.ua/uk/faq/912-android.html

5. Розробка мобільних додатків на Xamarin: гармонія усіх платформ [Електронний ресурс] – Режим доступу: https://internetdevels.ua/blog/xamarin-mobile-app-development

6. Reynolds M. Xamarin Essentials – «Packt Publishing», 2014 – 30-31 ст.

7. Xamarin [Електронний ресурс] – Режим доступу до ресурсу: https://visualstudio.microsoft.com/ru/xamarin/

8. Fragments [Електронний ресурс] - https://developer.android.com/guide/fragments

9. Weil A. Learn WPF MVVM - XAML, C# and the MVVM pattern: Be ready for coding away next week using WPF and MVVM / Arnaud Weil., 2016. – 176 с

10. Дизайн додатків ОС Android Material Design [Електронний ресурс]. - 2018. – Режим доступу: https://developer.android.com/develop/ui/views/theming/look-and-feel

ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО- НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка програмного забезпечення з моніторингу повсякденних фінансових операцій мовою C#

Виконав студент 4 курсу
групи ПД-44
ГрибінчакЄвгеній Валерійович

Керівникроботи
Доктор філософії, доцент кафедри ІПЗ Гребенюк Віктор Вікторович

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** підтримка процесів моніторингу повсякденних фінансових витрат за рахунок використання програмного забезпечення мовою C#.
- **Об’єкт дослідження** моніторинг повсякденних фінансових операцій
- **Предмет дослідження** програмне забезпечення з моніторингу повсякденних фінансових операції.

ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

1. Визначення функціональних і нефункціональних вимог до програмного забезпечення з моніторингу фінансових операцій..
2. Встановлення основних функцій, які має виконувати програма, включаючи додавання, редагування та видалення операцій, генерацію звітів та статистики, категоризацію операцій
3. Розробка програмного коду на мові C# для реалізації визначених функцій та архітектури.
4. Розробка інтерфейсу користувача, що відповідає вимогам та забезпечує зручну роботу з програмою.

АНАЛІЗ АНАЛОГІВ

	CoinKeeper	Spendee	MoneyWiz	Financier
Зрозумілий та простий інтерфейс	+	+	-	+
Створеннязаписів про витратита доходи з використанням різнихатрибутів	+	+	+	+
Мінімальна кількість кроків для додавання операції	+	+	+	+
Прості формати вводу	+	-	+	+
Візуалізація даних	+	+	+	+
Мінімалістичний дизайн	-	+	-	+

ВИМОГИ ДО ДОДАТКУ

Функціональні вимоги

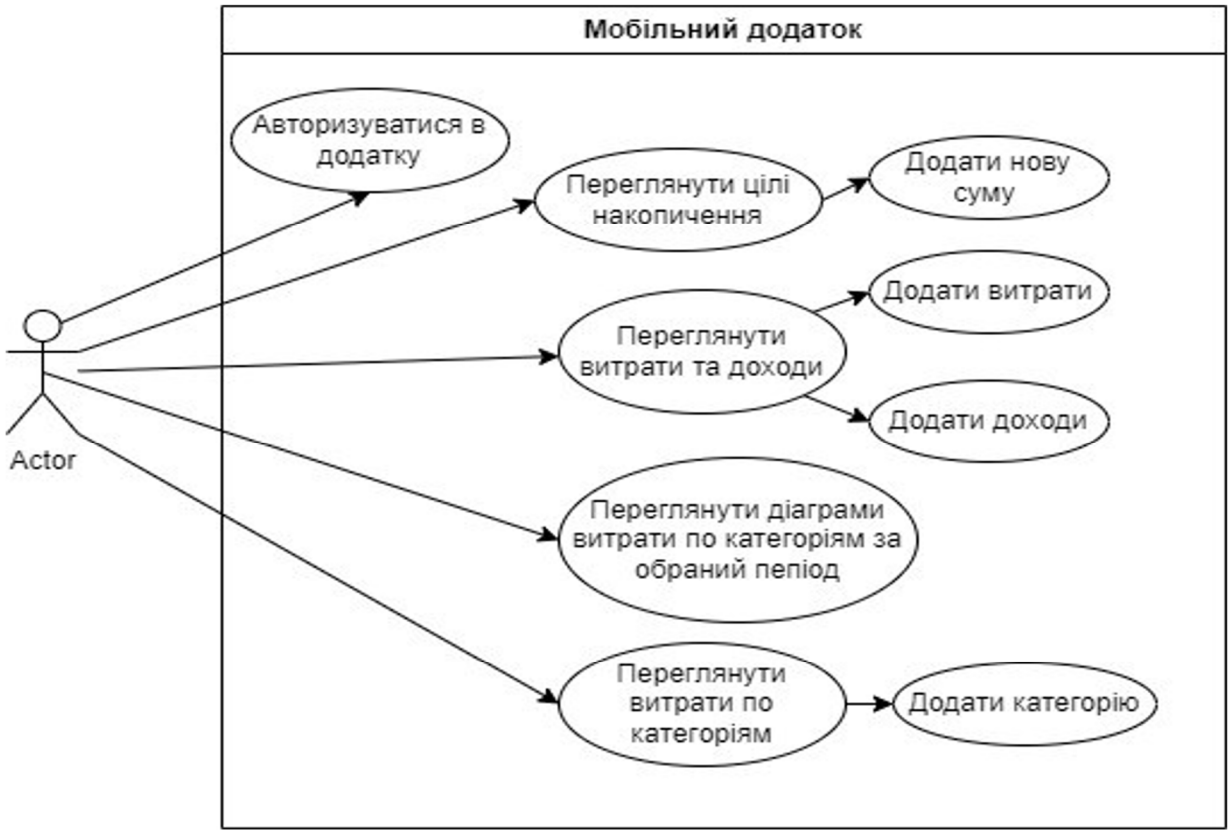
1. Створення записів про витрати та доходи з використанням різних атрибутів, таких як дата, категорія, сума, валюта, коментар.
2. Змінення або вилучення фінансових операцій, які були додані раніше.
3. Можливість створення, видалення та редагування категорій витрат і доходів.
4. Перегляд доходів та витрат з вибором діапазону часу.
5. Вибір валюти для фінансових операцій.
6. Перегляд статистики витрат і доходів у вигляді кругових діаграм.

Нефункціональні вимоги

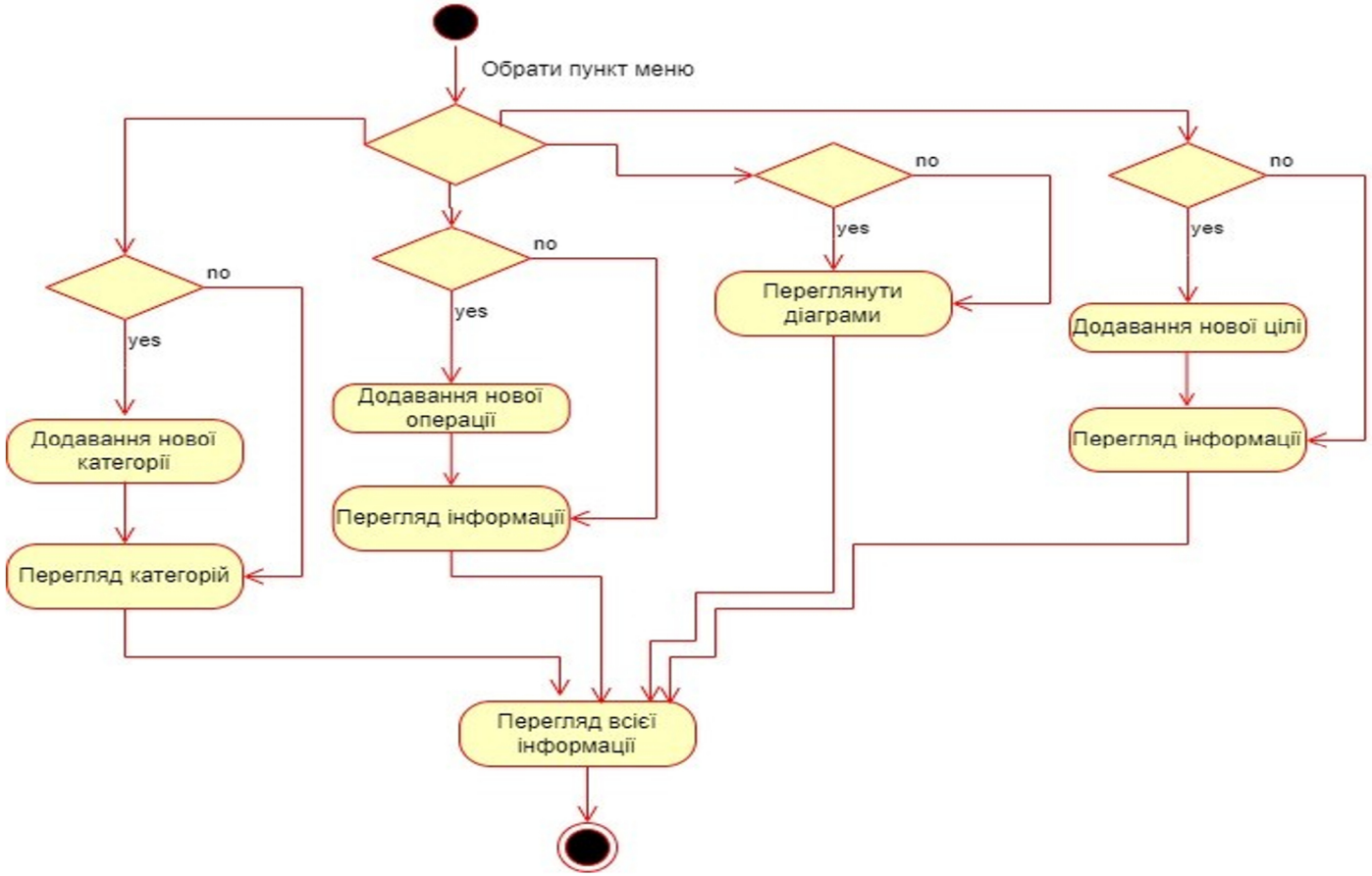
1. Мобільний додаток під систему Android .
2. Інтерфейс користувача для взаємодії з програмним забезпеченням.
3. Список валют з якими працює додаток: гривня.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

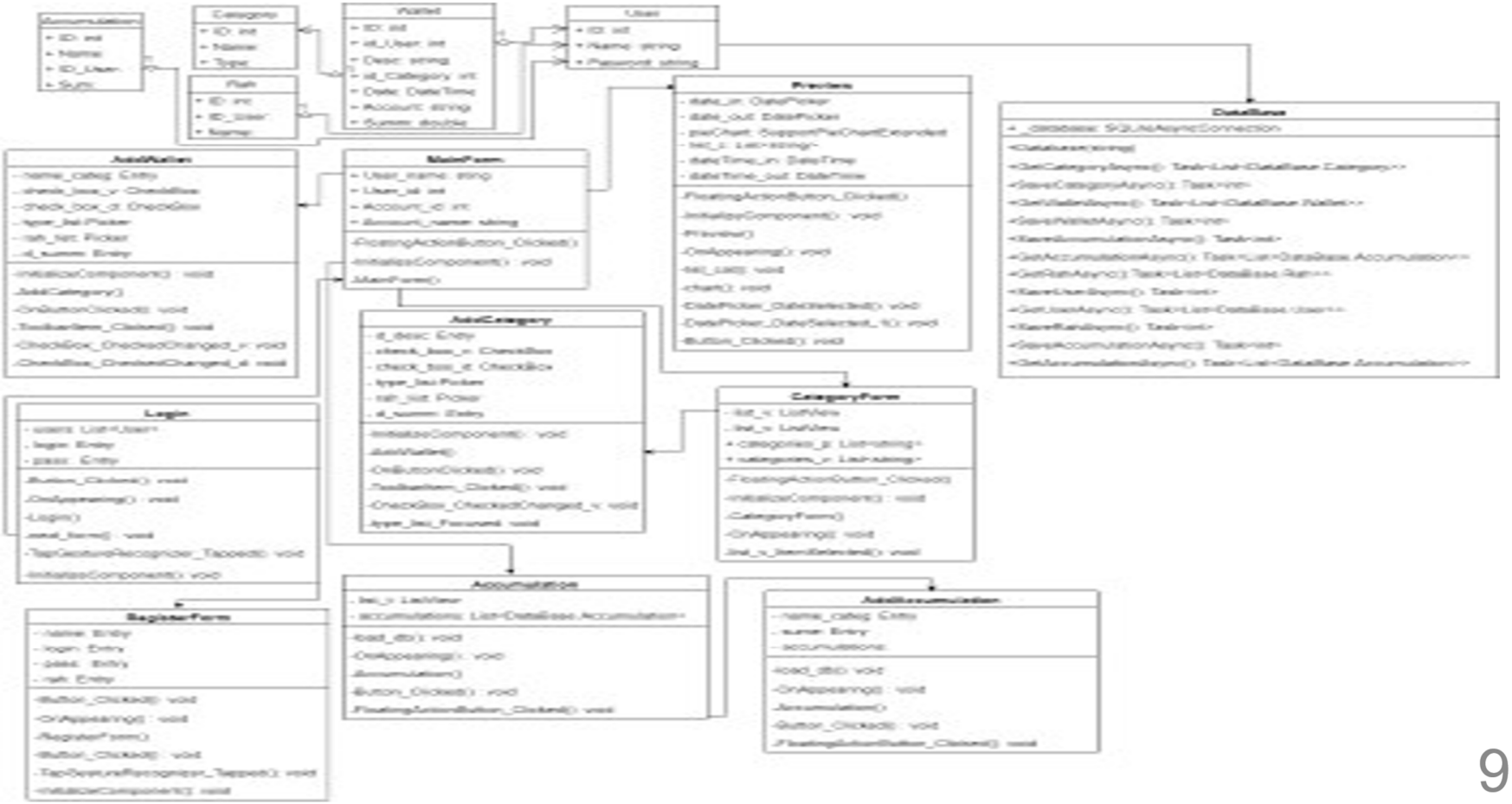
ДІАГРАМА ВАРІАНТІВ ВИКОРИСТАННЯ



ДІАГАРАМА ДІЯЛЬНОСТІ



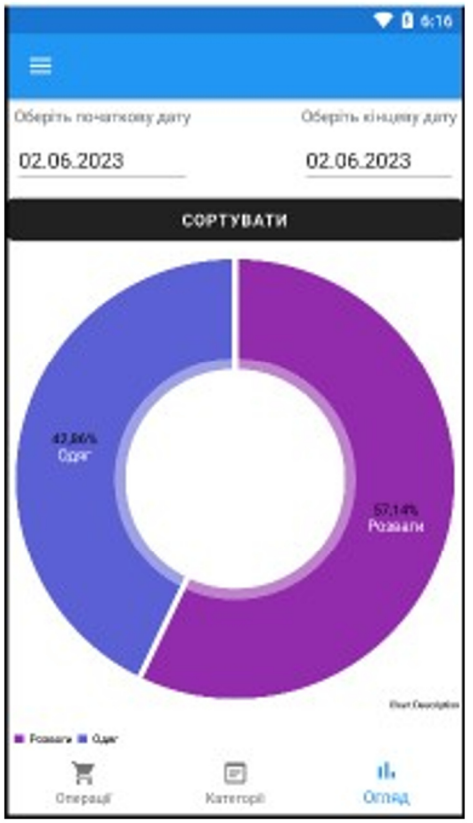
ДІАГРАМА КЛАСІВ



ЕКРАННІ ФОРМИ

Вікно ‘‘Категорій’’

Вікно входу



Вікно ‘‘Огляд’’

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Грибінчак Є.В Моніторинг повсчкдених фінансових операцій / Жебка В.В, Грибінчак Є.В // Сучасні аспекти діджиталізації та інформатизації в програмній та комп'ютерній інженерії, 01-03 червня 2023р., ДУТ, м. Київ – К: ДУТ, 2023. Подано до друку.

ВИСНОВКИ

1. Розглянуто основні технології та інструменти розробки програмного забезпечення на мові C#, що дозволило ефективно реалізувати поставлені завдання.
2. Розроблено програмне забезпечення з моніторингу повсякденних фінансових операцій, що дозволяє користувачам ефективно контролювати свої фінанси та управляти своїми витратами.
3. Розроблена програма забезпечує основні функції, включаючи додавання, редагування та видалення фінансових операцій та статистики.