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 МОБІЛЬНОГО ДОДАТКУ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ДОХОДАМИ ТА ВИТРАТАМИ МОВОЮ C#**»

Виконав: студент 4 курсу, групи ПД-42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)
Кривошапка Д.О.
(прізвище та ініціали)
Керівник Гаманюк І.М.
(прізвище та ініціали)
Рецензент _____
(прізвище та ініціали)

Київ –2021

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.
“____” _____2021 року

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

КРИВОШАПКА ДМИТРО ОЛЕКСАНДРОВИЧ
(прізвище, ім’я, по батькові)

1. Тема роботи: «Розробка мобільного додатку для управління особистими доходами та витратами мовою C#»

Керівник роботи: Гаманюк І.М., к.т.н., доцент кафедри ІПЗ
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «12» березня 2021 року №65
2. Строк подання студентом роботи 1.06.2021
3. Вихідні дані до роботи:
   3.1 Огляд ринку мобільних додатків;
   3.2 Системи та інструменти для створення додатку;
   3.3 Розробка структури проекту;
   3.4 Набір тест-кейсів та план проекту;
   3.5 Науково-технічна література.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   4.1 Ринок мобільних додатків;
   4.2 Огляд використовуваних систем та інструментів;
   4.3 Проектування проекту;
   4.4 Тестування та реалізація готового додатку;
   4.5 Висновки

5. Перелік графічного матеріалу.
   5.1 Основні характеристики роботи;
   5.2 Актуальність задачі;

5.3 Для чого потрібні додатки для контролю власних кошт;
5.4 Недоліки існуючих додатків;
5.5 Операційна система Android;
5.6 Xamarin;
5.7 Переносимість коду;
5.8 Метод Activity;
5.9 Патерн MVVM;
5.10 Model;
5.11 View;
5.12 ViewModel;
5.13 Зберігання даних;
5.14 Material Design;
5.15 Пакети Nuget;
5.16 Висновки;

6 Дата видачі завдання 19.04.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної літератури	19.04.21	Виконано
2	Створення вимог до додатку	20.04.21	Виконано
3	Вивчення фреймворку Xamarin	21.04.21	Виконано
4	Створення макету додатку	22.04.21	Виконано
5	Створення візуальної частини додатку	23.04.21	Виконано
6	Створення логіки додатка	28.04.21	Виконано
7	Вступ,висновки,реферат	30.04.21	Виконано
8	Розробка обов’язкових демонстраційних матеріалів	02.04.21	Виконано
9	Попередній захист роботи		
10	Здача роботи		

Студент _____ Кривошапка Д.О.

Керівник роботи Гаманюк І.М.

ЗМІСТ

РЕФЕРАТ

Текстова частина дипломної роботи 52 стр., 32 рис., 1 табл., 11 джерел

РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ДОХОДАМИ ТА ВИТРАТАМИ МОВОЮ C#

Об'єктом дослідження - є розробка мобільного додатку під систему Android.

Предметом дослідження – є розробка мовою C# мобільного додатку для контролю власного бюджету.

Мета роботи – отримати працюючий мобільний додаток для платформи Android, який дозволяє швидко, в режимі багатозадачності, вносити дані та зміни про свої фінансові операції, оперативно орієнтуватися в них за категоріями, датами, спостерігати їх статистику і тим самим контролювати одну з важливих сфер свого життя - свій бюджет.

Методи дослідження. У ході роботи було досліджено: мову програмування C# та фреймворк Xamarin, операційну систему Android, реляційну базу SQLite, принципи проектування додатків, дослідження в області дизайну мобільних додатків.

Додаток пристосований для людей які хочуть повністю контролювати свої кошти.

ВСТУП

Актуальність теми - У наш час майже неможливо уявити життя сучасної людини без мобільного телефону, у зв'язку з якою мобільна розробка стрімко розвивається, як у плані функціональності, так і в плані дизайну. Зараз смартфони для власного функціоналу практично підключені до комп'ютерів: на них виконують службові завдання, займаються пошуком в інтернеті, а також зловмисною інформацією, не кажучи вже про прості мультимедійні функції - огляд фільмів, фотографій, прослуховування музики і так далі.

У світі смартфонів популярні мобільні телефони на базі операційної системи Android і iOS, а також Windows Phone / Windows 10 Mobile. Існує певна статистика, яка впливає на те, що частина мобільних додатків створюється як для Android, так і для iOS. У разі, якщо розробляти програми окремо на кожній платформі, можна відкрити їх з низькими труднощами, наприклад:

- необхідність підлаштовувати додаток під інтерфейс конкретних платформ
- різні способи реалізації певних функцій
- різне середовище розробки.

Сьогодні смартфони та планшети активно використовуються для управління особливостями, а також робочими даними. Фінанси - невід'ємна частина нашого життя, а фінансова грамотність полегшує життя людей. Тенденція автоматизації всього, що можна автоматизувати, торкнулася і цієї області. Тепер людям не потрібно носити з собою блокнот, який легко пом'яти, намочити, втратити і який легко можуть просто вкрасти. Мати такий додаток у своєму смартфоні з додатковими можливостями планування, повідомлень та огляду статистики - непогана альтернатива папірцям. До того ж, здійснюючи дрібні витрати, наприклад, оплату можна просто забути і не занести в блокнот, в той час як телефон не потребує окремих зусиль у пошуку ручок та опори для блокнота .

Зміни в індустрії мобільних телефонів повинні робити наше життя легше, смартфони тим і відрізняються від комп'ютерів, тим що є завжди в нас під рукою.

Об'єктом дослідження - є розробка мобільного додатку під систему Android.

Предметом дослідження – є розробка мовою C# мобільного додатку для контролю власного бюджету.

Мета дипломної роботи - розробка додатку для мобільних пристроїв під управлінням операційної системи Android, який дозволятиме вести персональний фінансовий облік доходів та витрат.

Завдання дослідження:

- вивчення основ Android-розробки;
- вивчити технологію мобільної розробки на Xamarin;
- формування вимог до функціонала і інтерфейсу додатку;
- проектування структури проекту;
- безпосередня реалізація проекту;

Методи дослідження: для вирішення цього завдання буде побудована модель ведення фінансів: короткий опис назви транзакції, що вноситься в систему, сума (яка може бути позитивною, так і негативною), дата внесення транзакцій, більше докладної інформації щодо транзакцій у системі обліку.

Для написання програмного коду для цієї програми було створено середовище розробки Microsoft Visual Studio 2019.

1 РИНОК МОБІЛЬНИХ ДОДАТКІВ

1.1 Історія ринку мобільних додатків

Мобільний додаток - це комп'ютерна програма, створена спеціально для використання на мобільному телефоні, смартфоні або комунікаторі, яка призначена для виконання того чи іншого завдання. На перших мобільних пристроях було встановлено малу кількість мобільних додатків, що поставляються разом з програмним забезпеченням пристрою. Першим мобільним додатком можна вважати телефонний довідник - та частина програмного забезпечення апарату, яка впорядковувала контакти користувача.

У 1997 році Nokia 6110 включила вбудовану версію базової аркадної гри «Змійка», яку багато хто вважає першим мобільним додатком. Перший iPod також постачався з вбудованими іграми: Пасьянс та Тетріс.

Однак у 1983 році молодий Стів Джобс вперше задумав App Store або, принаймні, дуже базову його версію. Джобс уявляв місце, де програмне забезпечення можна було купити за телефонними лініями.[1]

До 2000-го року почався бурхливий розвиток ринку мобільного контенту в цілому і мобільних додатків зокрема. Поява нових технологій передачі даних за допомогою стільникового зв'язку (GPRS, EDGE) тепер дозволяє здешевити мобільний інтернет-трафік, що було певним стримуючим фактором зростання ринку мобільних технологій.

У 2000-х роках ринок мобільних пристроїв стільникового зв'язку стали поступово завойовувати смартфони і комунікатори. Маючи більш широкими можливостями і продуктивністю, вони відрізнялися від звичайних мобільних телефонів наявністю досить розвиненої операційної системи (Windows Mobile, Symbian OS, RIM, Android, Mac OS), яка є відкритою для розробки програмного забезпечення сторонніми розробниками.[2]

Починаючи з 2000-х років поширення мобільних пристроїв і додатків стрімко зростало, слідуючи за технологічним прогресом в даній області.

2007 рік став одним з найбільш важливих в історії розвитку кишенькових комп'ютерів і смартфонів. Саме в цьому році, 9 січня, на виставці Macworld Conference & Expo був представлений iPhone, що перевернув все представлення користувачів про смартфони. 2007-й став також роком анонса Android, який явно стався під тиском стрімко набираючого популярності продукту від Apple. Тоді Android фігурував лише у вигляді бета-версії комплекту для розробників (SDK), оснащеного емулятором.

2009 рік можна по праву вважати роком розквіту Android як мобільної ОС. Виробники мобільної техніки почали придивлятися до Android і анонсувати свої перші пристрої на його основі, Google продовжувала спішно допрацьовувати ОС, Латаючи множинні прогалини в її дизайні та функціональності.

Розвиток операційних мобільних систем відкривало можливості розвитку в області мобільних додатків. Операційні системи надавали все більший функціонал розробникам мобільних додатків. Мобільна розробка встала по затребуваності поруч з комп'ютерної та знайшла свою численну аудиторію.

AppAnnie - велика платформа мобільного аналітики підвела підсумки розвитку мобільного ринку на 2015 рік. Стало ясно, що в майбутньому веб-сайти вже не зможуть конкурувати з додатками. З кожним днем люди витрачають на додатки все більше часу і коштів, завантажуючи їх для використання в різних сферах життя. Впровадження таких пристроїв і платформ як Apple TV, AppleWatch і аналогічних їм розробок на базі Android, в майбутньому посприяє тому, що користувачі будуть ставати все більш «мобільними».

1.2 Для чого потрібні мобільні додатки для контролю власних кошт

Фінанси - невід'ємна частина нашого життя, а фінансова грамотність полегшує життя людей. Гарне програмне забезпечення для управління особистими фінансами може допомогти вам керувати вашими грошима. За даними статистики Федеральної резервної системи США майже половина дорослого населення в США не могли дозволити собі здійснювати незаплановані виплати в розмірі всього лише

чотирьох сотень доларів. За даними того ж дослідження майже 25% дорослих з роботою покривають свої щомісячні витрати. Фінансова картина не набагато краща для тих, хто живе в інших частинах світу. [3] Наприклад, у Великій Британії, середня заборгованість домашніх господарств перевищила 3000 фунтів за даними управління національної статистики. Це дослідження також показало, що майже кожен п'ятий британський резидент вважає свої заборгованості важким тягарем.

Взяти управління своїми особистими фінансами під контроль - це дуже серйозне питання. Наслідки занадто великі борги можуть бути серйозними: зіпсований кредитний рейтинг, вилучення автомобілів або викуп вашого будинку і навіть особисте банкрутство. Але вести облік своїх витрат досить важко. Занадто легко витратити трохи грошей то тут то там. Але перш ніж ви встигнете отямитися, всі ці маленькі витрати складуться в величезну суму.

Ось для чого потрібен бюджет. Добре складений бюджет може допомогти вам відстежувати, як ви витрачаєте ваші гроші, а також допомогти спланувати, як впорядкувати свої витрати. Хороший інструмент з планування бюджету може навіть допомогти вам розробити план заощаджень і випередити ваші витрати. Інші інструменти фінансового управління можуть допомогти вам зберегти або навіть інвестувати ваші гроші.

1.3 Аналіз ринку мобільних додатків для контролю власних кошт

На ринку мобільних додатків для контролю власних кошт є такі конкуренти, як «CoinKeeper», «Spendee», «Finkee», «MoneyWiz». Отже, розглянемо кожен додаток окремо і визначимо які функції, і на якому рівні вони надають.

Додаток «CoinKeeper» ідеально підходить тим, хто любить або хоче розвинути звичку докладно записувати всі статті витрат і доходів. У програмі передбачені дві основні категорії - готівкові та безготівкові гроші. Тут же можна планувати витрати, заздалегідь вказати їх розмір, розділивши за категоріями. Їх сума відображається в графі «У планах». Серед мінусів - неповний переклад додатку, деякі функції в програмі - платні. Наприклад, можливість вести сімейний

бюджет, детальна статистика, період за який ведеться облік, можна змінити тільки в платній версії. Набридає реклама як стороння, так і платних пакетів послуг всередині «CoinKeeper». Ще у додатку не вистачає коментарів до витрат (Рисунок 1.1).

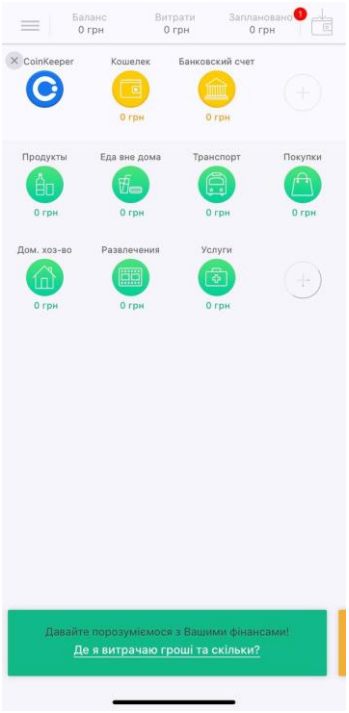


Рисунок 1.1 – Інтерфейс додатку «CoinKeeper»

Додаток «Spendee» має вельми незвичайний зовнішній вигляд - всі транзакції відображаються подібно стрічці в соціальних мережах. У програмі можна не просто записувати витрати, а вказувати конкретне місце, де вони були вчинені з назвою супермаркету чи ресторану. Можна навіть прикріпити фото цього місця. Також «Spendee» дає розширену статистику. Вона така ж, як і в «CoinKeeper», але доступна в безкоштовній версії. Одна з проблем додатку в тому, що в безкоштовній версії обмежена кількість категорій, фінансових акаунтів і бюджетів. Ще один мінус - відсутність нагадування про платежі і функції постановки цілей. Також як і в додатку «CoinKeeper» дуже поганий переклад (Рисунок 1.2).

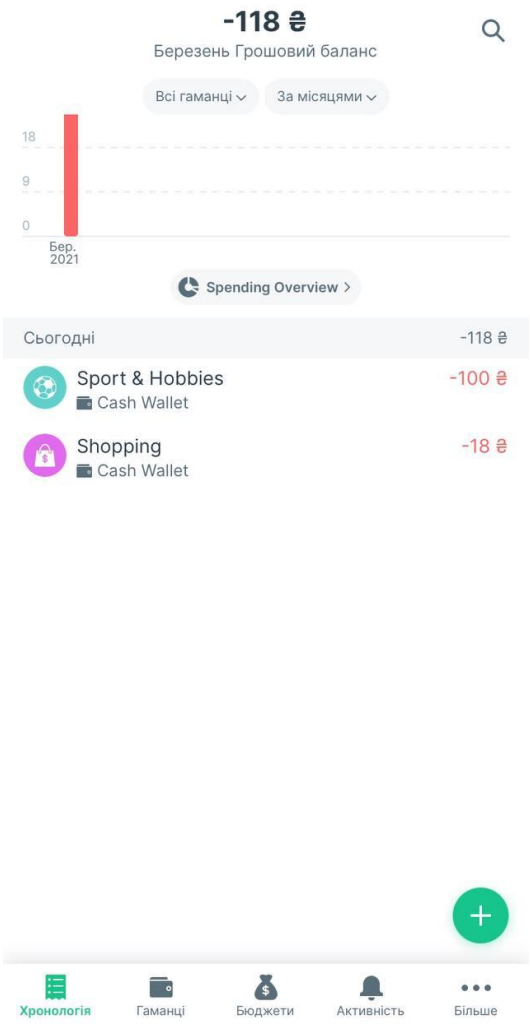


Рисунок 1.2 – Інтерфейс додатку «Spendee»

Додаток «Finkee» від Українських розробників з'явився на ринку недавно і ще не набрав своєї аудиторії, хоча і надає великий набір різноманітних функцій. Одної з яких немає в аналогічних програмах. Ви можете вести облік в криптовалютах. Головний мінус – відсутність інших мов крім англійської. Додаток незручний, введення транзакції гірше ніж в «CoinKeeper». До інтерфейсу важко адаптуватися, повний баланс намальовано маленькими літерами зверху які важко помітити з першого разу (Рисунок 1.3).

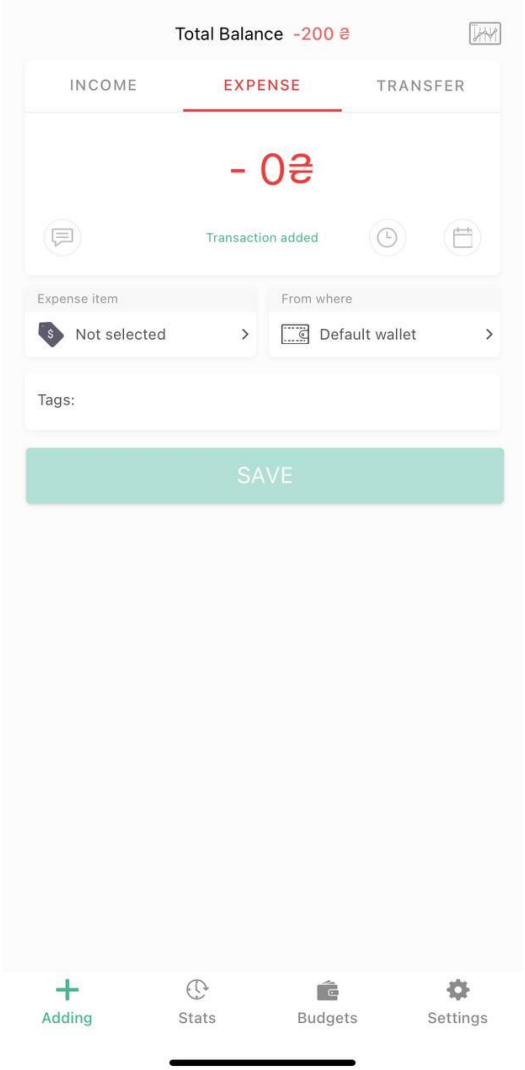


Рисунок 1.3 – Інтерфейс додатку «Finkee»

У додатку «MoneyWiz» є локалізація на 20 мовах, підтримка всіх світових валют та багаторівневі настроювання категорій. Всього в додатку більше 600 функцій, тому воно одне з найбільш універсальних у даній вибірці. Мінус тільки в підписці яка і надає доступ до даного додатку (Рисунок 1.4).

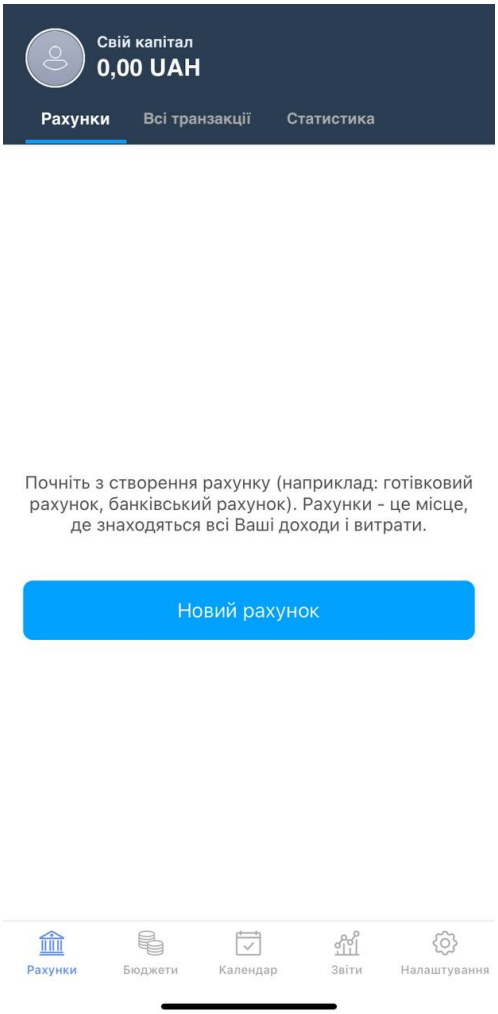


Рисунок 1.4 – Інтерфейс додатку «MoneyWiz»

1.5 Підсумки аналізу ринку. Постановка задачі

В ході аналізу ринку мобільних додатків з відповідною тематикою були виявлені можливості модернізації деяких наданих ними функцій, а так само додавання нових більш гнучких інструментів настройки роботи програми.

Одна з основних ідей роботи - це зручне відображення контенту. Користувач повинен легко «знаходити шлях» до потрібних йому функцій за кілька рухів пальців по екрану пристрою.

Тому були сформульовані наступні вимоги до основного функціоналу програми:

- інтуїтивно зрозумілий і зручний інтерфейс;

- створення записів витрат і доходів з можливістю застосування різних атрибутів: дата, категорія, сума, валюта, коментар та інше;
- редагування / видалення раніше доданих фінансових операцій;
- додавання / видалення / редагування категорій витрат і доходів;
- перегляд доходів і витрат з можливістю вибору часового проміжку часу;
- створення можливості вибору валюти для фінансових операцій;
- перегляд статистики витрат і доходів за допомогою кругових діаграм.

2 ОГЛЯД ВИКОРИСТОВУВАНИХ СИСТЕМ ТА ІНСТРУМЕНТІВ

2.1. Операційна система Android

Android - це операційна система, спроектована для роботи на смартфонах, планшетах та інших пристроях. Вона заснована на ядрі Linux, введено всього кілька оновлених понять і використовується більшість засобів Linux (віртуальна пам'ять, файлові системи, процеси, ідентифікатори користувачів, планування і т.д.).[4] Розроблено альянсом Open Handset Alliance (OHA), який зараз займається підтримкою і подальшим розвитком платформи. Основною мовою програмування серед розробників Android довгі роки вважалася Java , але в 2017 році Google оголосили основною мовою для створення додатків на Android Kotlin. Але крім цих двух мов також є можливість розробляти програми на Delphi, C# і т.д. У понад 83% смартфонах, проданих в 2018 році, була встановлена операційна система Android.

Android відрізняється від інших операційних систем підходом до універсалізації, використання єдиних схем управління компонентами апаратної платформи і взаємодії з кінцевим користувачем. Маючи декілька пристроїв на Android, можна мати на них ідентичний набір програм, контакти, календарі. При цьому користувачеві не потрібно прикладати ніяких зусиль.

Одна з головних переваг Android - це відкрита платформа. Вона доступна всім, і будь-який виробник мобільних пристроїв або розробник може

використовувати Android в своїх цілях: для створення додатків, пристроїв і навіть власної операційної системи. Це завжди буде її головною перевагою.

І останнє, але не менш важлива для користувача перевага на пристроях з встановленою ОС Android на сьогодні можна вибрати телефон з дійсно гігантського списку пропозицій. Дана ОС є відкритою і безкоштовною, тому конкуренція на ринку виробників мобільних пристроїв йде, що називається, до смерті. На Android доступні як смартфони бюджетного рівня, так і продукти елітного класу.

2.2 Xamarin

Xamarin - це платформа розробки мобільних додатків для створення нативних додатків iOS, Android і Windows із загального коду C # або .NET, яка дозволяє багаторазово використовувати між платформами від 75% до майже 100% коду.[5]

Програми, написані за допомогою Xamarin і C #, мають повний доступ до інтерфейсів API базової платформи і можливість створювати нативні призначені для користувача інтерфейси, а також компілювати код в машинний, тому вплив на продуктивність під час виконання є незначним.

Даний фреймворк включає в себе наступні частини:

• Xamarin.IOS - бібліотека класів для C # (для доступу до iOS SDK);

• Xamarin.Android - бібліотека класів для C # (для доступу до Android SDK);

• Компілятори для обох ОС;

• Середовища розробки - рідна IDE Xamarin Studio і плагін для Visual Studio.

Основа Xamarin - це open-source реалізація платформи .NET або Mono.

Вона включає в себе власний компілятор C #, середовище розробки та основний пакет .NET бібліотек. Дана технологія дозволяє запускати програми, написані на C #, на операційних системах, відмінних від Windows – Unix -системах, Mac OS і інших.

З точки зору виконання програми між iOS і Android є ключова відмінність - спосіб їх попередньої компіляції. В Android використовується Dalvik. Нативні додатки, які пишуться на Java, компілюються в свого роду проміжний байт-код, який Dalvik інтерпретує в команди процесора в момент виконання програми. такий вид компіляції називається Just-in-time (компіляція відразу). В iOS практикується інша модель - Ahead-of-Time (компіляція перед виконанням). Xamarin враховує особливості компіляції обох платформ і надає окремі компілятори для кожної з них. При цьому на виході виходять нативні додатки.

Для iOS, на відміну від Android з його Davlik, немає проміжного коду - він повинен бути заздалегідь скомпільований в машинний. Для цієї мети використовується AOT компілятор Mono.

2.3 Переносимість коду

Xamarin - це засіб крос-платформеної розробки, це означає, що додаток, написаний один раз, може бути запущено на різних мобільних платформах. Однак є певна особливість - при загальній логіці шар UI для кожної платформи потрібно писати окремо.

Якщо розбивати додаток на шари, то виходить така схема:

• Business Layer (BL) - Логіка додатка;

• Service Access Layer (SAL) - Взаємодія з віддаленими сервісами (наприклад, Json);

• Data Layer (DL) - Сховище даних (наприклад, SQLite)

• Data Access Layer (DAL) - Обгортка над сховищем для здійснення CRUD-операцій;

• Application Layer (AL) - Платформозалежний код (залежний від бібліотек monotouch.dll і monodroid.dll);

• User Interface Layer (UI) – Інтерфейс користувача.

Крос-платформеною є всі верстви, крім Application Layer і User Interface Layer. Бібліотека Xamarin.Mobile покликана для того, щоб збільшувати

платформонезалежніий код. Вона надає єдиний API для роботи з камерою і геолокації для різних платформ (Рисунок 2.1).[6]

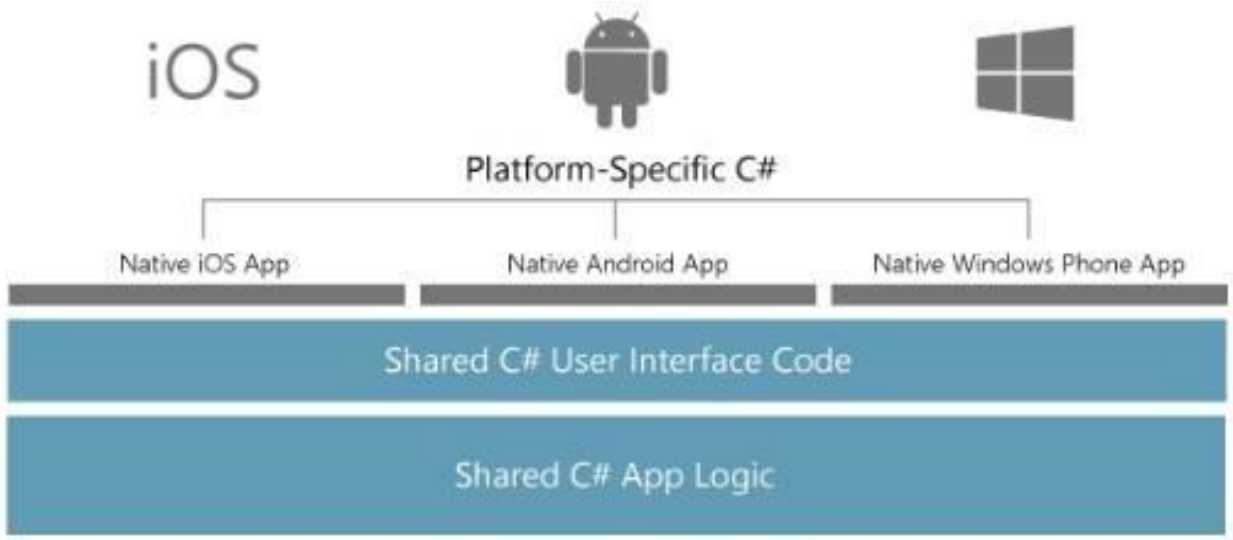


Рисунок 2.1 – Схема роботи кросплатформених додатків на платформі Xamarin

Xamarin Components Store

У Xamarin існує власний магазин сторонніх компонентів Xamarin Components Store. Він інтегрується в середовище розробки і дозволяє швидко і просто підключати до проекту різні компоненти, написані розробниками Xamarin і сторонніми розробниками. Магазин надає широкий вибір як платних, так і безкоштовних компонентів. Ці компоненти бувають двох видів: перший вид - це додаткові елементи призначеного для користувача інтерфейсу, а другий - бібліотеки класів. Наприклад, варіант для Mono відомої бібліотеки для роботи з Json - Json.NET. Не всі компоненти є крос-платформенню, деякі з них прив'язані до якоїсь однієї конкретної платформі. Xamarin використовує механізм Біндінг для зв'язування з нативними бібліотеками класів, що дозволяє перенести на C # будь-нативні бібліотеки класів. Для Xamarin.iOS існує спеціальна утиліта, яка автоматично генерує такі механізми зв'язування. Це дозволяє розробникам Xamarin встигати за всіма нововведеннями iOS. Таким чином, наприклад, в Xamarin.iOS майже відразу після виходу з'явилася можливість використовувати Dropbox API.[7]

IDE

Як середовище розробки можна використовувати або рідну Xamarin Studio, або Visual Studio.

Xamarin Studio – крос-платформена IDE, яка працює як на Mac OS X, так і на Windows. На вигляд ця вона виглядає дуже простий і доброзичливою, проте за зовнішньою простий ховається досить потужний інструмент, який включив в себе безліч фішок, звичних в Visual Studio і Resharper.

Xamarin надає можливість вести розробку в Visual Studio після установки спеціального плагіна. Для настройки роботи з iOS після встановлення плагіну для Visual Studio потрібно налаштувати з'єднання з Mac, яке буде використано при запуску проекту на виконання. Таким чином, після запуску програма автоматично буде пересилатися на Mac, де компілюватиметься і завантажуватися на пристрій або емулятор, однак процес налагодження, розстановка брейкпоінтів відбуватимуться в Visual Studio.

Працювати в Visual Studio можна кількома способами: перший - використовуючи віртуальну машину всередині Mac (наприклад, Parallels), куди ставиться Windows і Visual Studio, другий - використовуючи дві різні фізичні машини. Використовувати один Mac-пристрій для декількох PC-розробників важко, тому що налагодження вимагає маніпуляцій з симулятором. Третій спосіб - використовувати віртуальну машину з Mac OS X (так званий hackintosh). Є непоганим життєздатним варіантом, але має деякі обмеження (наприклад, в Xcode доведеться переміщатися по Storyboard тільки з використанням смуг прокручування, так як миша від Windows і миша від Mac - різні речі).

Розробники, знайомі з C #, .NET і Visual Studio, можуть розраховувати на такі ж можливості і продуктивність при роботі з Xamarin для мобільних додатків, включаючи віддалену налагодження на пристроях Android, iOS і Windows, без необхідності вивчати нативні мови, наприклад, Swift або Kotlin. Дивно, але багато високопродуктивних додатків з красивими користувача інтерфейсами - наприклад, NASCAR, Aviva і MixRadio - створені за допомогою Xamarin.

2.4 Activity

Одним з ключових компонентів при розробці мобільного додатку в Xamarin, зокрема при створенні візуального інтерфейсу є activity (активність).

Часто проводять паралель між активністю і окремим екраном або вікном додатка, а перемикання між вікнами в такому випадку пояснюють, як переміщення від однієї активності до іншого. Android-додаток може мати від однієї до кількох активностей. Всі об'єкти activity є об'єктами класу android.app.Activity, який містить базову функціональність для всіх activity. Головну активність або MainActivity прийнято наслідувати від класу AppCompatActivity, який, в свою чергу, побічно успадковується від базового класу Activity (Рисунок 2.2)

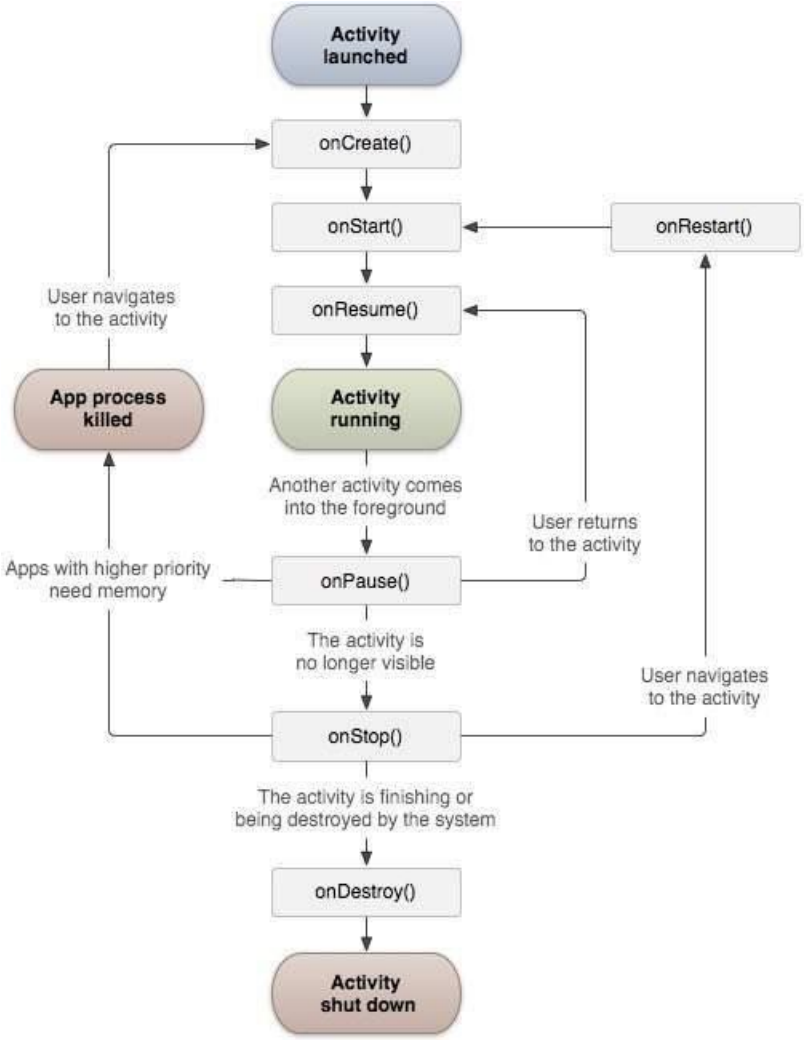


Рисунок 2.2 – Клас Activity та його життєвий цикл

2.5 Фрагменти

Додаток, що складається на основі кількох activity не завжди є оптимальним. Світ Android досить сильно фрагментований і включає в себе самі різні пристрої самих різних технічних характеристик. Якщо для мобільних пристроїв із середніми екранами використання декількох активностей виглядає непогано, то на планшетах і телевізорах зовнішній вигляд інтерфейсу буде помітно поступатися зовнішнім виглядом своїх побратимів менших розмірів. Концепція фрагментів і з'явилася в силу цієї причини.

Фрагмент має свій життєвий цикл, але він існує в контексті activity і поза контекстом його існування неможливо. Кожна активність може мати кілька фрагментів (Рисунок 2.3).[8]

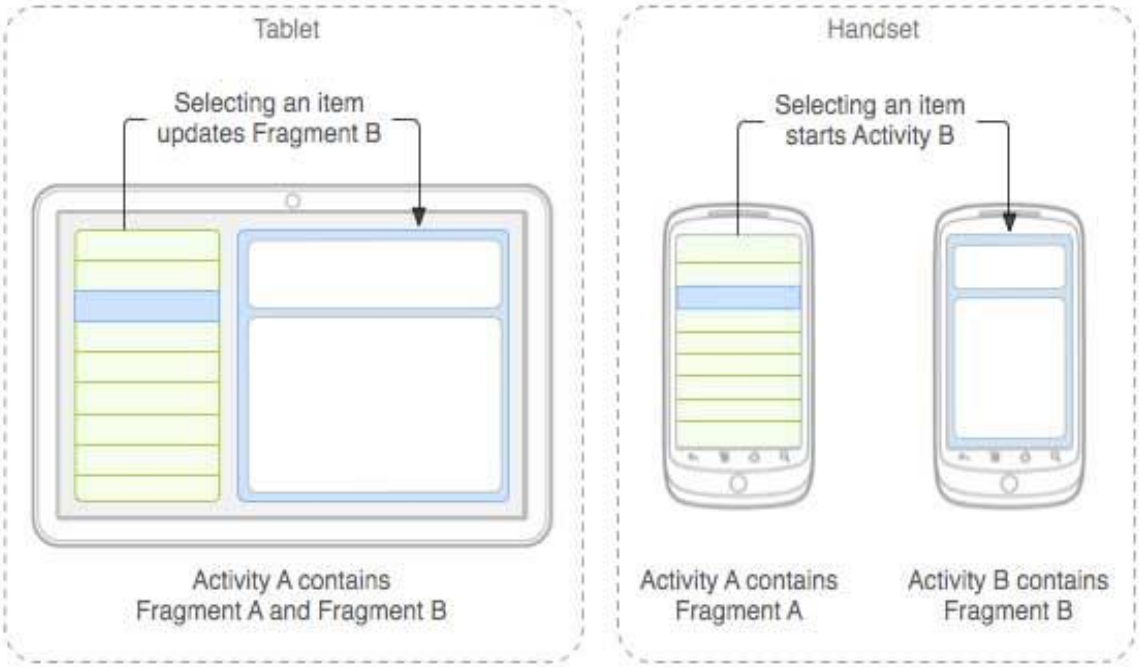


Рисунок 2.3 – Життєвий цикл фрагментів

Самі фрагменти успадковуються від android.app.Fragment. Існують різні підкласи фрагментів ListFragment, DialogFragment, PreferenceFragment, WebViewFragment і інші.

Для взаємодії між фрагментами використовується клас android.app.FragmentManager - спеціальний менеджер по фрагментах.

Цей клас не чинить роботу своїми руками, а використовує класи-помічники. Наприклад, для транзакцій (додавання, видалення, заміна) використовується клас android.app.FragmentTransaction.

Нижче представлені 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

Щоб система зрозуміла, що подальша робота буде здійснюватися з фрагментами, необхідно першим оголосити FragmentActivity (замість стандартного Activity).

В одному додатку можна використовувати нові фрагменти і фрагменти з бібліотеки сумісності.

Загальний алгоритм роботи з фрагментами:

У кожного фрагмента повинен бути свій клас. Клас успадковується від Fragment або схожих класів. Це схоже на створення нової активності або нового компонента.

За аналогією роботи з activity створюються різні методи типу onCreate () і так далі. Якщо фрагмент має розмітку, то використовується метод onCreateView () - це своєрідний аналог методу setContentView (). При цьому метод onCreateView () повертає об'єкт View, який є кореневим елементом розмітки фрагмента.

Розмітку для фрагмента можна створити програмно або декларативно через XML.

Створення розмітки для фрагмента нічим не відрізняється від створення розмітки для активності.

Клас фрагмента повинен успадковуватися від класу Fragment.

Для створення візуальної частини фрагмент переопределяет батьківський метод onCreateView (), який приймає три параметри:

- об'єкт LayoutInflater використовується для установки ресурсу розмітки для створення інтерфейсу

- параметр ViewGroup container встановлює контейнер інтерфейсу

- параметр Bundle savedInstanceState передає раніше збережений стан

Для створення інтерфейсу застосовується метод inflate () об'єкта LayoutInflater. Він отримує ресурс розмітки layout для даного фрагмента, контейнер, в який буде укладений інтерфейс, і третій булевий параметр вказує, чи треба прикріплювати розмітку до контейнера з другого параметра.

І як і в activity, тут ми можемо встановлювати обробники для елементів управління, обробляти їх подій. В даному випадку в текстовому полі виводиться поточна дата.

2.6 Патерн MVVM

Патерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку додатки від візуальної частини (подання). Даний патерн є архітектурним, тобто він задає загальну архітектуру програми.[9]

MVVM складається з трьох компонентів: моделі (Model), моделі подання (ViewModel) і уявлення (View) (Рисунок 2.4)

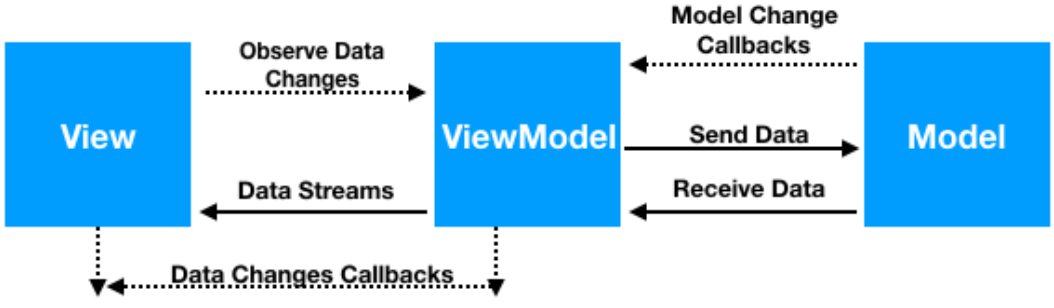


Рисунок 2.4 – Компоненти MVVM та їх зв’язок

Model

Модель описує використовувані в додатку дані. Моделі можуть містити логіку, безпосередньо пов'язану цими даними, наприклад, логіку валідації властивостей моделі. У той же час модель не повинна містити ніякої логіки, пов'язаної з відображенням даних і взаємодією з візуальними елементами управління. Нерідко модель реалізує інтерфейси INotifyPropertyChanged або INotifyCollectionChanged, які дозволяють повідомляти систему про зміни властивостей моделі. Завдяки цьому полегшується прив'язка до подання, хоча знову ж пряму взаємодію між моделлю і представленням відсутня.

View

View або подання визначає візуальний інтерфейс, через який користувач взаємодіє з додатком. Подання в Xamarin.Android - це код в xaml, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

Хоча вікно (клас Window) в WPF може містити як інтерфейс в xaml, так і прив'язаний до нього код C #, проте в ідеалі код C # не повинен містити якийсь логіки, крім хіба що конструктора, який викликає метод InitializeComponent і виконує початкову ініціалізацію вікна. Вся ж основна логіка додатки виноситься в компонент ViewModel.

Однак іноді в файлі пов'язаного коду все може знаходитися певна логіка, яку важко реалізувати в рамках патерна MVVM у ViewModel.

Подання і не виконує жодних події за рідкісним винятком, а виконує дії в основному за допомогою команд.

ViewModel

ViewModel або модель уявлення пов'язує модель і уявлення через механізм прив'язки даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде зміна відображуваних даних в поданні, хоча безпосередньо модель і уявлення не пов'язані.

ViewModel також містить логіку по отриманню даних з моделі, які потім передаються в уявлення. І також ViewModel визначає логіку по оновленню даних в моделі.

Оскільки елементи View, тобто візуальні компоненти типу кнопок, не використовують події, то View взаємодіє з ViewModel за допомогою команд.

Наприклад, користувач хоче зберегти введені в текстове поле дані. Він натискає на кнопку і тим самим відправляє команду під ViewModel. А ViewModel вже отримує передані дані і відповідно до них оновлює модель.

Підсумком застосування патерну MVVM є функціональний розподіл програми на три компонента, які простіше розробляти і тестувати, а також в подальшому модифікувати і підтримувати.

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

Для додатків iOS потрібно врахувати деякі специфічні моменти. Хоча Xamarin Studio для Mac надає все, що потрібно для розробки під iOS, користувачам Visual Studio на ПК все одно буде потрібно встановити Mac з інструментарієм Xamarin. Це дає можливість компілювати програми по мережі і тестувати їх в iOS Simulator або на пристрої iOS.

Xamarin дозволяє компілювати програми iOS і Android, написані на C # або F #, але використовують традиційний шаблон Model-View-Controller. При бажанні поліпшити тестованого, можливості в супроводі і портуванні, потрібен якийсь спосіб перенесення шаблону MVVM на ці платформи. Це MvvmCross.

MvvmCross для додатків Xamarin

MvvmCross – кросплатформена інфраструктура MVVM з відкритим вихідним кодом. Вона доступна для додатків Windows Phone, Windows 8, iOS, Android і WPF. MvvmCross вводить шаблон MVVM на платформи, де раніше його не було, наприклад, на iOS і Android.

Вона також підтримує зв'язування з даними в поданнях. Це потужний функціонал, що забезпечує поділ обов'язків (separation of concerns). View буде використовувати різні ViewModel, щоб забезпечувати потрібні поведінки в

додатку. MvvmCross навіть знаходить ці ViewModel в виділеному проект, щоб можна було легко посилатися на них і повторно використовувати в інших додатках.

Це найважливіше в MvvmCross. Знаходячи різні ViewModel в Portable Class Library (PCL), можна додавати їх як посилання в будь-які інші проекти. Звичайно, це не єдина цікава особливість MvvmCross. Ця інфраструктура також має архітектуру на основі плагінів, підтримує вбудовування залежностей (dependency injection, DI) і ін.

Застосування MvvmCross в Android / iOS

Застосовувати MvvmCross легко, тому що вона складається всього з декількох NuGet-пакетів, які додаються в проекти. Після цього потрібно виконати кілька простихопераций перед запуском програми. Ці операції в iOS відрізняються від таких в Android, але все одно дуже схожі.

2.7 Зберігання даних

SQLite - це полегшене, що не використовує сервер, ядро баз даних з відкритим вихідним кодом, яке спрощує створення локальних баз даних і виконує операції над даними. Інформація зберігається в таблицях, а операції над даними можна виконувати написанням коду на C # і LINQ-запитів.[10]

SQLite відмінно підходить для крос-платформена розробки, оскільки це ядро портує бази даних. Ядро заздалегідь встановлюється як в iOS, так і в Android і може бути легко розгорнуто в Windows. З цієї причини SQLite також є відмінним супутнім компонентом при створенні крос-платформених, орієнтованих на дані мобільні додатки за допомогою Xamarin.Forms, якій потрібна локальна база даних.

Створивши проект, буде потрібно керований спосіб доступу до баз даних SQLite.

SQLitePCL може бути використана для реалізації локальної бази даних в додатках для Windows, Windows Store, Windows Phone, Android (Xamarin) і iOS (Xamarin). Вона безкоштовна і її код відкритий для всіх бажаючих.

Для роботи з SQLite можна використовувати різні підходи і бібліотеки, проте рекомендованої є SQLite.NET, яка представляє просте ORM-рішення (Object Relational Mapping) для Xamarin.

Вона дозволяє працювати з базою даних як зі сховищем об'єктів і маніпулювати даними як об'єктами стандартних класів C # без використання виразів на мові SQL.

В головний проект у вузол References за допомогою менеджера пакетів NuGet потрібно додати SQLite.NET PCL. Треба зауважити, що в NuGet існує безліч бібліотек з подібною назвою та схожою функціональністю. У потрібної бібліотеки будуть наступні дані: автор - Frank A. Krueger, id

- sqlite-net-pcl.

При роботі з SQLite в Xamarin є один мінус - повний шлях до бази даних на кожній окремій платформі буде відрізнятися. Щоб вирішити дану проблему, необхідно використовувати механізм впровадження залежностей (dependency injection).

2.8 Пакети Nuget

Nuget - це одне з розширень Visual Studio, яке дозволяє з легкістю встановлювати, оновлювати і видаляти бібліотеки (збірки), компоненти, інструменти.[11]

Він бере на себе всі кроки цього процесу, в тому числі:

• пошук бібліотеки;

• завантаження необхідного для установки пакета;

• перевірка його хеш-значення на відповідність із заданим сервером (перевірка цілісності);

• розпакування файлів бібліотеки в потрібне місце;

• додавання посилань (reference) на її збірки в проект;

• модифікацію файлів конфігурації (web.config, app.config) при необхідності.

Все це здійснюється з NuGet в кілька натискань кнопки миші або введенням однієї консольної команди. При цьому враховуються всі залежності завантажується бібліотеки від інших компонентів. Останні, при необхідності, також будуть завантажені і додані в проект.

У розробці свого застосування я використовував такі пакети:

MvvmCross, MvvmCross.Binding, MvvmCross.Platform, MvvmCross.Core, MvvmCross.Plugin.SQLitePCL, sqlite-net-pcl, Xamarin.Android.Support.Compat, Xamarin.Android.Support.Core, Xamarin.Android.Support.Fragments, Xamarin.Android.Support.Core.UI, Xamarin.Android.Support.v4, Xamarin.Android.Support.v7.AppCompat

2.9 Material Design

Для того щоб програми були зручними і приємними для користувача, вони повинні слідувати принципам Material Design [7]. За допомогою специфікації кольорів Material Design були зроблені стилізовані теми додатку, де кольори панелі програми, стану і контекстні панелі повинні поєднуватися з кольорами заголовків, текст на кнопках і елементами інтерфейсу (checkbox і текстові поля). Рекомендовано застосовувати для яскравості значення 500 для основного кольору і 700 для панелі стану з однієї кольорової палітри. Ще один інструмент для підбору кольорів - це системна палітра, де можна вибрати два кольори і подивитися, як буде виглядати додаток, і нижче буде відображатися список кольорів, які будуть гармонічно виглядати в цій колірній палітрі.

Також Material Design дає рекомендації щодо поліпшення розташування елементів екрану і поради за розміром тексту на складових екрану. У специфікації Material Design відображений найкращий досвід по дизайну додатка. Специфікації представляють собою керівництво по вибору стилю, зображень, анімації і т.д. Для читабельності тексту в розділі по типографії були вказані поради щодо вибору розміру тексту. У керівництві Material Design рекомендовано застосування інтервалів в 8dp. Вирівнювання компонентами екрану уздовж однієї лінії спрощує

читання і роботу з додатком. Так, додаток буде кастомізоване за допомогою колірного дизайну, розстановки відступів і кордонів, а також грамотного розташування об'єктів на екрані.

Material Design по суті є комплексною, кросплатформенною концепцією, розробленою Google для створення додатків з візуалізацією і рухомими елементами. Це ціла філософія дизайну для спілкування розробника з користувачем. Дотримуючись вище перерахованим рекомендаціям Material Design, можна створювати додатки з унікальною графікою, красиві, зручні і інтуїтивно зрозумілі інтерфейси.

3 ПРОЕКТУВАННЯ ПРОЕКТУ

3.1. Розробка діаграм

Діаграма прецедентів (варіантів використання) відображає відносини між акторами і прецедентами системи і дозволяє описати систему на концептуальному рівні. Прецедент - можливість модельованої системи, частина її функціональності, завдяки якій актор може отримати конкретний, вимірний і потрібний йому результат. Прецедент відповідає окремому сервісу системи, визначає один з варіантів її використання і описує типовий спосіб взаємодії користувача з системою. Діаграма прецедентів зазвичай застосовується для специфікації зовнішніх вимог до системи

В ході проектування був виділений наступний актор.

Користувач - користувач додатка, якому доступна можливість використання всіх його функцій.

На основі функціональних вимог до мобільного додатку була створена діаграма прецедентів (Рисунок 3.1).

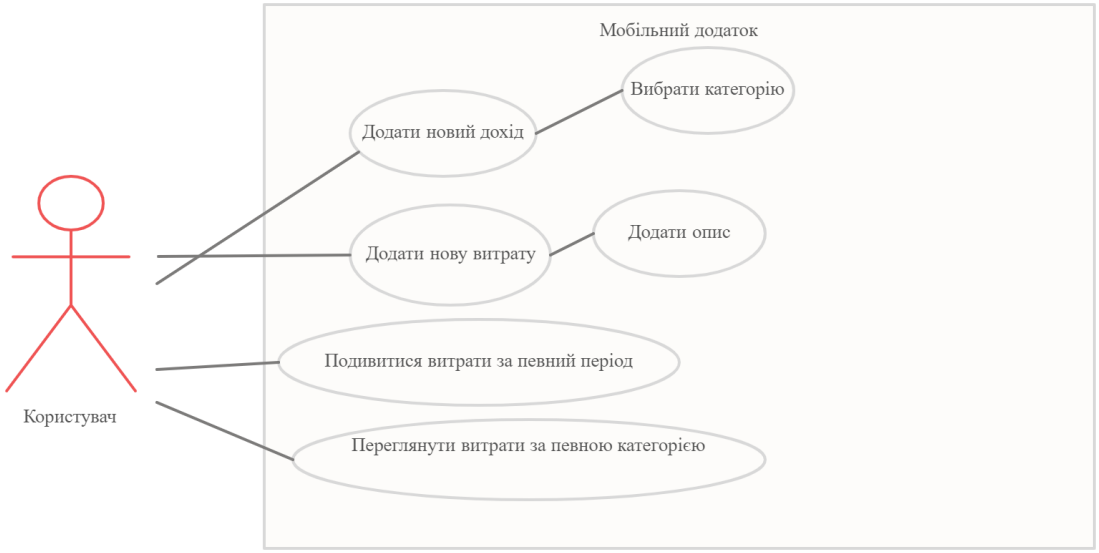


Рисунок 3.1 – Діаграма прецедентів

Додати новий дохід – додавання нової кількості доходів у світовій валюті.

Вибрати категорію – вибирається категорія з якої був отриманий дохід: зарплата, депозит, збереження.

Додати нову витрату – додавання нової витрати з категорій: Їжа, Житло, Здоров’я, Одежа, Транспорт, Подарунки.

Додати опис – додається любий опис витрати.

Подивитися витрати за певний період – переглянути статистику своїх витрат за вибраний період часу: День, Тиждень, Місяць, Рік, Весь час.

Переглянути витрати за певною категорією – перегляд статисти по витратам за певними категоріями.

Діаграма послідовності (sequence diagram) - UML-діаграма, яка представляє взаємодію між лініями життя і упорядкована послідовність подій. На Рисунку 3.2 представлена діаграма послідовності, що описує процес додавання доходів та витрат, згідно з сформульованими вимогам до мобільного додатку.

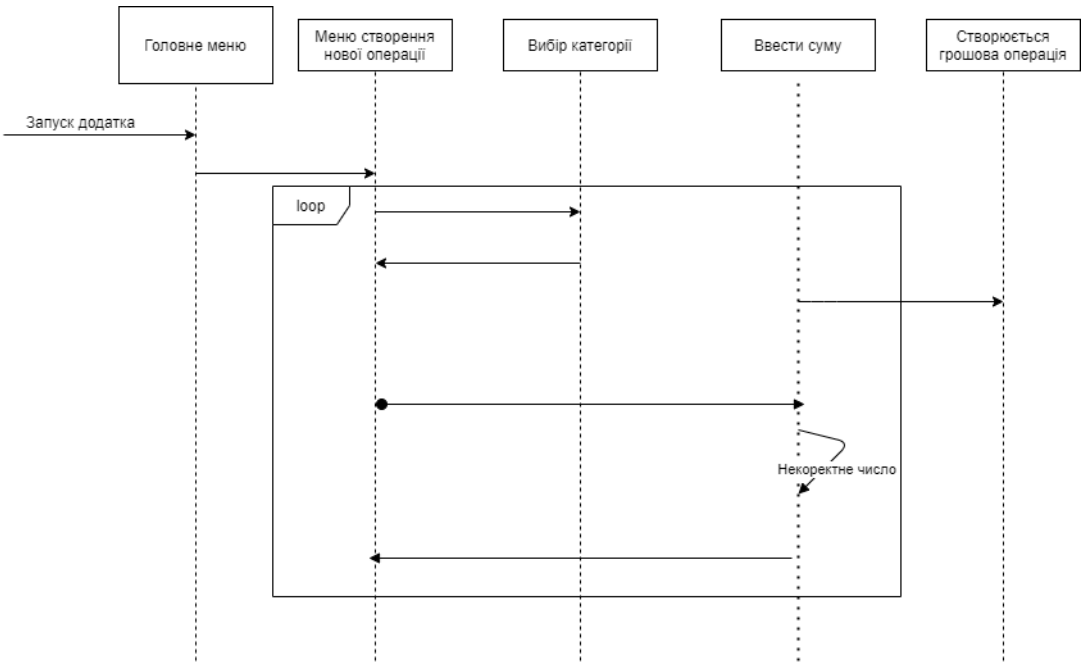


Рисунок 3.2 – Діаграма послідовності

Також була створена діаграма діяльності для повного бачення процесів які робить користувач коли користується додатком (Рисунок 3.3). В даному процесі користувач додає витрати або доходи у полі «Додати», потім потрібно вибрати категорію. Після додавання всіх необхідних доходів чи витрат, користувач натискає кнопку «Додати дохід/витрату».

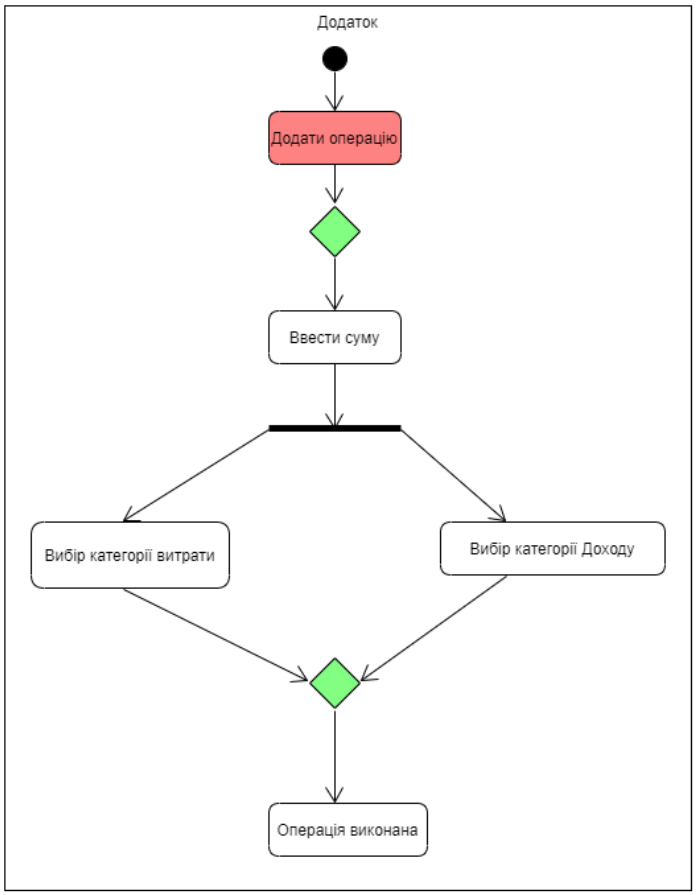


Рисунок 3.3 – Діаграма діяльності

Для того, щоб створити якісний додаток, важливо розуміти, які саме зв’язки об’єктів між собою. А для цього потрібно знати, які об'єкти потрапляють в предметну область проектованого додатку і які логічні зв'язки між ними існують. Для формування такого розуміння використовуються діаграми предметної області.

Метою побудови логічної моделі є отримання графічного представлення логічної структури досліджуваної предметної галузі.

Логічна модель предметної галузі ілюструє суті, а також їх взаємовідносини між собою (Рисунок 3.4)

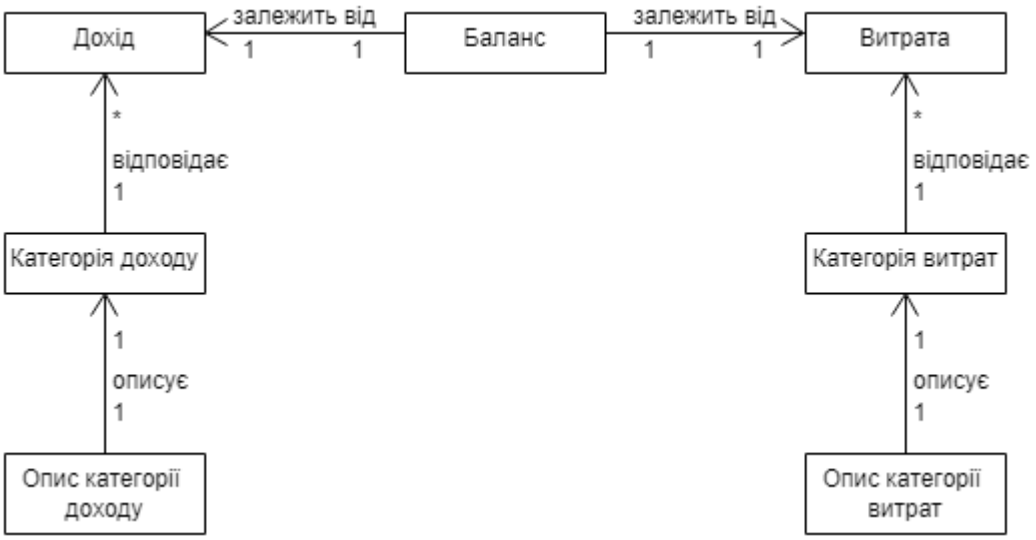


Рисунок 3.4 – Діаграма предметної галузі

Була розроблена діаграма артефактів (Рисунок 3.5) для демонстрації логіки інтерфейсу користувача.

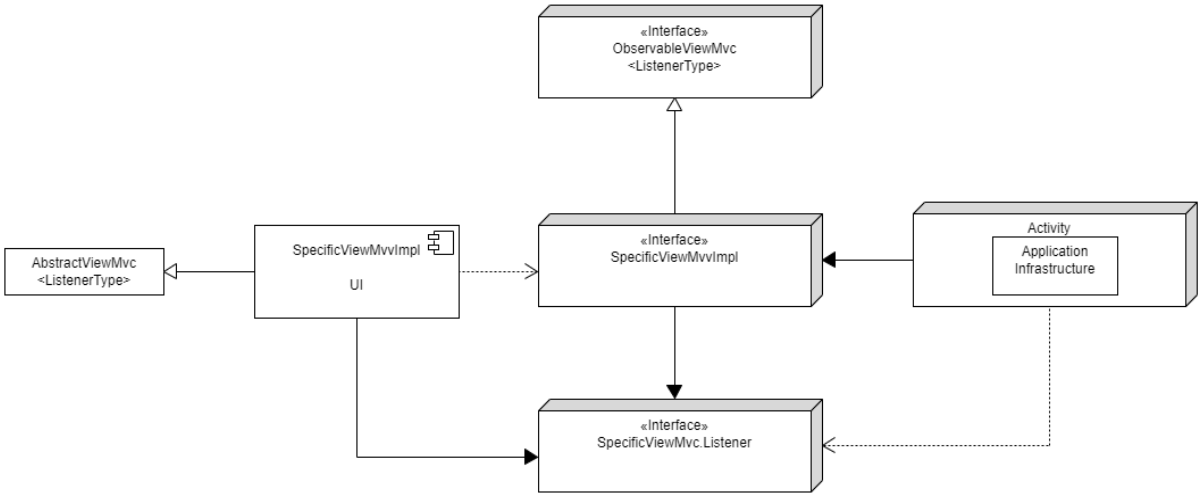


Рисунок 3.5 - Діаграма артефактів

Для кращого проектування додатку була створена діаграма пакетів (Рисунок 3.6). Пакети не дають відповіді на питання, яким чином можна зменшити кількість залежностей в системі, що розробляється, проте вони допомагають виділити ці залежності. Зведення до мінімуму кількості залежностей дозволяє знизити зв'язаність компонентів системи.

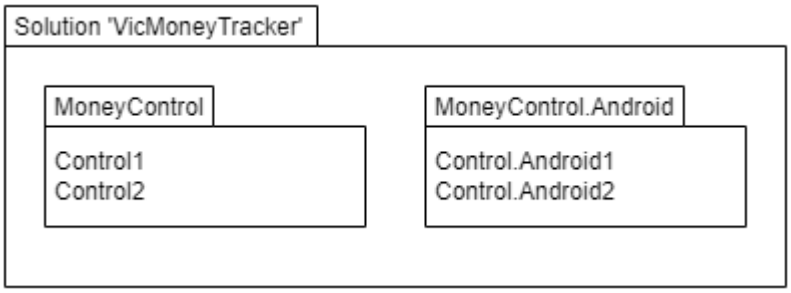


Рисунок 3.6 – Діаграма пакетів

3.2. Макет активностей додатку

На основі діаграм прецедентів та послідовності, а також дотримуючись вимог до функціоналу додатку, був спроектований макет проекту, графічно описано те, як буде виглядати додаток: його вікна і компоненти. На даній моделі було описано подальше розміщення елементів інтерфейсу програми: кнопки, списки, що випадають, поля введення, іконки і т.д.

3.2.1. Головне вікно. Доходи. Категорії доходів.

Головне вікно відображає поточні доходи, витрати і баланс, також воно виконує функцію навігації по основним вікнам додатка.

При переході в розділ «Доходи» відкривається вікно, в якому реалізується додавання інформації про доходи.

Натискаючи на «Категорію» при додаванні доходів, відбувається перехід у вікно «Категорії доходів», де відображається список категорій доходу. Тут також можна створити нову категорію або редагувати вже нинішню категорію (Рисунок 3.7).

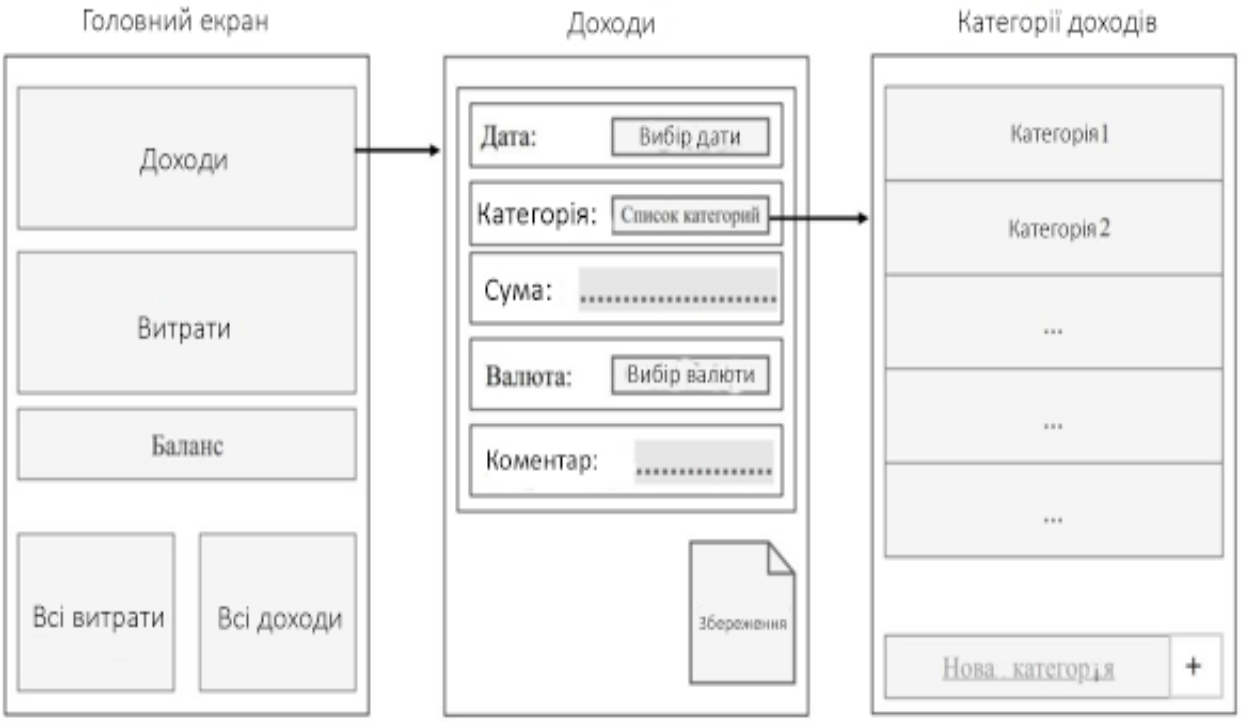


Рисунок 3.7 - Макети головного вікна, доходів і категорій доходів

3.2.2. Всі доходи. Діаграма доходів. Список доходів по категорії

При натисканні на кнопку «Всі доходи» на головному вікні відкриється одноіменне вікно з інформацією про суму доходів за вказаний часовий період. Натиснувши на кнопку «Діаграма», відкриється нове вікно, де буде зображена діаграма, що відображає відсоткове співвідношення суми доходів за категоріями.

Також натиснувши на певну категорію в розділі «Всі доходи» можна побачити список записів з цієї категорії. Є можливість відредагувати або видалити записи (Рисунок 3.8).

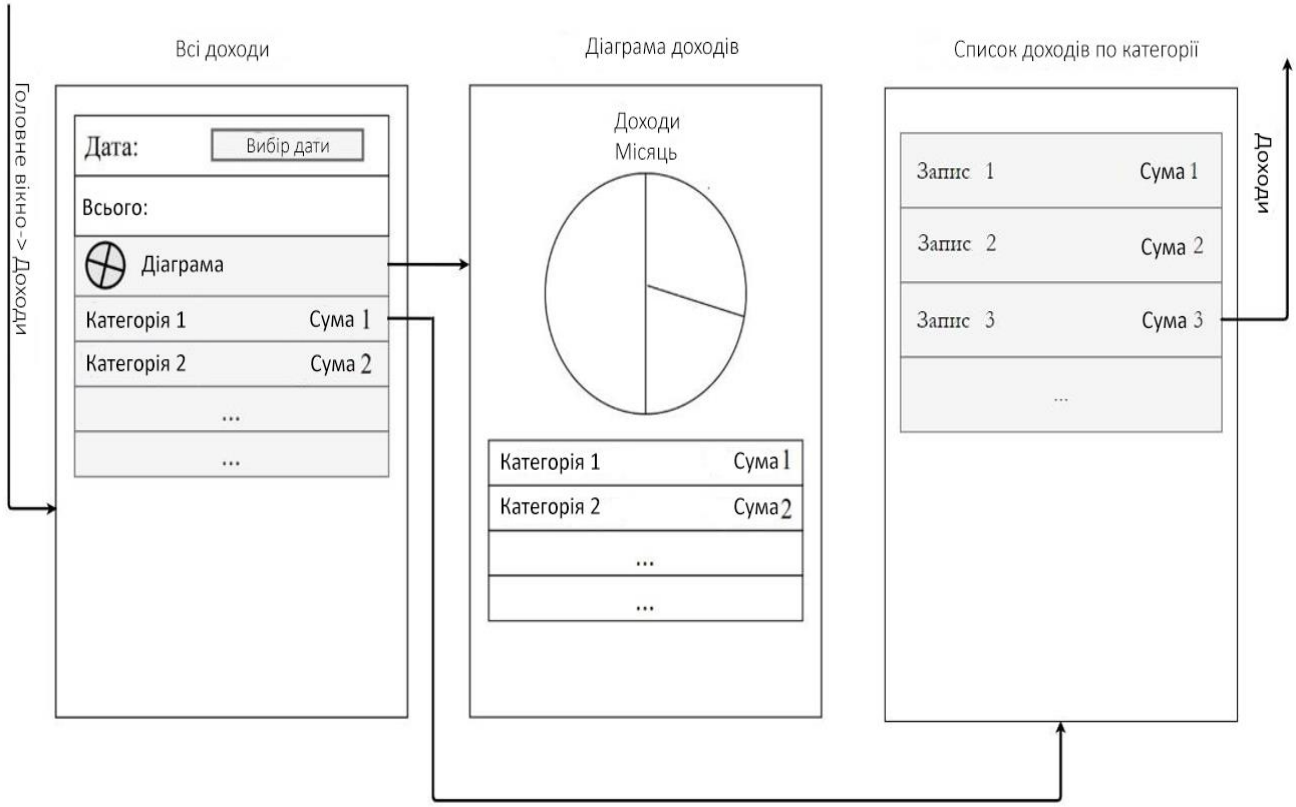


Рисунок 3.8 - Макет всіх доходів, діаграми доходів і списку доходів по категорії

3.2.3. Витрати. Категорії витрат. Баланс.

Вибравши розділ «Витрати», відкривається активність з додаванням операцій за видатками. Натискаючи на «Категорію», відбувається перехід у вікно

«Категорії витрат», де відбивається список категорій, де потрібно вибрати конкретну категорію або створити самому, також є можливість відредагувати або видалити категорії.

При переході в розділ «Баланс» з головного вікна, відображається статистика по місяцях за видатками та доходами разом (Рисунок 3.9).

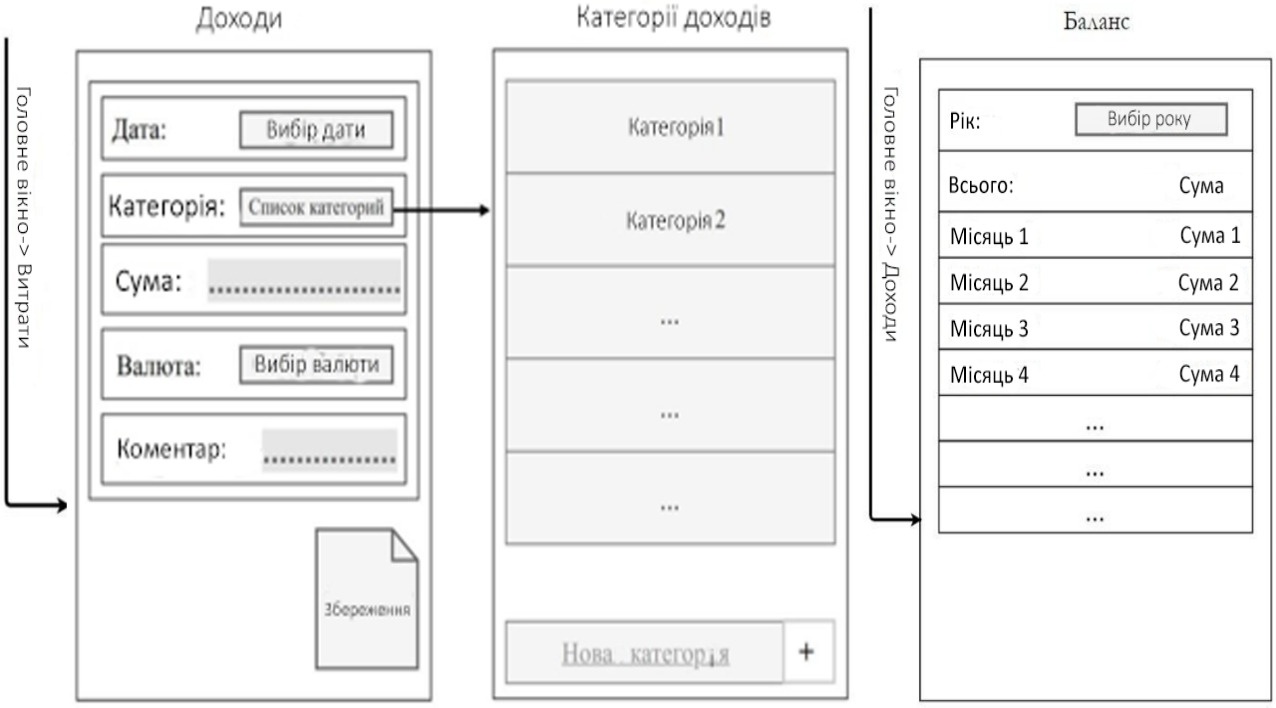


Рисунок 3.9 - Макет витрат, категорій витрат і балансу

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

При переході в розділ «Усі витрати» з головного вікна, користувач отримує інформацію про витрати за категоріями за вказаний часовий проміжок. Натиснувши на кнопку «Діаграма», відкриється активність «Діаграма витрат», на якій буде представлена кругова діаграма, представляє собою процентне співвідношення суми за категоріями витрат.

Натиснувши на категорію у вікні «Усі витрати», в новому вікні буде відображатися перерахування витрат за цим типом категорії. Є можливість внести зміни в записах (Рисунок 3.10).

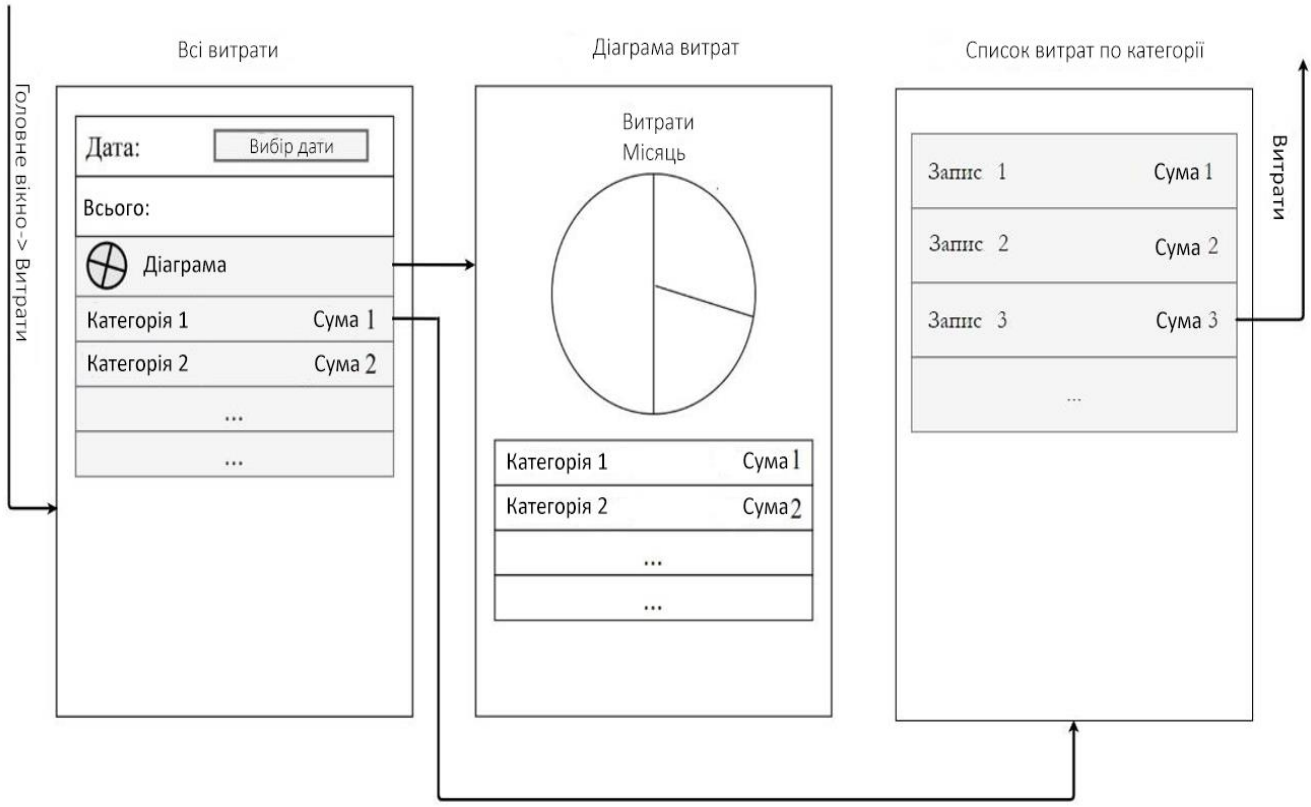


Рисунок 3.10 - Макет всіх витрат, діаграми витрат і списку витрат по категорії

4 ТЕСТУВАННЯ ТА РЕАЛІЗАЦІЯ ДОДАТКУ

4.1 Тестування додатку

У процесі розробки програми проводилося поетапне тестування. Для цього були створені емулятори смартфона і планшета з різними діагоналями екрана для різних версій Android. Тестований програмний продукт послідовно запускався на цих емуляторах, його поведінка аналізувалася, і при необхідності за результатами аналізу вносилися зміни в код.

Були проведені наведені нижче тести:

1. У поля вводу навмисно вносилися неприпустимі дані, які могли бути невірно інтерпретовані програмою. Потім мною аналізувалася поведінка активності під час обробки неприпустимих даних.

2. Додаток було запущено на пристроях, що працюють під керуванням різних версій Android з метою виявлення особливостей роботи програми, запущеного в різних операційних системах.

3. Після завершення циклу розробки, програмний продукт тестувався на реальних пристроях таких як Xiaomi Mi A1, Xiaomi Redmi 8 Pro, Xiaomi Redmi 5a та One Plus 8.

4.2 Структура додатка

Виходячи з принципу MVVM (який докладніше описаний в розділі 2.6), мій додаток має наступну структуру (Таблиця 4.1):

Таблиця 4.1 – 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>

Де Core (або Ядро) - частина програми, в якій прописана вся логіка програми, а User Interface (інтерфейс користувача) - описує інтерфейс програми.

У свою чергу, в розділі Ядра прописуються класи для моделей і моделей уявлення, а в розділі UI - описуються класи, які представляють візуальну частину програми.

У моєму проекті структура MVVM визначена таким чином (Рисунок 4.1):

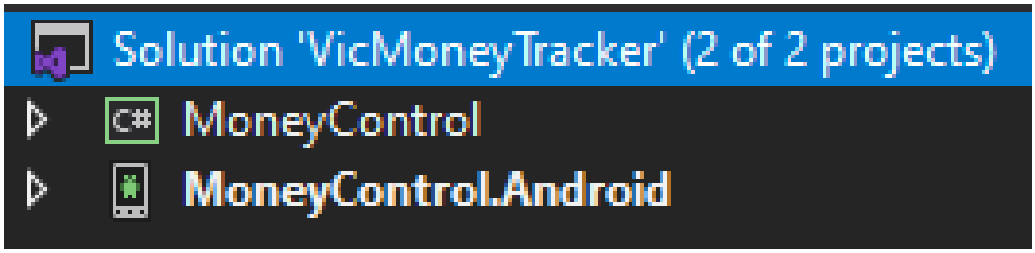


Рисунок 4.1 – Структура проекта

Нижче наведено короткий алгоритм дій по здійсненню цього принципу в моєму проекті:

Core

1) Створення PCL-проекту (Portable Class Library - для забезпечення переносимості коду і можливості в подальшому перебудувати його під інші ОС)

2) Установка в редакторі пакетів Nuget пакета MvvmCross

3) Додавання папки Model, в якій будуть зберігатися дві моделі - Transaction.cs і Wallet.cs

4) Додавання класу WalletViewModel.cs, в якому буде описана основна логіка програми, цей клас буде успадковуватися від MvxViewModel.

В даному класі будуть описуватися властивості доходів, витрат, поточного балансу, а також опис команд, за допомогою яких буде здійснюватися зв'язування елементів управління з логікою.

5) Додавання класу App.cs, успадковані від MvxApplication, в якому буде реєструватися стартова точка, і

який необхідний для запуску програми.

UI

1) Додавання пакета MvvmCross.

2) В папці Resources / layout будуть зберігатися уявлення, тобто візуальна частина програми. Інтерфейс в даному проекті, так як це не XamarinForms, здійснюється за допомогою розмітки xml.

У файлах xml розмітка сторінки буде задаватися шляхом опису всіх кнопок, елементів TextBox і інших, там же відбувається зв'язування елемента управління з його логікою. Наприклад, кнопка описується так:

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

1) Створення папки Views, де буде міститися Activity. Головна Activity повинна містити метод OnCreate (), в якому і буде прописуватися, яке вікно відкривається при запуску програми.

2) Створення класу Setup, який буде посилатися на проект Core і дасть знати виконуючою середовищі, як створити екземпляр програми (Рисунок 4.2):

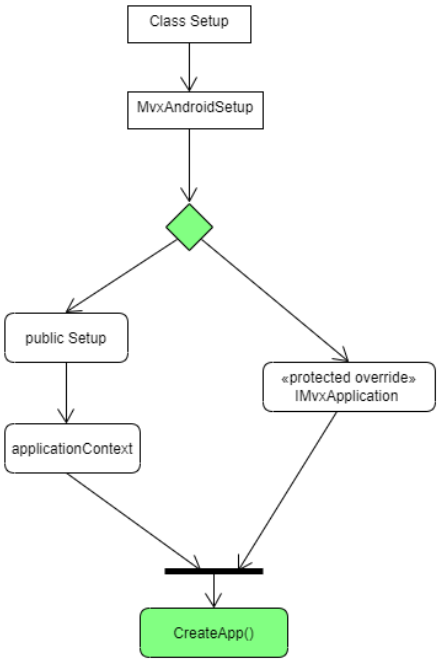


Рисунок 4.2 – Створення класу Setup

4.3 SQLite

SQLite - компактна вбудована реляційна база даних. «Вбудована» означає, що SQLite не використовує парадигму клієнт-сервер, тобто движок SQLite не є окремо працюючим процесом, з яким взаємодіє програма, а надає бібліотеку, з якої програма компонується і движок стає складовою частиною програми. SQLite має низку наступних переваг:

- надійність (при випуску версії вона проходить через ряд найсерйозніших автоматичних тестів (проводиться ~ 2 млн тестів), покриття коду тестами 100%)

- простота і зручність вбудовування (доступний на будь-якому Android пристрої і не вимагає окремої установки)

- висока продуктивність (для більшості типових завдань додаток, побудоване на SQLite, працює швидше, ніж при використанні MySQL, в 2-3 рази і швидше PostgreSQL в 10-15 разів)

SQLite чудово підходить для Android-додатків, коли основна маса запитів являє собою запити на читання, також дуже добре підійде для невеликих проектів. Так як SQLite працює з такими ж SQL запитами, що і бази даних серверного типу, це дає можливість легко перейти на іншу базу даних, не вносячи кардинальних змін в логіку програми.

Використання таблиць:

1) Є багато бібліотек, що дозволяють працювати з базами даних SQLite в Microsoft .NET Framework, але нам потрібна спеціальна портована бібліотека, яка також розрахована на додатки Xamarin. Вона називається SQLite-net і є полегшеною бібліотекою з відкритим вихідним кодом для додатків .NET, Mono і Xamarin. Вона доступна у вигляді NuGet-пакета sqlite-net-pcl. Додамо цей пакет на повний проект (як в ядро, так і в призначений для користувача інтерфейс).

2) Код звертається до бази даних SQLite через рядок підключення. Оскільки база даних SQLite - це файл, що розміщується в локальній папці, конструювання рядки підключення вимагає вказати шлях до бази даних.

Формування рядка підключення вимагає коду, специфічного для платформи. Потім ви викликаєте рядок підключення через вбудовування залежностей (dependency injection).

```
public interface IDatabaseConnection

{

  SQLite.SQLiteConnection DbConnection();

}
```

Цей інтерфейс надає метод DbConnection, який буде

реалізований в кожному специфічному для платформи проект і який буде повертати належну рядок підключення.

3) Створюємо окремі класи для кожної таблиці. Кожен клас повинен

реалізувати інтерфейс INotifyPropertyChanged, що повідомляє викликають про зміни в зберігаються цим класом даних.

4) Здійснюємо виконання CRUD-операцій

Операції (create, read, update and delete, CRUD) також надзвичайно важливі. Об'єкт SQLiteConnection надає методи Insert, InsertAll, Update і UpdateAll для вставки або відновлення об'єктів в базі даних.

InsertAll і UpdateAll виконують операцію вставки або поновлення в наборі, який реалізує IEnumerable <T>, який передається в якості аргументу.

Операція вставки або поновлення виконується в пакетному режимі, і обидва методи також дозволяють виконувати операція в вигляді транзакції.

4.4 Реалізація інтерфейсу

Два найважливіших поняття в інтерфейсі Android - це Activity і View.

Activity - це та частина програми, з якою взаємодіє користувач. Можна назвати її «вікном» в термінології десктопних ОС. Усередині Activity розташовані дочірні елементи інтерфейсу.

View - елемент інтерфейсу (наприклад, кнопка, поле для введення тексту, контейнер для картинки і т.д.)

Так само важливий елемент - ViewGroup. Фактично це модифікований View, створений для того, щоб служити контейнером для інших View.

Layout - загальна назва для декількох спадкоємців ViewGroup. Лейаута служать контейнерами для View, і створені вони для того, щоб ми могли зручно розташовувати всілякі кнопочки, поля для введення тексту і інші елементи інтерфейсу.

Основні layouts:

• LinearLayout
• FrameLayout
• RelativeLayout

4.4.1 Головне меню

Головний екран представляє собою меню з інформацією про витрати та доходи, які розташовані у стовбчик по даті додання (Рисунок 4.3).

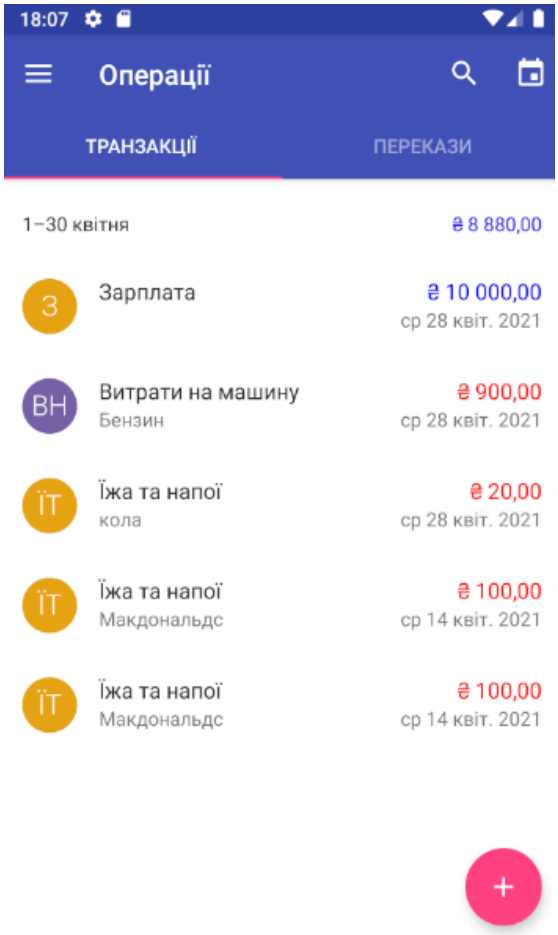


Рисунок 4.3 – Головне меню

4.4.2 Нова операція. Додавання доходів та витрат

Перехід на вкладку «Нова операція» (Рисунок 4.4) відбувається після натискання на кнопку з зображеним на ній знаком «+».

У вкладці «Нова операція» пропонується заповнити наступні поля: опис, категорія, дата, час, рахунок, подія, люди, місце.

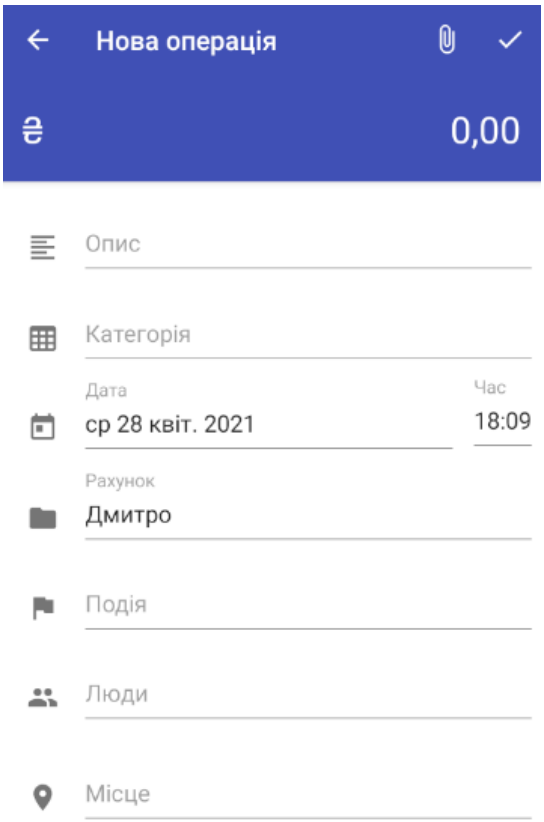


Рисунок 4.4 – Вкладка «Нова операція»

За замовчуванням дата та час - це момент коли був здійснений перехід на цю активність. Зміна дати реалізована за допомогою компонента CalendarView з розділу Widgets, який виводить на екран календар (Рисунок 4.5). Достатньо натиснути на кнопку із зображенням календаря.

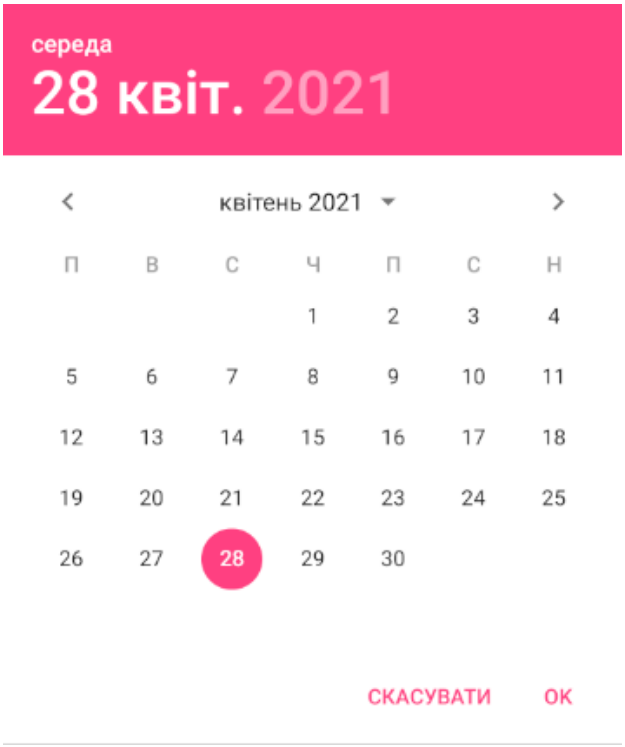


Рисунок 4.5 - Установка дати запису доходу

Для коректного введення суми використовується атрибут android: inputType = "NumberDecimal", це здійснює обмеження на введення буквених значень.

Опис є необов'язковим атрибутом при додаванні - за замовчуванням це поле пусте. При введенні опису кількість знаків 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 вікні категорії зверху є вибір додавання доходу або витрати. У вкладці Дохід представлені наступні категорії: аванс, зарплата, премія, стипендія, фріланс (Рисунок 4.6).

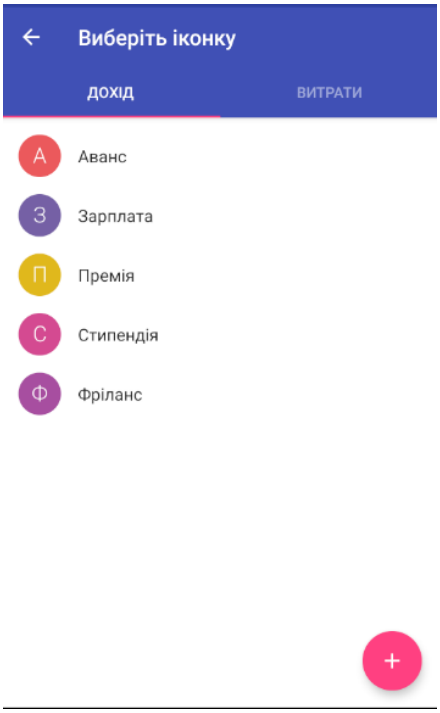


Рисунок 4.6 – Вибір категорії доходу

У вкладці витрати є наступні категорії: їжа та напої, витрати на машину, друзі, подорожі, технології, хоббі (Рисунок 4.7). Для вибору категорії необхідно натиснути на потрібну категорію.

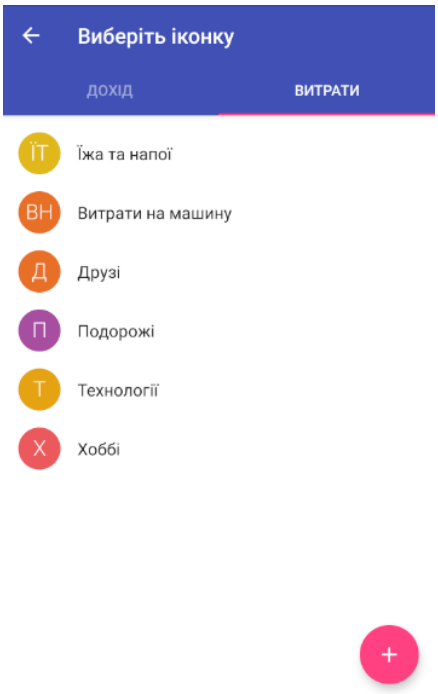


Рисунок 4.7 – Вибір категорії витрати

Після вибору категорії вікно категорій закривається і відкривається знову вікно для додавання операції.

Для зручності користувачів є можливість додати свої категорії. Для цього потрібно натиснути на кнопку «+». У цій активності здійснюється додавання в базу даних назв категорій, які будуть введені користувачем в поле введення. Потім при зберіганні відображається введене значення на екрані, це реалізовано компонентом RecyclerView, який спрощує роботу зі списком (Рисунок 4.8).

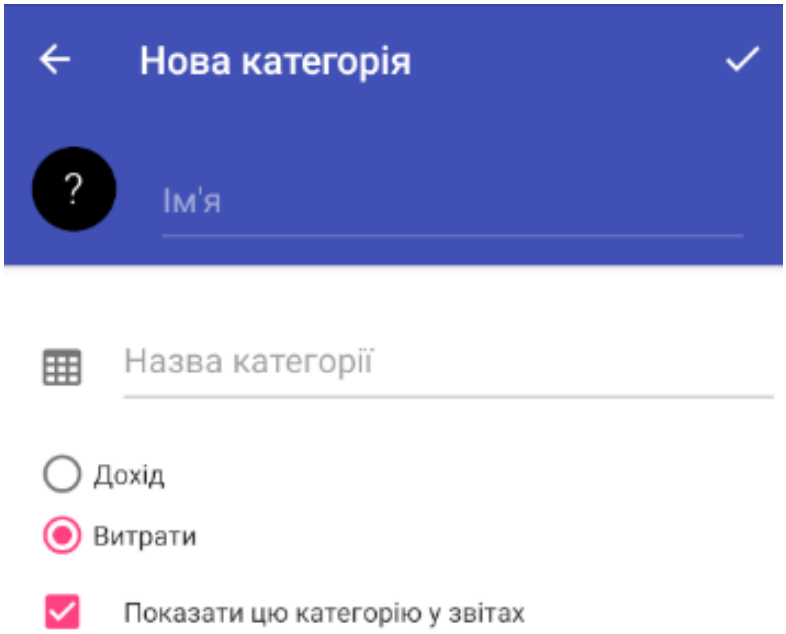


Рисунок 4.8 – Додавання своєї категорії

Для коректного введення обов'язковими атрибутом є введення суми. Введення суми відбувається завдяки влаштованому в додаток калькулятору де є можливість відразу підрахувати суму якщо це декілька товарів. Після введення потрібного числа необхідно натиснути на значок «=», після чого сума запишеться і відбудеться повернення до меню створення операції (Рисунок 4.9).

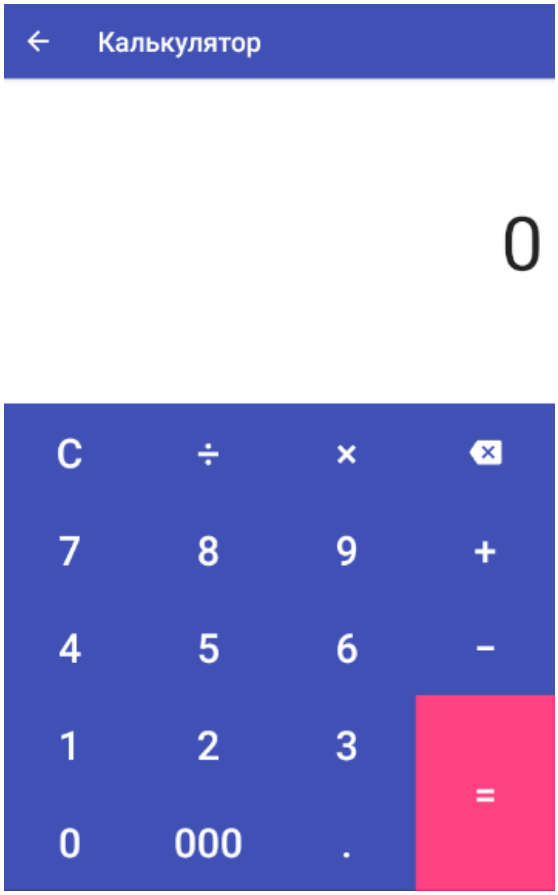


Рисунок 4.9 – Введення суми завдяки вбудованому калькулятору

4.4.3 Navigation Drawer – випадаюче меню

Для створення меню було використано Android Design Support Library. Саме випадаюче меню включає в себе такі пункти як: операції, категорії, огляд, місця, калькулятор, пошук банкоматів та пошук банків (Рисунок 4.10).

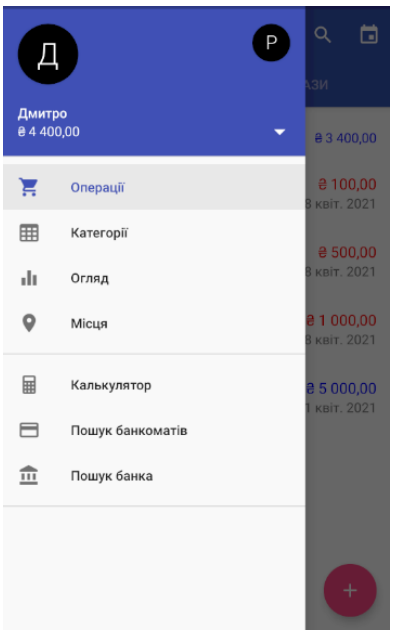


Рисунок 4.10 – Випадаюче меню

4.4.4 Реалізація діаграм для статистики

В активності «Огляд» розташовуються всі сумарні доходи та витрати за конкретний часовий проміжок (Рисунок 4.11). Ці значення вираховуються за допомогою запитів SQLite.

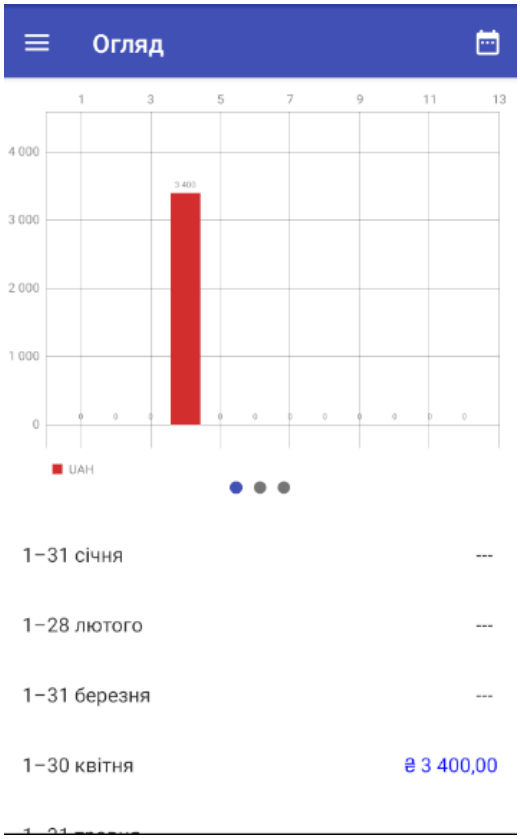


Рисунок 4.11 – Меню огляд

Побачити статистику за категоріями можна натиснувши на проміжок часу за який потрібно подивитися статистику. Тим самим відкриється нова активність «Діаграма». Створюється кругова діаграма, яка була реалізована завдяки бібліотеці OxyPlot, яка дозволяє малювати графіки різної складності. Діаграма відображає статистику, яка формується на основі інформації з бази даних за вказаний часовий період. У цьому меню можна окремо подивитися як діаграму доходів так і витрат і спільну діаграму (Рисунок 4.12).

Рисунок 4.12 – Кругова діаграма всіх витрат за вибраний проміжок часу

4.4.5. Список доходів та витрат по категорії

При натисканні на певну категорію у вікні «Категорії» відкривається список доходів або витрат в залежності від вибраної категорії. (Рисунок 4.13).

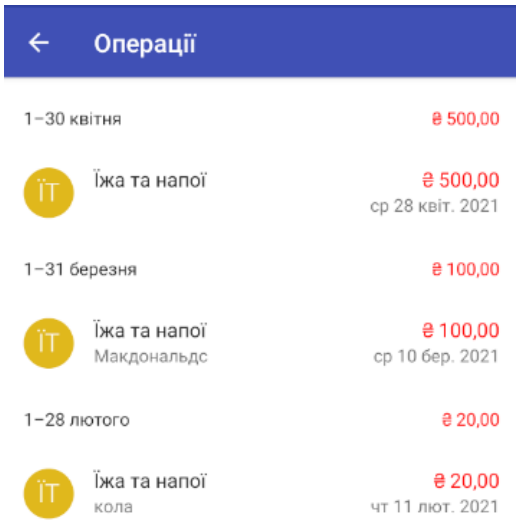


Рисунок 4.13 - Список записів витрати по конкретної категорії

Також є діалогове вікно для вибору редагування або видалення введених записів. Для цього потрібно просто натиснути на необхідний запис (Рисунок 4.14).

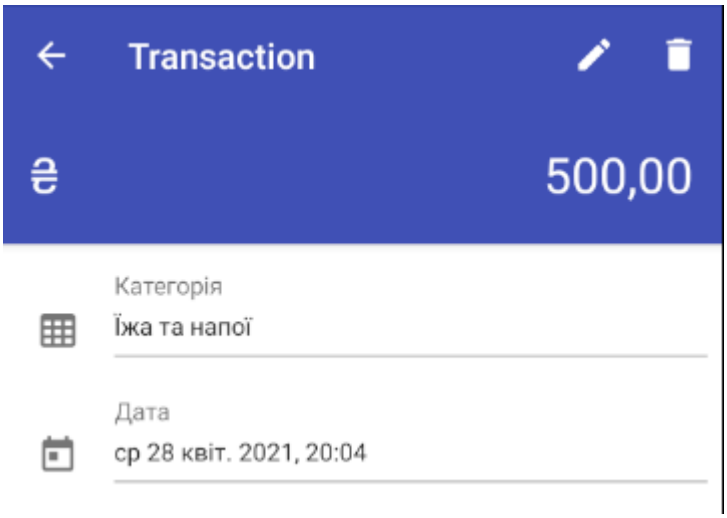


Рисунок 4.14 – Редагування запису

4.4.6 Інші можливості

Також були реалізовані деякі маленькі, але потрібні функції такі як: вибір місця де була здійснена операція, пошук банку та банкомата. Ці функції були реалізовані завдяки зверненню до гугл карт, але для їх роботи потрібен робочий інтерне

ВИСНОВКИ

1. Таким чином, в ході проведеної роботи було розроблено мобільний додаток під Android для управління особистими доходами і витратами, яке задовольняє цілі створення.

Було досліджено історичні дані, хронологію розвитку мобільних додатків, а також порівняння з конкурентами. Наведено статистичні дані щодо популярності розробки під мобільні телефони для підтвердження актуальності теми.

Висвітлено інформацію щодо використовуваних систем у роботі. Досліджено основні елементи створення мобільного додатку. Обрано та застосовано інструментарій для написання додатку, перелічено основні бібліотеки та технології, що використовувались.

2. Відбулося проектування проекту. Були змодульовані моделі: прецендентів (Рисунок 3.1), послідовності (Рисунок 3.2), діяльності (Рисунок 3.3), предметної галузі (Рисунок 3.4), артефактів (Рисунок 3.5) та пакетів (Рисунок 3.6). Також були створені макети меню для додатка які показані на рисунках 3.7 – 3.10.

3. На фінальному етапі розробки проводилося тестування створеної системи, як єдиного цілого. Даний процес не виявив особливих порушень, так як проміжне тестування під час розробки, безсумнівно, допомогло запобігти їх.

4. Додаток добре функціонує і виконує основну задачу: дозволяє вести свій бюджет і керувати ним. Додаток дозволяє додавати всі свої фінансові операції, редагувати і видаляти їх. Можна переглядати статистику за категоріями доходу і витратами з виведенням інформації, скільки по даній категорії витрачено. У додатку немає вікових обмежень.

5. За час роботи над дипломним проектом були вивчені:

- ОС Android (особливості і принципи)

- Платформа для кроссплатформенної розробки Xamarin;
- Реляційна база SQLite
- Принципи проектування додатків;
- Дослідження в розділі дизайну мобільних додатків;

6. Надалі планується покращувати і модернізувати даний додаток за рахунок додавання нового функціоналу:

- поліпшення інтерфейсу;
- створення особистого кабінету;
- курс валют онлайн;
- віджети;
- захист паролем;
- синхронізація на декількох пристроях;
- реалізувати зчитування смс-повідомлень від банків, тим самим забезпечити автоматизоване заповнення безготівкових витрат і доходів;
- перенести додаток на iOS;

Проте, розроблений додаток в тому вигляді, в якому він є зараз, вже може бути завантажено на майданчик Google Play і поширюватися у вільному доступі.

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%20of%20it

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/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/guide/topics/ui/look-and-feel

11. Quickstart: Create and publish a package using Visual Studio (.NET Framework, Windows) [Електронний ресурс] - https://docs.microsoft.com/nuget/quickstart/create-and-publish-a-package-using-visual-studio-net-framework

Додаток

ДИПЛОМНА РОБОТА

«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ УПРАВЛІННЯ ОСОБИСТИМИ ДОХОДАМИ ТА ВИТРАТАМИ МОВОЮ C#»

Виконав
Студент
Кривошапка Д.О.
Керівник
Гаманюк І.М.

Київ - 2021

Основні характеристики роботи

- **Метою** розробити якісно працюючий мобільний додаток на платформі Xamarin за ведення особистої бухгалтерії. Для вирішення цього завдання буде побудована модель ведення фінансів: короткий опис назви транзакції, що вноситься в систему, сума, дата внесення транзакцій, більше докладної інформації щодо транзакцій у системі обліку.
- **Об'єктом дослідження** роботи є розробка мобільного додатку під систему Android.

Актуальність задачі

У наш час майже неможливо уявити життя сучасної людини без мобільного телефону, у зв'язку з якою мобільна розробка стрімко розвивається, як у плані функціональності, так і в плані дизайну. Зараз смартфони для власного функціоналу практично підключені до комп'ютерів: на них виконують службові завдання, займаються пошуком в інтернеті, а також зловмисною інформацією, не кажучи вже про прості мультимедійні функції - огляд фільмів, фотографій, прослуховування музики і так далі.

Для чого потрібні мобільні додатки для контролю власних кошт

За даними статистики Федеральної резервної системи США майже половина дорослого населення в США не могли дозволити собі здійснювати незаплановані виплати в розмірі всього лише чотирьох сотень доларів. За даними того ж дослідження майже 25% дорослих з роботою покривають свої щомісячні витрати. Фінансова картина не набагато краща для тих, хто живе в інших частинах світу. Взяти управління своїми особистими фінансами під контроль - це дуже серйозне питання. Наслідки занадто великі борги можуть бути серйозними: зіпсований кредитний рейтинг, вилучення автомобілів або викуп вашого будинку і навіть особисте банкрутство. Ось для чого потрібен додаток для контролю бюджету. Добре складений бюджет може допомогти вам відстежувати, як ви витрачаєте ваші гроші, а також допомогти спланувати, як впорядкувати свої витрати.

Недоліки існуючих додатків

Незважаючи на те що на даний момент схожих додатків велика кількість, але у більшості з них присутні однакові проблеми:

- Інтерфейс перевантажений непотрібними речами з-за чого користування додатками ускладнюється
- Більшість потрібних функцій платні
- Відсутня українська локалізація

Операційна система Android

Android - це операційна система, спроектована для роботи на смартфонах, планшетах та інших пристроях. У понад 83% смартфонах, проданих в 2018 році, була встановлена операційна система Android. Одна з головних переваг Android - це відкрита платформа. Вона доступна всім, і будь-який виробник мобільних пристроїв або розробник може використовувати Android в своїх цілях: для створення додатків, пристроїв і навіть власної операційної системи. Це завжди буде її головною перевагою.

Xamarin

Для розробки додатку я використовував платформу Xamarin.

Xamarin - це платформа розробки мобільних додатків для створення нативних додатків iOS, Android і Windows із загального коду C # або .NET, яка дозволяє багаторазово використовувати між платформами від 75% до майже 100% коду.

Даний фреймворк включає в себе наступні частини:

- Xamarin.IOS - бібліотека класів для C # (для доступу до iOS SDK);
- Xamarin.Android - бібліотека класів для C # (для доступу до Android SDK);
- Компілятори для обох ОС;
- Середовища розробки - рідна IDE Xamarin Studio і плагін для Visual Studio.

Переносимість коду

Xamarin - це засіб крос-платформеної розробки, це означає, що додаток, написаний один раз, може бути запущено на різних мобільних платформах. Однак є певна особливість - при загальній логіці шар UI для кожної платформи потрібно писати окремо. Крос-платформеною є всі верстви, крім Application Layer і User Interface Layer.

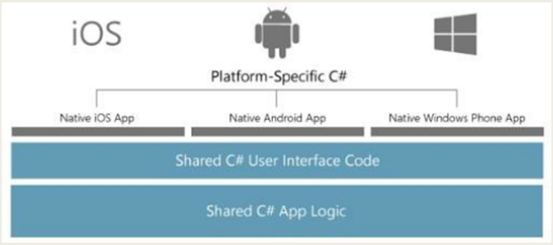


Рисунок 1 – Схема роботи кросплатформених додатків на платформі Xamarin

Activity

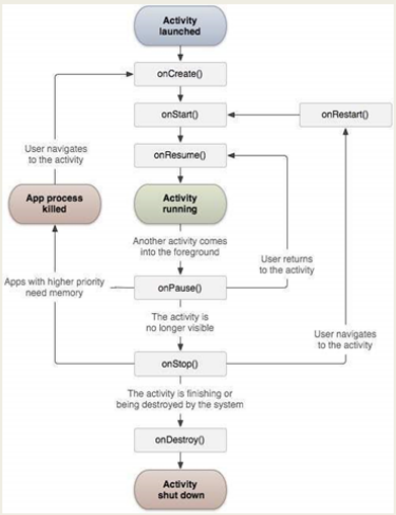


Рисунок 2 - Клас Activity та його життєвий цикл

Одним з ключових компонентів при розробці мобільного додатку в Xamarin, зокрема при створенні візуального інтерфейсу є activity (активність).

Android-додаток може мати від однієї до кількох активностей. Всі об'єкти activity є об'єктами класу android.app.Activity, який містить базову функціональність для всіх activity. Головну активність або MainActivity прийнято наслідувати від класу AppCompatActivity, який, в свою чергу, побічно успадковується від базового класу Activity

Патерн MVVM

Патерн MVVM (Model-View-ViewModel) дозволяє відокремити логіку додатки від візуальної частини (подання). Даний патерн є архітектурним, тобто він задає загальну архітектуру програми. MVVM складається з трьох компонентів: моделі (Model), моделі подання (ViewModel) і уявлення (View)

Рисунок 3 - Компоненти MVVM та їх зв'язок

Model

Модель описує використовувані в додатку дані. Моделі можуть містити логіку, безпосередньо пов'язану цими даними, наприклад, логіку валідації властивостей моделі. У той же час модель не повинна містити ніякої логіки, пов'язаної з відображенням даних і взаємодією з візуальними елементами управління. Нерідко модель реалізує інтерфейси INotifyPropertyChanged або INotifyCollectionChanged, які дозволяють повідомляти систему про зміни властивостей моделі. Завдяки цьому полегшується прив'язка до подання, хоча знову ж пряму взаємодію між моделлю і представленням відсутня.

View

View або подання визначає візуальний інтерфейс, через який користувач взаємодіє з додатком. Подання в Xamarin. Android - це код в xaml, який визначає інтерфейс у вигляді кнопок, текстових полів та інших візуальних елементів.

Однак іноді в файлі пов'язаного коду все може знаходитися певна логіка, яку важко реалізувати в рамках патерна MVVM у ViewModel.

ViewModel

ViewModel або модель уявлення пов'язує модель і уявлення через механізм прив'язки даних. Якщо в моделі змінюються значення властивостей, при реалізації моделлю інтерфейсу INotifyPropertyChanged автоматично йде зміна відображуваних даних в поданні, хоча безпосередньо модель і уявлення не пов'язані.

Оскільки елементи View, тобто візуальні компоненти типу кнопок, не використовують події, то View взаємодіє з ViewModel за допомогою команд. Наприклад, користувач хоче зберегти введені в текстове поле дані. Він натискає на кнопку і тим самим відправляє команду під ViewModel. А ViewModel вже отримує передані дані і відповідно до них оновлює модель.

Зберігання даних

База даних - це організована структура, призначена для зберігання, зміни та обробки взаємозалежної інформації.

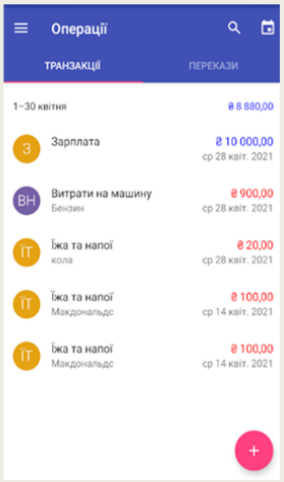
Для роботи була використана СУБД SQLite.

SQLite - це вбудована реляційна база даних, з відкритим кодом. Тобто вона не використовує звичну модель роботи бази даних клієнт-сервер і не є окремим працюючим процесом, як наприклад база даних MySQL.

При такому підході база даних SQLite (з усіма таблицями) являє собою звичайний текстовий файл, який можна розташувати в зручному для нас місці.

Базі даних SQLite не потрібно окремий сервер для роботи. Двигун SQLite вбудовується прямо в додаток і вимагає тільки доступ до файлів.

Material Design



Для того щоб програми були зручними і приємними для користувача, вони повинні слідувати принципам Material Design. За допомогою специфікації кольорів Material Design були зроблені стилізовані теми додатку, де кольори панелі програми, стану і контекстні панелі повинні поєднуватися з кольорами заголовків, текст на кнопках і елементами інтерфейсу (checkbox і текстові поля).

Також Material Design дає рекомендації щодо поліпшення розташування елементів екрану і поради за розміром тексту на складових екрану. У специфікації Material Design відображений найкращий досвід по дизайну додатка. Специфікації представляють собою керівництво по вибору стилю, зображень, анімації і т.д. Для читабельності тексту в розділі по типографії були вказані поради щодо вибору розміру тексту. У керівництві Material Design рекомендовано застосування інтервалів в 8dp. Вирівнювання компонентами екрану уздовж однієї лінії спрощує читання і роботу з додатком. Так, додаток буде кастомізоване за допомогою колірного дизайну, розстановки відступів і кордонів, а також грамотного розташування об'єктів на екрані.

Пакети Nuget

Nuget - це одне з розширень Visual Studio, яке дозволяє з легкістю встановлювати, оновлювати і видаляти бібліотеки (збірки), компоненти, інструменти.

Він бере на себе всі кроки цього процесу, в тому числі:

- пошук бібліотеки;
- завантаження необхідного для установки пакета;
- перевірка його хеш-значення на відповідність із заданим сервером (перевірка цілісності);
- розпакування файлів бібліотеки в потрібне місце;
- додавання посилань (reference) на її збірки в проект;
- модифікацію файлів конфігурації (web.config, app.config) при необхідності.

Висновки

- Таким чином, в ході проведеної роботи було розроблено мобільний додаток під Android для управління особистими доходами і витратами, яке задовольняє цілі створення.
- За час роботи над дипломним проектом були вивчені: ОС Android (особливості і принципи), Платформа для кроссплатформенної розробки Xamarin, Реляційна база SQLite, Принципи проектування додатків, Дослідження в розділі дизайну мобільних додатків
- Надалі планується покращувати і модернізувати даний додаток за рахунок додавання нового функціоналу: поліпшення інтерфейсу, створення особистого кабінету, курс валют онлайн, віджети, захист паролем, синхронізація на декількох пристроях, реалізувати зчитування смс-повідомлень від банків, тим самим забезпечити автоматизоване заповнення безготівкових витрат і доходів. перенести додаток на iOS.