МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Львівський національний університет імені Івана Франка
Факультет електроніки та комп'ютерних технологій
Кафедра радіоелектронних і комп'ютерних систем

Допустити до захисту
Завідувач кафедри
_____проф. Оленич Ігор Богданович
«___»_____2023 р.

Кваліфікаційна робота
Бакалавр
(освітній ступінь)

РОЗРОБЛЕННЯ МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВІДОБРАЖЕННЯ ДАНИХ РИНКУ КРИПТОВАЛЮТ З ДОПОМОГОЮ ФРЕЙМВОРКУ XAMARIN

Виконав:
студент IV курсу групи ФЕП-41
спеціальності:
121 Інженерія програмного забезпечення
_____Щербина Олег Юрійович
Науковий керівник:
_____асист. Павлик Михайло Романович
«___»_____2023 р.

Рецензент:

_____ _____
(підпис) (ПІБ)

Львів 2023

АНОТАЦІЯ

У цій дипломній роботі представлено комплексний аналіз та структуру дизайну для мобільного додатку, призначеного для відображення даних криптовалютного ринку в реальному часі. Оскільки криптовалюти набувають все більшого значення у фінансовому ландшафті, потреба в точній і доступній інформації стає вирішальною як для інвесторів, так і для ентузіастів. Використовуючи можливості мобільних пристроїв, це дослідження вивчає важливість розробки мобільного додатку, який спеціально призначений для представлення даних про ринок криптовалют.

Для розробки додатку було проаналізовано існуючу літературу про криптовалюти, аналіз ринку та мобільні додатки у сфері фінансових послуг. Аналізуючи існуючі рішення, їх дизайн, основний функціонал та методи розробки.Для перевірки ефективності запропонованої структури проектування проводиться етап проектування та впровадження. Це включає розробку архітектури системи, дизайн інтерфейсу користувача, розробку бекенду, інтеграцію API та комплексне тестування.

ABSTRACT

This thesis presents a comprehensive analysis and design framework for a mobile application designed to display real-time cryptocurrency market data. As cryptocurrencies become increasingly important in the financial landscape, the need for accurate and accessible information becomes crucial for investors and enthusiasts alike. Leveraging the power of mobile devices, this study explores the importance of developing a mobile application that is specifically designed to present data on the cryptocurrency market.

The existing literature on cryptocurrencies, market analysis, and mobile applications in the financial services industry was analysed to develop the app. Analysing existing solutions, their design, main functionality and development methods.To test the effectiveness of the proposed design framework, the design and implementation phase is carried out. This includes system architecture development, user interface design, backend development, API integration and comprehensive testing. ping a mobile application that is specifically designed to present data on the cryptocurrency market.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, СКОРОЧЕНЬ І ТЕРМІНІВ

UI	User Interface
API	Application Programming Interface
IDE	Integrated Development Environment
SQL	Structured query language
URL	Uniform Resource Locator
XAML	eXtensible Application Markup Language
MVVM	Model-View-ViewModel
JSON	JavaScript Object Notation

ВСТУП

В останні роки криптовалюти стали популярним об'єктом інвестування. Зі збільшенням кількості криптовалют зростає і попит на інформацію про них. Мобільні додатки можуть бути цінним інструментом для інвесторів і трейдерів, які хочуть бути в курсі останніх даних криптовалютного ринку.

У цій дипломній роботі я розробляю мобільний додаток для відображення даних криптовалютного ринку. Я почну з обговорення переваг використання мобільних додатків для цієї мети. Потім я розгляну різні типи доступних мобільних додатків, а також порівняю і зіставлю їх функції. Нарешті, я розгляну фактори, які слід враховувати при виборі мобільного додатку для відображення даних криптовалютного ринку.

Використання мобільних додатків для відображення даних криптовалютного ринку має ряд переваг. По-перше, мобільні додатки є портативними і доступні з будь-якого місця. Це особливо важливо для інвесторів і трейдерів, які знаходяться в дорозі. По-друге, мобільні додатки часто більш актуальні, ніж традиційні веб-сайти. Це пов'язано з тим, що їх можна частіше оновлювати. По-третє, мобільні додатки можуть надавати користувачам різноманітні функції, які недоступні на традиційних веб-сайтах. Ці функції можуть включати оновлення цін в режимі реального часу, діаграми і графіки, а також новини та аналітику.

Існує кілька різних типів мобільних додатків для відображення даних криптовалютного ринку. Деякі з найпопулярніших типів включають:

- Програми для відстеження цін: ці програми надають користувачам оновлення цін на криптовалюти в режимі реального часу.
- Графіки та діаграми: ці програми надають користувачам візуальне представлення даних криптовалютного ринку.
- Новини та аналітичні програми: ці програми надають користувачам новини та аналітику про ринок криптовалют.

При виборі мобільного додатку для відображення даних криптовалютного

ринку важливо враховувати наступні фактори:

- Спектр криптовалют, які підтримуються: Деякі додатки підтримують лише обмежену кількість криптовалют, в той час як інші підтримують більш широкий спектр.
- Частота оновлення даних: Деякі програми оновлюють свої дані частіше, ніж інші.
- Інтерфейс користувача: Користувацький інтерфейс повинен бути простим у використанні та навігації.
- Ціна: Вартість додатку має бути доступною.

Безпека - ще один важливий фактор, який слід враховувати при виборі мобільного додатку для відображення даних криптовалютного ринку - це безпека. Ринки криптовалют часто стають мішенню для хакерів, тому важливо вибрати додаток, який має сильні функції безпеки.

Точність даних, що відображаються в додатку, також важлива. Деякі додатки можуть бути не такими точними, як інші. Важливо перевірити точність даних перед використанням програми.

Користувацький досвід також важливий. Додаток має бути простим у використанні та навігації. Інтерфейс користувача повинен бути чітким і зрозумілим.

Дизайн додатку також важливий. Додаток повинен бути візуально привабливим і легким для сприйняття.

РОЗДІЛ 1. АНАЛІЗ РИНКУ КРИПТОВАЛЮТ ТА МОБІЛЬНИХ ДОДАТКІВ ДЛЯ ВІДСТЕЖЕННЯ ЦЬОГО РИНКУ

1.1. Аналіз ринку криптовалют

Аналіз ринку криптовалют - це дослідження та оцінка ринку криптовалют, яка може проводитися з використанням різноманітних методів та підходів, що дозволяють отримати більш детальну інформацію про ринок криптовалют та його стан.

Стосовно теми дипломної роботи "Мобільний додаток для відображення даних криптовалютного ринку" аналіз ринку криптовалют може бути корисним для розуміння потреб користувачів у подібних додатках, а також для забезпечення додатку актуальними та точними даними.

Існує ряд різних методів, які можна використовувати для аналізу ринку криптовалют. Деякі з найпоширеніших методів включають:

- Аналіз ринкової капіталізації: Цей метод використовується для оцінки загальної ринкової капіталізації ринку криптовалют. Цей метод дозволяє зрозуміти стан ринку і динаміку його змін. Крім того, він дозволяє порівняти різні криптовалюти і їх ринкову вартість.
- Аналіз цінових трендів: Цей метод використовується для визначення тенденцій зміни цін на криптовалюту. Аналіз цінових трендів використовує графіки цінових трендів і різні технічні індикатори, щоб зрозуміти моменти зміни цін.
- Аналіз новин та інформації: Цей метод використовується для аналізу впливу новин та інформації на ринок криптовалют. Для цього використовуються різні засоби, такі як соціальні мережі, новини тощо.

Ринок криптовалют - дуже динамічний і нестабільний ринок, інвестори і трейдери повинні бути постійно уважними і швидко реагувати на зміни. Тому дослідження цього ринку є актуальним і важливим завданням для будь-якого інвестора або трейдера. Для того, щоб проаналізувати ринок криптовалют,

необхідно вивчити кілька ключових аспектів, таких як динаміка цін, рейтинги криптовалют, технічний аналіз, новини та події, які можуть вплинути на ціни.

- Динаміка цін - один з найважливіших аспектів аналізу ринку криптовалют. Вона дозволяє зрозуміти поточний стан ринку і виявити тенденції, які можуть вплинути на ціни криптовалют в майбутньому.
- Рейтинги криптовалют - ще один важливий аспект аналізу криптовалютного ринку. Вони допомагають оцінити потенціал різних криптовалют і визначити ті, які мають найбільші шанси на успіх.
- Технічний аналіз - це метод аналізу цінових графіків та інших технічних показників для виявлення тенденцій і закономірностей, які можуть вплинути на ціни криптовалют.
- Новини та події також можуть мати значний вплив на ціни криптовалют. Важливо бути в курсі останніх новин і подій на ринку криптовалют, щоб приймати обґрунтовані інвестиційні рішення.

Детальний аналіз ринку криптовалют допоможе зрозуміти його стан і напрямки розвитку, а також приймати обґрунтовані інвестиційні рішення. Розробка мобільного додатку для відображення даних криптовалютного ринку є важливим кроком на шляху надання інформаційної підтримки інвесторам і трейдерам на ринку криптовалют.

Крім перерахованих вище методів, існує ряд інших факторів, які можна враховувати при аналізі ринку криптовалют. До таких факторів відносяться:

- Регулювання: Регулювання криптовалют в різних країнах може мати значний вплив на ціни криптовалют.
- Технології: Розвиток нових технологій, пов'язаних з криптовалютами, таких як технологія блокчейн, також може мати значний вплив на ціни криптовалют.

- Суспільні настрої: Суспільні настрої щодо криптовалют також можуть мати значний вплив на ціни криптовалют.

Враховуючи всі ці фактори, можемо отримати більш повне уявлення про ринок криптовалют і приймати більш обґрунтовані інвестиційні рішення.

1.2. Огляд існуючих мобільних додатків для відстеження ринку криптовалют

Огляд існуючих мобільних додатків для відстеження ринку криптовалют є важливою частиною дослідження перед розробкою власного мобільного додатку. Даний розділ містить огляд та аналіз популярних мобільних додатків, які використовуються для відстеження ринку криптовалют.

У цьому розділі буде розглянуто такі мобільні додатки:

1. Blockfolio
2. CoinMarketCap
3. Delta

Для кожного додатку будуть проаналізовані такі параметри:

- Функціональні можливості;
- Дизайн та інтерфейс;
- Наявність та якість графіків;
- Доступність та точність даних;
- Цінова політика.

1. Blockfolio.

Blockfolio - це популярний додаток для відстеження криптовалют, який дозволяє користувачам відстежувати понад 10 000 різних криптовалют і токенів. Він також дозволяє користувачам додавати транзакції та створювати портфелі. Blockfolio також пропонує ряд інших функцій, таких як можливість отримувати сповіщення про зміни цін та інші події.

Плюси:

- Простота у використанні
- Хороший користувацький інтерфейс
- Широкий спектр функцій
- Безкоштовне використання

Мінуси:

- Графіки може бути важко читати
- Немає інструментів технічного аналізу
- Немає темного режиму

Отже Blockfolio хороший варіант для користувачів, які шукають простий і зрозумілий додаток для відстеження криптовалют. Він пропонує широкий спектр функцій і є безкоштовним у використанні. Однак графіки може бути важко читати, а інструменти технічного аналізу відсутні.

2. CoinMarketCap.

CoinMarketCap - популярний додаток для відстеження криптовалют, який надає користувачам доступ до інформації про понад 8 000 криптовалют і токенів. Він містить функціонал, який дозволяє користувачам шукати монети за назвою, символом або ринковою капіталізацією. Також можна відстежувати ціни і зміни ринкової капіталізації, переглядати інформацію про біржі, які торгують криптовалютами, і багато іншого.

Плюси:

- Широкий спектр функцій

- Точні дані
- Безкоштовне використання

Мінуси:

- Інтерфейс може бути захаращений
- Немає темного режиму

Отже CoinMarketCap хороший варіант для користувачів, які шукають комплексний додаток для відстеження криптовалют. Він пропонує широкий спектр функцій і точні дані. Однак інтерфейс може бути захаращеним, а темний режим відсутній.

3. Delta

Delta - популярний додаток для відстеження криптовалют, який дозволяє користувачам відстежувати понад 6 000 криптовалют і токенів, додавати транзакції та створювати портфелі. Він також надає можливість відстежувати зміни цін, відсоткові зміни та графіки в реальному часі. Крім того, користувачі можуть налаштувати оповіщення, щоб отримувати сповіщення про зміни цін або інші події.

Плюси:

- Привабливий користувацький інтерфейс
- Темний режим
- Інструменти технічного аналізу
- Безкоштовна та платна версії

Мінуси:

- Обмежений функціонал у безкоштовній версії

- Немає інструментів для ребалансування портфеля

Отже Delta хороший варіант для користувачів, які шукають додаток для відстеження криптовалют з сучасним дизайном і інструментами технічного аналізу. Безкоштовна версія обмежена в можливостях, але платна версія пропонує більш широкий спектр функціональних можливостей.

Висновок: всі три додатки пропонують різноманітні функції та можливості, тому вибір найкращого додатку для вас буде залежати від ваших індивідуальних потреб та вподобань. Якщо ви шукаєте простий і зрозумілий додаток з хорошим користувацьким досвідом, то Blockfolio - хороший варіант. Якщо ви шукаєте додаток з більш широким спектром функцій і можливостей, то CoinMarketCap або Delta є хорошими варіантами.

1.3. Опис функціоналу та вимог до мобільного додатку

Виходячи з результатів аналізу, нам потрібно забезпечити наступне для нашого додатку:

- Функціональність:
    a. Наявність детальної інформації про криптовалюти, такої як історія курсів, відкриті та закриті ордери, капіталізація та інші показники;
    b. Можливість відстежувати курси криптовалют у різних валютних парах;
    c. Наявність графіків, які дозволяють аналізувати рух курсів криптовалют за певний період часу.
- Дизайн та інтерфейс:
    a. Зрозумілий та зручний інтерфейс, що дозволяє користувачам швидко та легко знаходити необхідну інформацію;

b. Різноманітні налаштування, які дозволяють користувачам налаштувати додаток на свій смак.

- Наявність та якість графіків:

a. Наявність різних типів графіків, які дозволяють аналізувати рух курсів криптовалют за різні періоди часу;

b. Висока якість графіків, що дозволяє користувачам досить точно аналізувати рух курсу.

При розробці Android-додатку для відображення інформації про криптовалюти важливо враховувати наступні тенденції:

- Material Design: Material Design - це мова дизайну, розроблена компанією Google, яка надає вказівки та принципи для створення користувацьких інтерфейсів, які є одночасно візуально привабливими та функціональними. Xamarin підтримує Material Design, що дозволяє легко створювати додатки, які виглядають і відчуваються як рідні на пристроях Android.

- Motion: Рух стає все більш важливим у дизайні інтерфейсу мобільних додатків. Використовуючи рух, ви можете створювати більш цікаві та інтерактивні додатки. Xamarin підтримує рух, що дозволяє легко додавати анімацію та переходи у ваші програми.

- Dark Mode: Темний режим стає все більш популярним на мобільних пристроях. Використовуючи темний режим, ви можете створювати додатки, які є легшими для очей і заощаджують заряд акумулятора. Xamarin підтримує темний режим, що дозволяє легко створювати додатки, які можна використовувати як у світлому, так і в темному режимі.

- Microinteractions: Мікровзаємодії - це невеликі, ледь помітні взаємодії, які можуть додати вашому додатку індивідуальності та завершеності. Xamarin

підтримує мікровзаємодії, що дозволяє легко додавати ці дрібні деталі до ваших додатків.

Це лише деякі з сучасних тенденцій у дизайні інтерфейсу мобільних додатків, які використовуються в Xamarin і в усіх додатках для Android. Дотримуючись цих тенденцій, можна створювати візуально привабливі, функціональні та цікаві додатки. Ось кілька конкретних прикладів того, як ці тенденції можна застосувати до додатку для відображення криптовалют:

- Матеріальний дизайн: Додаток може використовувати елементи матеріального дизайну, такі як картки, плаваючі кнопки дій і закусочні, щоб створити візуально привабливий і функціональний користувальницький інтерфейс.

- Fluent Design: Додаток може використовувати елементи Fluent Design, такі як акрил, слюда і тіні, щоб створити візуально привабливий і цікавий користувацький інтерфейс.

- Рух: Додаток може використовувати рух для створення більш привабливого та інтерактивного користувацького досвіду. Наприклад, додаток може використовувати анімацію, щоб показати користувачеві, як користуватися додатком, або забезпечити зворотній зв'язок, коли користувач виконує якусь дію.

- Темний режим: Додаток може підтримувати темний режим, щоб полегшити роботу з ним для очей і заощадити заряд акумулятора.

- Мікровзаємодія: Додаток може використовувати мікровзаємодії, щоб додати користувацькому інтерфейсу індивідуальності та відшліфувати його. Наприклад, додаток може використовувати мікровзаємодію, щоб показати користувачеві, як закрити модальний діалог.

Ось чому в додатку потрібно буде зробити наступне в інтерфейсі:

- Оновлення цін у реальному часі: Додаток повинен відображати оновлення цін в реальному часі для останніх цін на криптовалюту.

- Графіки: Додаток повинен відображати графіки, які показують історичні зміни цін на криптовалюти.

- Списки: Додаток повинен мати візуально приємні списки, які показують всю доступну інформацію про криптовалюту в невеликому, зручному форматі.

- Сортування: Додаток повинен відображати елементи з можливістю змінювати сортування (ціни, ринки, назва, капіталізація) за криптовалютами.

- Темний режим: Додаток повинен підтримувати темний режим.

На додаток до вищезазначеного, додаток також повинен мати наступні функції:

- Відстеження портфеля: Додаток повинен дозволяти користувачам відстежувати свої криптовалютні портфелі. Це включає можливість додавати і видаляти монети, а також переглядати поточну вартість портфеля.

- Сповіщення: Додаток повинен дозволяти користувачам отримувати сповіщення про зміни цін, новини та інші події, пов'язані з криптовалютами.

- Інтеграція з соціальними мережами: Додаток повинен дозволяти користувачам ділитися показниками свого портфеля та іншою інформацією з друзями та підписниками в соціальних мережах.

Завдяки цим функціям додаток стане цінним інструментом для криптовалютних інвесторів і трейдерів. Він надасть їм інформацію, необхідну для прийняття обґрунтованих інвестиційних рішень, а також інструменти для ефективного управління своїми портфелями.

1.4. Аналіз методів розробки додатків для Android

Аналіз методів розробки додатків для Android на мовах C#, Java та Python:

1. C# - це скомпільована мова, яка використовується для розробки широкого спектру додатків, включаючи мобільні, веб-додатки та десктопні програми. Це потужна та універсальна мова, яка підтримується великою спільнотою розробників. C# є гарним вибором для розробки додатків для Android, оскільки вона має низку переваг над іншими мовами, зокрема:
   - Швидкість: C# є скомпільованою мовою, що означає, що вона перетворюється в машинний код перед виконанням. Це робить програми на C# швидшими, ніж інтерпретовані мови, такі як Java.
   - Простота використання: C# є відносно легкою мовою для вивчення, і існує багато ресурсів, які допоможуть розробникам розпочати роботу.
   - Підтримка: C# добре підтримується великою спільнотою розробників. Це означає, що існує багато бібліотек та інструментів, які допоможуть розробникам створювати свої додатки.
2. Java - ще одна популярна мова для розробки додатків для Android. Java також є скомпільованою мовою і підтримується великою спільнотою розробників. Java є хорошим вибором для розробки додатків для Android, оскільки вона має ряд переваг над іншими мовами, зокрема:
   - Переносимість: Java є платформонезалежною мовою, що означає, що програми на Java можна запускати на будь-якій платформі, яка підтримує Java.
   - Безпека: Java - це безпечна мова, яка призначена для запобігання виконанню шкідливого коду.
   - Об'єктно-орієнтована: Java є об'єктно-орієнтованою мовою, що дозволяє легко розробляти складні додатки.
3. Python - це інтерпретована мова, яка використовується для розробки широкого спектру додатків, включаючи мобільні, веб- та десктопні додатки.

Python - це потужна та універсальна мова, яка підтримується великою спільнотою розробників. Python є хорошим вибором для розробки додатків для Android, оскільки вона має низку переваг над іншими мовами, зокрема:

- Швидкість: Python - це інтерпретована мова, що означає, що вона не така швидка, як скомпільовані мови, такі як C# або Java. Однак Python можна пришвидшити, використовуючи техніку компіляції Just-In-Time (JIT).
- Простота використання: Python є відносно легкою мовою для вивчення, і існує багато ресурсів, які допоможуть розробникам розпочати роботу.
- Підтримка: Python добре підтримується великою спільнотою розробників. Це означає, що існує багато бібліотек та інструментів, які допоможуть розробникам створювати свої програми.

Інструменти:

Існує ряд інструментів, які можна використовувати для розробки додатків для Android на C#, Java та Python. Деякі з найпопулярніших інструментів включають:

- Visual Studio(Xamarin або Unity): Visual Studio - це інтегроване середовище розробки (IDE), яке використовується для розробки програмних додатків. Visual Studio підтримує C#, Java та Python і має ряд функцій, які роблять його гарним вибором для розробки додатків для Android.
- Android Studio: Android Studio - це IDE, яка спеціально розроблена для розробки додатків для Android. Android Studio підтримує Java та Kotlin, а також має ряд функцій, які роблять його гарним вибором для розробки додатків для Android.
- Eclipse: Eclipse - це IDE, яка використовується для розробки програмних додатків. Eclipse підтримує C#, Java та Python, а також має ряд функцій, які роблять його гарним вибором для розробки додатків для Android.

C# та Xamarin є гарним вибором для розробки додатків для Android, оскільки вони мають ряд переваг над іншими мовами та платформами, зокрема:

- Швидкість: Додатки на C# та Xamarin зазвичай працюють швидше, ніж додатки на Java.
- Простота використання: C# та Xamarin відносно прості у вивченні мови, і існує багато ресурсів, які допоможуть розробникам розпочати роботу.
- Підтримка: C# та Xamarin добре підтримуються і мають велику спільноту розробників. Це означає, що існує багато бібліотек та інструментів, які допоможуть розробникам створювати свої додатки.

Крім того, C# та Xamarin пропонують ряд функцій, які роблять їх добре придатними для розробки додатків для Android, в тому числі:

- Xamarin Forms: Xamarin Forms - це крос-платформний фреймворк, який дозволяє розробникам створювати нативні мобільні додатки, які виглядають і відчуваються як нативні додатки.
- Xamarin.Android: Xamarin.Android - це набір інструментів, що дозволяє розробникам створювати нативні додатки для Android за допомогою C#.
- Xamarin.iOS: Xamarin.iOS - це набір інструментів, що дозволяє розробникам створювати нативні програми для iOS за допомогою C#.

Загалом, C# та Xamarin є гарним вибором для розробки додатків для Android, оскільки вони мають ряд переваг над іншими мовами та платформами.

РОЗДІЛ 2. ПЛАТФОРМИ РОЗРОБКИ .NET ТА XAMARIN

2.1. Засоби розробки

C# - це потужна та універсальна мова програмування, яка використовується для розробки широкого спектру додатків, включаючи мобільні, веб-додатки та десктопні програми. Це компільована мова, що означає, що перед виконанням вона перетворюється на машинний код. Це робить програми на C# швидшими та ефективнішими, ніж інтерпретовані мови, такі як JavaScript.

Xamarin - це кросплатформенна платформа розробки, яка дозволяє розробляти мобільні додатки для Android та iOS за допомогою C#. Це означає, що ви можете написати єдину кодову базу, яку можна використовувати для створення додатків для обох платформ. Це може заощадити вам багато часу і зусиль, оскільки вам не потрібно вивчати дві різні мови програмування або середовища розробки.

Visual Studio - це інтегроване середовище розробки (IDE), яке використовується для розробки програмних додатків. Це потужний інструмент, який надає широкий спектр можливостей для написання, налагодження та тестування коду. Visual Studio доступна для Windows та macOS і підтримує широкий спектр мов програмування, включаючи C#, Visual Basic та F#.

Git - це система контролю версій, яка використовується для відстеження змін у коді. Git є популярним вибором для розробки додатків для Android, оскільки він дозволяє розробникам спільно працювати над кодом і відстежувати зміни в часі.

JSON - це легкий формат обміну даними, який легко читати і записувати. JSON - популярний вибір для розробки додатків для Android, оскільки це простий і ефективний спосіб зберігання та передачі даних.

SQL - це мова баз даних, яка використовується для створення, читання, оновлення та видалення даних з баз даних. SQL є популярним вибором для розробки додатків для Android, оскільки це потужний та ефективний спосіб

управління даними.

MVVM - це патерн проектування програмного забезпечення, який використовується для відокремлення інтерфейсу користувача від основних даних та логіки. MVVM є популярним вибором для розробки додатків для Android, оскільки він полегшує розробку підтримуваних і тестованих додатків.

XAML - це мова розмітки, яка використовується для визначення користувацького інтерфейсу програми для Android. Це потужна та гнучка мова, яка дозволяє розробникам створювати складні та візуально привабливі користувацькі інтерфейси.

2.2. Мова програмування C#

C# - це об'єктно-орієнтована мова програмування загального призначення, розроблена компанією Microsoft. Це компільована мова, що означає, що вона перетворюється в машинний код перед виконанням. Це робить програми на C# швидшими та ефективнішими, ніж інтерпретовані мови, такі як JavaScript. C# також є сильно типізованою мовою, що означає, що типи змінних і виразів повинні бути оголошені явно. Це допомагає запобігти помилкам і робить код більш читабельним і зручним для супроводу. Деякі з ключових особливостей C#:

- Object-oriented: C# є об'єктно-орієнтованою мовою, що означає, що вона використовує об'єкти для представлення даних та поведінки.
- Compiled: C# є скомпільованою мовою, що означає, що вона перетворюється в машинний код перед виконанням. Це робить програми на C# швидшими та ефективнішими, ніж інтерпретовані мови.
- Strong typing: C# є мовою з сильною типізацією, що означає, що типи змінних та виразів повинні бути оголошені явно. Це допомагає запобігти помилкам і робить код більш читабельним та зручним для супроводу.
- Garbage collection: C# використовує збір сміття для автоматичного керування пам'яттю. Це звільняє розробників від необхідності турбуватися про управління пам'яттю, що може бути складним і схильним до помилок

завданням.

- Interoperability: C# розроблено для взаємодії з іншими мовами, такими як Visual Basic та Java. Це дозволяє легко використовувати C# для розробки додатків, які взаємодіють з іншими програмами та сервісами.

C# - це потужна та універсальна мова, яку можна використовувати для розробки широкого спектру додатків, включаючи:

- мобільні додатки
- веб-додатки
- десктопні програми
- Ігри
- Сервіси

Ось деякі з популярних фреймворків та бібліотек, які використовуються з C#:

- .NET Framework: .NET Framework - це комплексний набір бібліотек та інструментів, які можна використовувати для розробки різноманітних додатків.
- Xamarin: Xamarin - це кросплатформенна платформа розробки, яка дозволяє розробляти мобільні додатки для Android та iOS за допомогою C#.
- Unity: Unity - це ігровий рушій, який можна використовувати для розробки ігор для різних платформ, включаючи ПК, консолі та мобільні пристрої.
- ASP.NET: ASP.NET - це фреймворк для веб-розробки, який можна використовувати для розробки веб-додатків за допомогою C#.
- Entity Framework: Entity Framework - це фреймворк доступу до даних, який можна використовувати для доступу та управління даними в різних джерелах даних, включаючи бази даних, веб-сервіси та файлові системи.

Це лише деякі з багатьох фреймворків і бібліотек, доступних для C#.

2.3. Платформа .NET Framework

.NET Framework - це програмний фреймворк, розроблений компанією

Microsoft, який працює переважно на Microsoft Windows. Вона включає велику бібліотеку і забезпечує мовну сумісність (CLI) між кількома мовами програмування. Програми, написані для .NET Framework, виконуються в програмному середовищі, яке називається Common Language Runtime (CLR), віртуальна машина додатків, яка надає такі послуги, як безпека, управління пам'яттю та обробка винятків. Фреймворк складається з двох основних частин: середовища виконання спільної мови (CLR) і бібліотеки класів .NET Framework. CLR - це виконавчий механізм, який надає такі послуги, як безпека, управління пам'яттю та обробка винятків. Бібліотека класів .NET Framework - це повний набір багаторазових типів, які програмісти можуть використовувати для розробки додатків.

.NET Framework - це потужна та універсальна платформа, яку можна використовувати для розробки широкого спектру додатків. Це популярний вибір для розробки веб-додатків, десктопних і мобільних додатків. Ось деякі з ключових особливостей .NET Framework:

- Переносимість: .NET Framework є крос-платформним фреймворком, що означає, що його можна використовувати для розробки додатків для різних платформ, включаючи Windows, macOS і Linux.
- Можливість багаторазового використання: Бібліотека класів .NET Framework - це повний набір багаторазових типів, які програмісти можуть використовувати для розробки додатків.
- Простота використання: .NET Framework розроблений для простоти використання, з простою та інтуїтивно зрозумілою об'єктно-орієнтованою моделлю програмування.
- Безпека: .NET Framework включає в себе ряд функцій, які допомагають захистити додатки від загроз безпеки.
- Продуктивність: .NET Framework розроблений для того, щоб бути ефективним і високопродуктивним.

2.4. Платформа Xamarin

Xamarin - це кросплатформенна платформа для розробки мобільних додатків, яка дозволяє розробникам створювати нативні мобільні додатки для Android, iOS та macOS за допомогою C#. Xamarin складається з набору інструментів і бібліотек, які дозволяють розробникам писати код, що може бути скомпільований і запущений на різних платформах. Це дає можливість створити єдину кодову базу, яку можна використовувати для створення додатків для всіх трьох платформ. Xamarin тісно інтегрується з Visual Studio, що дозволяє легко розробляти додатки на Xamarin за допомогою знайомого IDE Visual Studio. Visual Studio надає ряд функцій, які спрощують розробку Xamarin додатків, зокрема:

- IntelliSense: Visual Studio надає IntelliSense, який допомагає розробникам писати код швидше і точніше.
- Редактор коду: Visual Studio надає потужний редактор коду, який полегшує написання та налагодження коду.
- Налагоджувач: Visual Studio надає потужний відладчик, який допомагає розробникам знаходити та виправляти помилки в коді.
- Модульне тестування: Visual Studio надає фреймворк для модульного тестування, який допомагає розробникам писати і запускати модульні тести для свого коду.
- Розгортання: Visual Studio містить майстер розгортання, який полегшує розгортання додатків Xamarin в App Store та Google Play.

На додаток до інтеграції з Visual Studio, Xamarin також пропонує ряд інших можливостей, які роблять його популярним вибором для розробки мобільних додатків:

- Нативна продуктивність: Додатки Xamarin компілюються у нативний код для кожної платформи, а це означає, що вони можуть досягти такого ж рівня продуктивності, як і додатки, створені за допомогою нативних інструментів розробки.

- Спільна кодова база: Програми Xamarin використовують єдину кодову базу, що дозволяє заощадити час і гроші під час розробки.
- Крос-платформна розробка: Xamarin дозволяє створювати додатки для різних платформ, використовуючи єдину кодову базу.
- Інтеграція з Visual Studio: Xamarin тісно інтегрується з Visual Studio, що дозволяє легко розробляти додатки Xamarin за допомогою знайомого IDE Visual Studio.
- Підтримка спільноти: Xamarin має велику і активну спільноту розробників, яка надає підтримку і ресурси для розробників Xamarin додатків.

Xamarin також дозволяє розробникам запускати додатки для Android прямо зі студії. Це робиться за допомогою програвача Xamarin Android Player, який є вбудованим емулятором, що дозволяє розробникам тестувати свої програми на пристроях Android без необхідності підключати реальний пристрій до комп'ютера. Xamarin Android Player - це потужний інструмент, який може заощадити розробникам багато часу та зусиль під час процесу розробки.

2.5. Середовище розробки Visual Studio

Visual Studio - це інтегроване середовище розробки (IDE) від Microsoft. Воно використовується для розробки комп'ютерних програм, зокрема веб-сайтів, веб-додатків, веб-сервісів і мобільних додатків. Це потужний інструмент, який надає ряд функцій, що допомагають розробникам писати код, налагоджувати код, а також збирати і розгортати додатки. Деякі з ключових можливостей Visual Studio включають:

- Редактор коду з IntelliSense, завершенням коду та підсвічуванням синтаксису
- Налагоджувач з точками зупинки, годинником та крокуванням
- Провідник проектів та рішень для керування проектами та файлами
- Система збірки для компіляції та компонування коду
- Система розгортання для публікації додатків
- фреймворк тестування для написання та запуску модульних тестів

Visual Studio доступна для Windows і macOS. Це комерційний продукт, але існує безкоштовна версія для студентів, розробників з відкритим вихідним кодом та індивідуальних розробників.

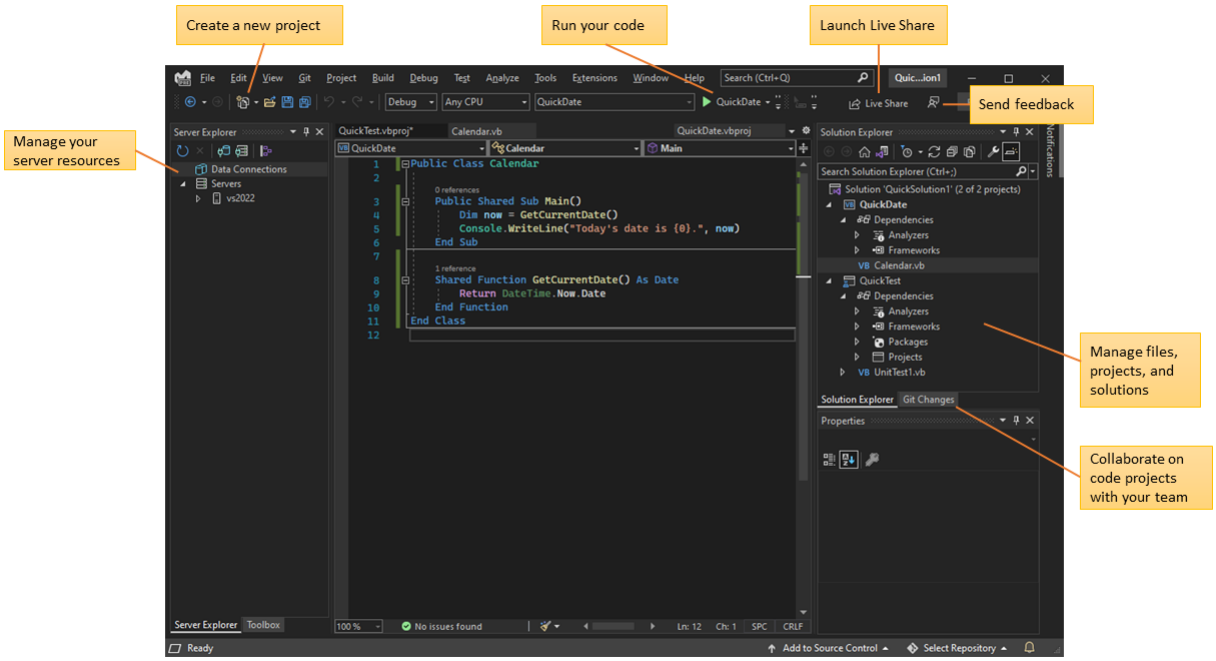

Рис. 2.1. Інтерфейс Visual Studio

Ось деякі з переваг використання Visual Studio:

- Потужність: Visual Studio - це потужний інструмент, який надає ряд можливостей, що допомагають розробникам писати код, налагоджувати код, а також збирати і розгортати програми.
- Ефективність: Visual Studio може допомогти розробникам працювати ефективніше завдяки таким функціям, як IntelliSense, завершення коду та підсвічування синтаксису.
- Спільна робота: Visual Studio може допомогти розробникам ефективніше співпрацювати, надаючи такі можливості, як інтеграція контролю вихідного коду та інтеграція з сервером командної основи.
- Масштабованість: Visual Studio можна масштабувати для задоволення потреб великих команд розробників.

2.6. Система контролю версій Git

Git - це розподілена система контролю версій. Вона використовується для відстеження змін у будь-якому наборі комп'ютерних файлів, зазвичай використовується для координації роботи програмістів, які спільно розробляють вихідний код під час розробки програмного забезпечення. Його цілі включають швидкість, цілісність даних і підтримку розподілених, нелінійних робочих процесів. Git був створений Лінусом Торвальдсом у 2005 році для розробки ядра Linux, а інші розробники ядра зробили свій внесок у його початкову розробку. З 2005 року Junio C Hamano є супровідником ядра. Git є вільним програмним забезпеченням з відкритим вихідним кодом на умовах Стандартної публічної ліцензії GNU. Воно доступне для використання на всіх основних операційних системах, включаючи Linux, macOS, Windows і Solaris.

Ось деякі з переваг використання Git'у:

- Контроль версій: Git дозволяє відстежувати зміни у коді з плином часу. Це може бути корисно для налагодження, співпраці з іншими та повернення до попередніх версій коду.
- Спільна робота: Git дозволяє легко співпрацювати з іншими над кодом. Ви можете ділитися своїм кодом з іншими, а вони можуть вносити до нього зміни, не впливаючи на вашу копію.
- Портативність: Git доступний для всіх основних операційних систем. Це означає, що ви можете використовувати Git на будь-якому комп'ютері, незалежно від того, яка операційна система на ньому встановлена.
- Вільний та відкритий код: Git - це безкоштовне програмне забезпечення з відкритим вихідним кодом. Це означає, що ви можете користуватися ним безкоштовно і можете змінювати його відповідно до своїх потреб.

2.7. Засіб передачі даних JSON

JSON (JavaScript Object Notation) - це легкий формат обміну даними, який широко використовується для передачі та зберігання даних. Це текстовий формат,

який легко читати і записувати як людям, так і машинам. JSON походить від мови програмування JavaScript, але він не залежить від мови і може використовуватися з різними мовами програмування. JSON представляє дані у структурованому вигляді за допомогою пар ключ-значення. Дані організовані в об'єкти, які взяті у фігурні дужки {}. Кожен об'єкт складається з однієї або декількох пар ключ-значення, де ключ - це рядок, взятий у подвійні лапки "", за яким слідує двокрапка :, а значення може бути різних типів даних, таких як рядок, число, логічний, нуль, масив або інший вкладений об'єкт (Рис.2.2.).

Рис. 2.2. Приклад простого JSON-об'єкта.

JSON широко використовується у веб-розробці та API (інтерфейсах прикладного програмування) як формат даних для обміну інформацією між різними системами. Він підтримується більшістю мов програмування за допомогою бібліотек або вбудованих функцій, які надають можливості синтаксичного аналізу та серіалізації, що дозволяє розробникам конвертувати дані JSON у нативні структури даних у відповідних мовах програмування і навпаки.

Простота, читабельність і сумісність JSON роблять його популярним вибором для передачі та зберігання даних у широкому спектрі додатків, включаючи веб-сервіси, мобільні додатки та пристрої IoT (Інтернет речей).

2.8. Мова баз даних SQL

SQL розшифровується як Structured Query Language (мова структурованих запитів). Це стандартна мова для доступу до даних і маніпулювання ними в реляційних системах управління базами даних (СКБД). SQL використовується для виконання найрізноманітніших операцій над даними, в тому числі:

- Створення та видалення таблиць

- Вставлення, оновлення та видалення даних
- Вибір даних з таблиць
- Об'єднання таблиць між собою
- Сортування та фільтрація даних
- Створення та керування поданнями
- Надання та відкликання дозволів

SQL - це декларативна мова, що означає, що ви вказуєте базі даних, що вам потрібно, а не як це отримати. Це робить мову SQL дуже потужною і легкою у вивченні. Існує багато різних СУБД, які підтримують SQL, зокрема MySQL, PostgreSQL, Oracle та Microsoft SQL Server. Кожна СУБД має свої унікальні функції та можливості, але всі вони підтримують основну мову SQL.

SQL - дуже популярна мова для розробки баз даних. Її використовують найрізноманітніші організації, включаючи підприємства, державні установи та навчальні заклади. SQL також є популярною мовою для аналізу даних і створення звітів.

2.9. Патерн проектування MVVM

Model-View-ViewModel (MVVM) - це патерн проектування програмного забезпечення, який відокремлює інтерфейс користувача (UI) від базових даних і бізнес-логіки. Таке розділення проблем полегшує розробку, тестування та підтримку додатків. Патерн MVVM складається з трьох частин:

- Model: Модель представляє дані та бізнес-логіку додатку.
- View: Подання - це інтерфейс програми. Воно відображає дані і дозволяє користувачам взаємодіяти з додатком.
- ViewModel: Модель представлення є мостом між моделлю і представленням. Вона переводить дані з моделі у формат, зрозумілий для представлення, і переводить користувацьке введення з представлення в команди, зрозумілі для моделі.

Патерн MVVM має кілька переваг:

- Розділення проблем: Розділення завдань полегшує розробку, тестування та підтримку додатків.
- Можливість тестування: Патерн MVVM полегшує тестування інтерфейсу користувача та бізнес-логіки окремо.
- Повторне використання: Модель представлення може бути повторно використана в декількох представленнях.
- Гнучкість: Паттерн MVVM можна адаптувати до різних фреймворків інтерфейсу користувача.

Паттерн MVVM є популярним вибором для розробки додатків, які використовують графічний інтерфейс користувача (GUI). Це добре зарекомендував себе патерн, який був використаний у багатьох успішних додатках.

2.10. Мова розмітки XAML

XAML розшифровується як Extensible Application Markup Language. Це декларативна мова на основі XML, розроблена Microsoft для ініціалізації структурованих значень та об'єктів. Вона доступна за програмою Microsoft Open Specification Promise. XAML широко використовується в Windows Presentation Foundation (WPF), Silverlight, Workflow Foundation (WF), Windows UI Library (WinUI) та Universal Windows Platform (UWP). У WPF і UWP XAML - це мова розмітки інтерфейсу користувача для визначення елементів інтерфейсу, прив'язки даних і подій. У WF, однак, XAML визначає робочі процеси. XAML - це потужна та універсальна мова, яку можна використовувати для створення широкого спектру користувацьких інтерфейсів. Її легко вивчити і використовувати, і вона підтримується широким спектром інструментів розробки.

Ось кілька прикладів коду XAML:

Цей код створює кнопку з назвою "Button1" і текстом "Click Me". При натисканні на кнопку буде викликано обробник події з іменем "Button1_Click"(Рис.2.3.).

```
<Button Name="Button1" Click="Button1_Click">Click Me</Button>
```

Рис. 2.3. Приклад створення кнопки.

Цей код створює текстове поле з іменем TextBox1 і початковим текстом "Hello, World!"(Рис.2.4.).

```
<TextBox Name="TextBox1" Text="Hello, World!"></TextBox>
```

Рис. 2.4. Приклад створення текстового поля.

Цей код створює сітку з панеллю стека всередині. Панель стека містить текстовий блок і кнопку(Рис.2.5.).

```
<Grid>
  <StackPanel Orientation="Vertical">
    <TextBlock Text="This is a text block."></TextBlock>
    <Button Name="Button1" Click="Button1_Click">Click Me</Button>
  </StackPanel>
</Grid>
```

Рис. 2.5. Приклад створення сітки з панеллю стека всередині.

РОЗДІЛ 3. РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ ДЛЯ ВІДОБРАЖЕННЯ ДАНИХ РИНКУ КРИПТОВАЛЮТ

3.1. Опис архітектури мобільного додатку

Під час розробки Xamarin-додатків важливо мати чітке розуміння дизайну та функціональності програми. Це дозволяє сфокусуватися на розробці, гарантуючи, що необхідні функції будуть реалізовані без зайвих складнощів.

У Xamarin структура додатку обертається навколо різних сторінок, кожна з яких слугує певній меті. Обговоримо сторінки, необхідні для додатку:

1. Домашня сторінка – слугує початковим екраном при запуску програми. Вона забезпечує привітний інтерфейс для користувачів і служить центральним вузлом для навігації до інших розділів програми. На цій сторінці ви можете відображати відповідну інформацію про криптовалюти, таку як найкращі показники, оновлення новин або ринкові тенденції.
2. Сторінка налаштувань – дозволяє користувачам налаштовувати параметри програми. Вона надає опції для налаштування параметрів, пов'язаних з уподобаннями користувача, сповіщеннями, зовнішнім виглядом програми або будь-якими іншими параметрами, що налаштовуються. Користувачі можуть персоналізувати додаток відповідно до своїх конкретних вимог.
3. Сторінка валют – призначена для відображення інформації про різні криптовалюти. Вона може демонструвати список популярних криптовалют, їхні поточні ціни, ринкову капіталізацію, обсяг та інші відповідні дані. Користувачі можуть вивчити детальну інформацію про конкретні валюти, наприклад, історичні графіки цін, ринкові тенденції та додаткову статистику.
4. Сторінка обміну – надає користувачам можливість торгувати або обмінювати криптовалюти. Вона може мати зручний інтерфейс для здійснення торгів, відображення книг замовлень, цін в реальному часі та історії транзакцій. Користувачі можуть розміщувати замовлення на купівлю/продаж, відстежувати свій портфель і керувати своїми інвестиціями безпосередньо в додатку.

5. Сторінка ринків – фокусується на наданні користувачам огляду криптовалютних ринків. Вона може відображати ринкові індекси, коливання цін і показники різних криптовалют на різних біржах. Користувачі можуть відстежувати ринкові тенденції, аналізувати рух цін і приймати обґрунтовані рішення на основі наявних даних.
6. Сторінка графіків – представляє графічне представлення криптовалютних даних, що дозволяє користувачам аналізувати і відстежувати історичні показники різних криптовалют. Вона може відображати інтерактивні графіки цін, свічкові діаграми, лінійні графіки або інші візуалізації для демонстрації цінових тенденцій, обсягів торгів або ринкових індикаторів. Користувачі можуть налаштовувати параметри графіків, застосовувати інструменти технічного аналізу та отримувати інформацію про ринкові закономірності, щоб приймати обґрунтовані інвестиційні рішення.

Сформулювавши вимоги, можна зрозуміти, що потрібно розробити шість сторінок. Для цього ми використовуючи патерн MVVM створимо розбивку компонентів що потрібні для цих шести сторінок, це допоможе нам полегшити розробку та встановити правильне зв'язування даних між Views і ViewModels за допомогою XAML і використовувати команди для обробки взаємодії з користувачем.

Ось та розбивка компонентів, які потрібно створити для кожної з шести сторінок (Рис.3.1.):

1. Home Page:
    - Model: Моделі даних, що представляють інформацію, яка буде відображатися на головній сторінці, наприклад, провідні криптовалюти або оновлення новин.
    - View: Макет XAML для головної сторінки, включаючи відповідні прив'язки до ViewModel.
    - ViewModel: Модель відображення яка отримує необхідні дані з API або інших джерел і відображає їх у View. Вона також повинна обробляти

навігацію на інші сторінки.

2. Settings Page:
   - Model: Будь-які моделі даних, пов'язані з налаштуваннями.
   - View: Макет XAML для сторінки налаштувань, включаючи елементи керування для взаємодії з користувачем.
   - ViewModel: Модель відображення відповідальну за обробку користувацьких налаштувань та оновлення відповідних параметрів.

3. Currency Page:
   - Model: Моделі даних, що представляють інформацію про криптовалюту, таку як ціна або ринкова капіталізація.
   - View: Макет XAML для сторінки валюти, включаючи прив'язки для відображення даних про криптовалюту.
   - ViewModel: Модель відображення яка отримує дані про криптовалюту з API і відображає їх у View. Вона також повинна обробляти будь-яку додаткову логіку, пов'язану з вибором або фільтрацією валют.

4. Exchange Page:
   - Model: Моделі даних, що представляють торгові ордери.
   - View: Макет XAML для сторінки обміну, включаючи елементи керування для розміщення ордерів і відображення відповідної інформації.
   - ViewModel: Модель відображення, що відповідає за виконання торгових операцій, отримання історії транзакцій та управління криптовалютним портфелем користувача.

5. Markets Page:

- Model: Моделі даних, що представляють ринкові індекси, коливання цін або інші відповідні ринкові дані.
- View: Макет XAML для сторінки ринків, включаючи прив'язки для відображення ринкової інформації.
- ViewModel: Модель відображення, яка отримує ринкові дані з API і відображає їх у поданні. Вона також може обробляти додаткову логіку для аналізу ринку або надання інформації користувачеві.

6. Charts Page:
   - Model: Моделі даних, що представляють дані та конфігурації графіків.
   - View: Макет XAML для сторінки діаграм, включно з елементами керування для налаштування відображення діаграми.
   - ViewModel: Модель відображення, що відповідає за отримання даних діаграми, застосування необхідних перетворень і відображення їх у поданні для рендерингу.

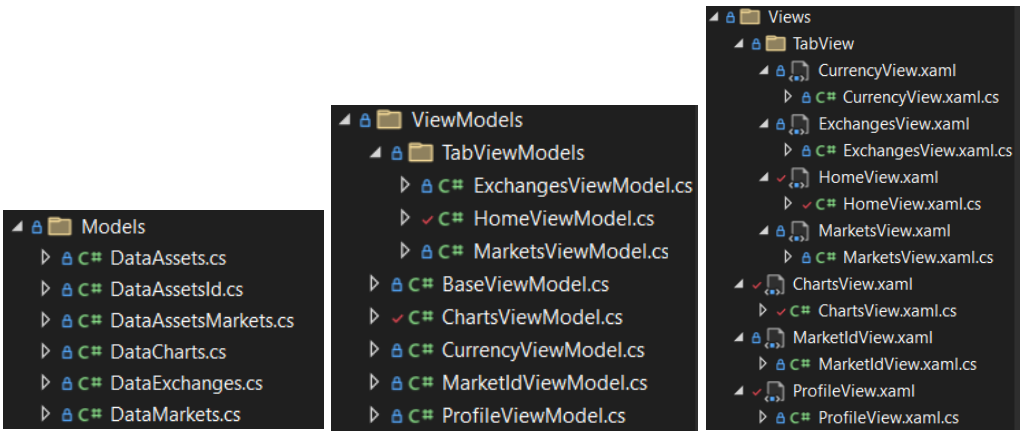

Рис. 3.1. Зображення розбивки додатку по патерну MVVM:

а – Models; *б* – VieModels; *в* – Views

Частина з логуванням (Рис.3.2.)

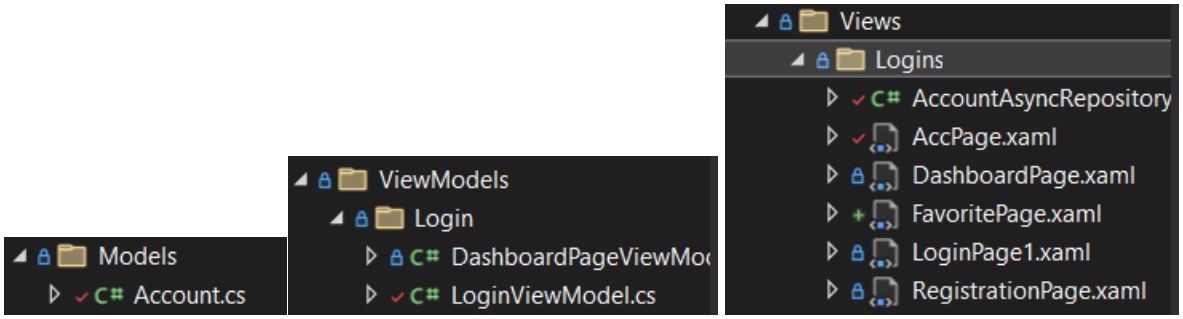

Рис. 3.2. Патерн розробки для створення профілю користувача

Додатково також потрібно буде створити базову модель відображення що б зменшити кількість повторюваного коду.

Також створимо окремий клас який відповідатиму за взаємодію з API. Цей клас відповідатиме за створення HTTP-запитів, розбір відповіді та перетворення її у відповідні моделі даних. Він інкапсулює комунікаційну логіку API і виступає в якості рівня абстракції між ViewModel і API (Рис.3.3.).

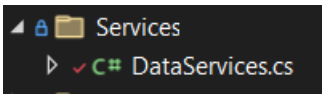

Рис. 3.3. Клас запитів для API

Також окремо створимо місця збереження наших стилей, шрифтів, тем(Рис.3.4.).

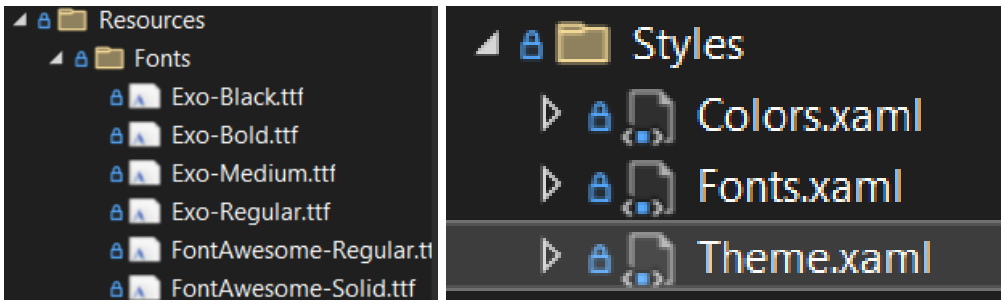

Рис.3.4. *а* – Шрифти; *б* – Теми та стилі

3.2. Розробка функціоналу для відображення даних та графічних елементів

Почнемо з того що ми вже створили, а саме з моделей для зчитування та

майбутніх данних так як вони будуть незмінні. Для початку нам потрбно буде отримати перші данні з нашого API для цього можна використати два способи:

- За допомогою url - api.coincap.io/v2/{тип даних}
- За допомогою консолі вписуючи веб сайт, наш ключ до API та тип авторизації.

За допомогою url - api.coincap.io/v2/assets ит отримаємо дані активів (Рис.3.5.).


{"data":
[{"id":"bitcoin","rank":"1","symbol":"BTC","name":"Bitcoin","supply":"19381531.0000000000000000","maxSupply":"21000000.0000000000000000","marketCapUsd":"529446837534.0620666276641373","volumeUsd24Hr":"363979220
0.2036338680123103","priceUsd":"27317.0802417034065383","changePercent24Hr":"1.2359414157766331","vwap24Hr":"27215.2552573497670898","explorer":"https://blockchain.info/"},
{"id":"ethereum","rank":"2","symbol":"ETH","name":"Ethereum","supply":"120266407.1670825200000000","maxSupply":null,"marketCapUsd":"222675786911.4208805095815175","volumeUsd24Hr":"266238432.4228448704633203","
priceUsd":"1852.2843756522678795","changePercent24Hr":"1.6956585620371175","vwap24Hr":"1843.6831794005818970","explorer":"https://etherscan.io/"},
{"id":"tether","rank":"3","symbol":"USDT","name":"Tether","supply":"82962559131.3005800000000000","maxSupply":null,"marketCapUsd":"83010267286.5415715271640445","volumeUsd24Hr":"7203559770.1494845356905856","pr
iceUsd":"1.0005750564560754","changePercent24Hr":"0.0335143077094591","vwap24Hr":"1.0000700114360612","explorer":"https://www.omniexplorer.info/asset/31"},{"id":"binance-
coin","rank":"4","symbol":"BNB","name":"BNB","supply":"166801148.0000000000000000","maxSupply":"166801148.0000000000000000","marketCapUsd":"52282629320.2983850106658488","volumeUsd24Hr":"148108815.7739656577033
666","priceUsd":"313.4428626372426706","changePercent24Hr":"1.1796762286818668","vwap24Hr":"312.1470341931999536","explorer":"https://etherscan.io/token/0xB8c77482e45F1F44dE1745F52C74426C631bDD52"},{"id":"usd-
coin","rank":"5","symbol":"USDC","name":"USD
Coin","supply":"29403407188.6838100000000000","maxSupply":null,"marketCapUsd":"29412725093.4246308083904442","volumeUsd24Hr":"597091782.0670220833735885","priceUsd":"1.0003168988097545","changePercent24Hr":"-0
027309900192321 5","vwap24Hr":"1.0000289502011058","explorer":"https://etherscan.io/token/0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48"},
{"id":"xrp","rank":"6","symbol":"XRP","name":"XRP","supply":"45404028640.0000000000000000","maxSupply":"100000000000.0000000000000000","marketCapUsd":"20966012862.562350612183840","volumeUsd24Hr":"344072141.8
63809048185015","priceUsd":"0.4617654752356431","changePercent24Hr":"0.4050232940216199","vwap24Hr":"0.4612113646758240","explorer":"https://xrpcharts.ripple.com/#/graph/"},
{"id":"cardano","rank":"7","symbol":"ADA","name":"Cardano","supply":"34865581796.9030000000000000","maxSupply":"45000000000.0000000000000000","marketCapUsd":"12945561348.610624450213843 4","volumeUsd24Hr":"52633
321.6856392255301297","priceUsd":"0.3712991632843063","changePercent24Hr":"0.6968046542513439","vwap24Hr":"0.3703238952378570","explorer":"https://cardanoexplorer.com/"},
{"id":"dogecoin","rank":"8","symbol":"DOGE","name":"Dogecoin","supply":"139463966383.7052000000000000","maxSupply":null,"marketCapUsd":"10171101340.3690041405291957","volumeUsd24Hr":"63304946.1723451720526418","
priceUsd":"0.0729299589284977","changePercent24Hr":"-0.2696290057292430","vwap24Hr":"0.0731690227912427","explorer":"http://dogechain.info/chain/Dogecoin"},
{"id":"polygon","rank":"9","symbol":"MATIC","name":"Polygon","supply":"9279469069.2849300000000000","maxSupply":"10000000000.0000000000000000","marketCapUsd":"8223475577.161121866216598","volumeUsd24Hr":"6914
713.7217062537721496","priceUsd":"0.8862010871269403","changePercent24Hr":"2.0251378051698811","vwap24Hr":"0.8783779266125000","explorer":"https://etherscan.io/token/0x7D1AfA7B718fb893dB30A3aBc0Cfc608AaCfeB0"
},
{"id":"solana","rank":"10","symbol":"SOL","name":"Solana","supply":"396086552.9655501000000000","maxSupply":null,"marketCapUsd":"7881513148.5399514004920182","volumeUsd24Hr":"70977916.4957216595924416","priceUs
d":"19.8984618227232670","changePercent24Hr":"0.4535740995107533","vwap24Hr":"19.7029944591354164","explorer":"https://explorer.solana.com/"},
{"id":"tron","rank":"11","symbol":"TRX","name":"TRON","supply":"90327765146.0770300000000000","maxSupply":null,"marketCapUsd":"7040825097.5991499217516032","volumeUsd24Hr":"89369803.4442231469167352","priceUsd
":"0.0779475179775876","changePercent24Hr":"-1.5308903748890269","vwap24Hr":"0.0786994977564458","explorer":"https://tronscan.org/#/"},
{"id":"litecoin","rank":"12","symbol":"LTC","name":"Litecoin","supply":"72983464.3093235200000000","maxSupply":"84000000.0000000000000000","marketCapUsd":"6693520255.2704583569364895","volumeUsd24Hr":"139847970
.6062943253414358","priceUsd":"91.7128327438873291","changePercent24Hr":"0.1714661183111311","vwap24Hr":"91.6848671940180945","explorer":"http://explorer.litecoin.net/chain/Litecoin"},
{"id":"polkadot","rank":"13","symbol":"DOT","name":"Polkadot","supply":"1235089495.7744500000000000","maxSupply":null,"marketCapUsd":"6658761143.4784816472803531","volumeUsd24Hr":"40025791.4139325877693205","pr
iceUsd":"5.3913187394595844","changePercent24Hr":"1.4852123347946329","vwap24Hr":"5.3587839423974515","explorer":"https://polkascan.io/polkadot"},{"id":"binance-usd","rank":"14","symbol":"BUSD","name":"Binance
USD","supply":"5460859528.5733760000000000","maxSupply":null,"marketCapUsd":"5462904488.6629504878442609","volumeUsd24Hr":"225334361.4115247029272115","priceUsd":"1.0003744758638955","changePercent24Hr":"-0.010
704075797743","vwap24Hr":"1.0002219055011851","explorer":"https://etherscan.io/token/0x4fabb145d64652a948d72533023f6E7A623C7C53"},{"id":"shiba-inu","rank":"15","symbol":"SHIB","name":"Shiba
Inu","supply":"589355039935792.8000000000000000","maxSupply":null,"marketCapUsd":"5273924789.1097195989258463","volumeUsd24Hr":"5723124.5506247847045278","priceUsd":"0.0000089486378019","changePercent24Hr":"1
2338805767773770","vwap24Hr":"0.0000089176462402","explorer":"https://etherscan.io/token/0x95ad61b0a150d79219dcf64e1e6cc01f0b64c4ce"},
{"id":"avalanche","rank":"16","symbol":"AVAX","name":"Avalanche","supply":"334589990.6926298000000000","maxSupply":"720000000.0000000000000000","marketCapUsd":"4930595397.8778966551020019","volumeUsd24Hr":"360
7409.8480165508756795","priceUsd":"14.7362310141760352","changePercent24Hr":"0.8432921110341058","vwap24Hr":"14.7453995515712272","explorer":"https://avascan.info/"},{"id":"multi-collateral-
dai","rank":"17","symbol":"DAI","name":"Multi Collateral
DAI","supply":"4852805772.7768540000000000","maxSupply":null,"marketCapUsd":"4857165650.2249576277792597","volumeUsd24Hr":"25867452.2871090258431277","priceUsd":"1.0008984240565657","changePercent24Hr":"-0.048
121638888660","vwap24Hr":"1.0003257348553884","explorer":"https://etherscan.io/token/0x89d24a6b4ccb1b6faa2625fe562bdd9a23260359"},{"id":"wrapped-bitcoin","rank":"18","symbol":"WBTC","name":"Wrapped
Bitcoin","supply":"156148.9836121400000000","maxSupply":null,"marketCapUsd":"4265431319.2356335017109317","volumeUsd24Hr":"35273456.6709822999258796","priceUsd":"27316.4206424204489994","changePercent24Hr":"1.1
553087463956183","vwap24Hr":"27142.7164676916203721","explorer":"https://etherscan.io/token/0x2260fac5e5542a773aa44fbcfedf7c193bc2c599"},
{"id":"chainlink","rank":"19","symbol":"LINK","name":"Chainlink","supply":"517099970.4527867000000000","maxSupply":"1000000000.0000000000000000","marketCapUsd":"3386728523.2576187255760759","volumeUsd24Hr":"42
60134.4480101219032936","priceUsd":"6.5494657063935002","changePercent24Hr":"0.1828646271809175","vwap24Hr":"6.5484012342370792","explorer":"https://etherscan.io/token/0x514910771af9ca656af840dff83e8264ecf986ca
"},{"id":"unus-sed-leo","rank":"20","symbol":"LEO","name":"UNUS SED
LEO","supply":"930302105.9000000000000000","maxSupply":null,"marketCapUsd":"3284351666.587964876311 2099","volumeUsd24Hr":"115643.9696410393264249","priceUsd":"3.5304140942587593","changePercent24Hr":"0.2294892
46097957","vwap24Hr":"3.5149734337140915","explorer":"https://eospark.com/account/bitfinexleo1"},

Рис.3.5. Дані за запитом Get в api.coincap.io

Скопійовані дані нам знадобляться для створення класу зчитування та розпізнання. Для цього зайдемо в Visual Studio та в перейдемо в класс DataAssets, оберемо View/Paste Special/Paste JSON As Classes (Рис.3.6.) це допоможе згенерувати клас від скопійованого JSON за для уникнення помилок в написанні вручну. Інші класи обробляємо так само.

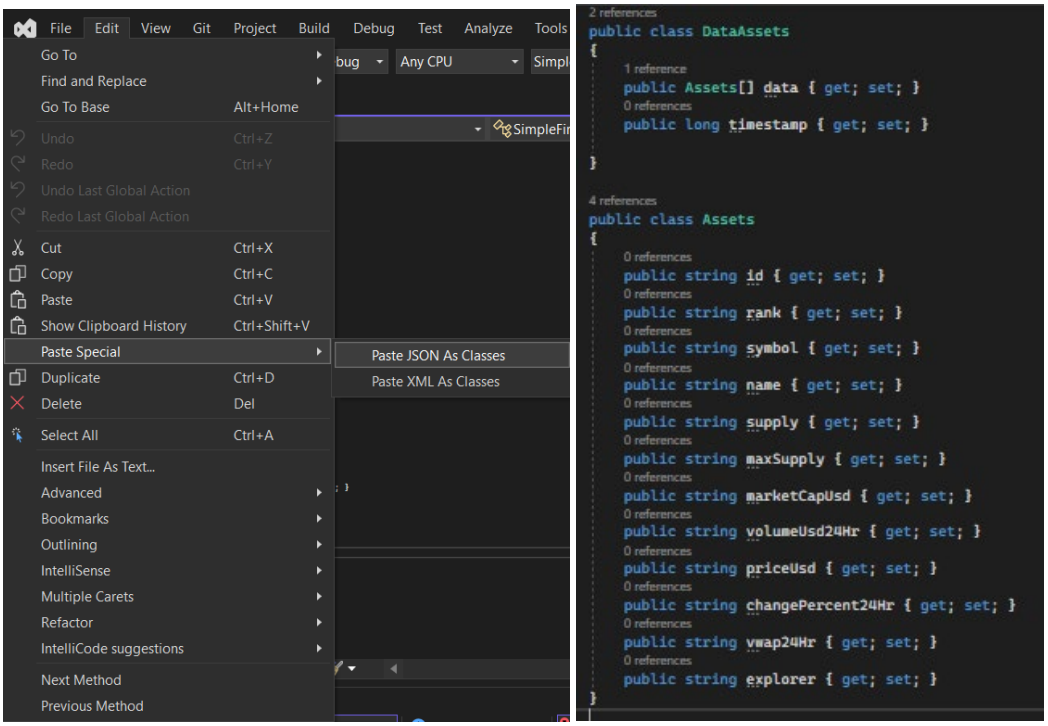

Рис.3.6. *а* – Функція; *б* – Результат генерації

Наступним потрібно написати скрипти отримання даних з API та десерелізацію JSON файлів. Для цього в папці Services створено клас DataServices, який обробляє зв'язок з REST API для отримання даних, пов'язаних з криптовалютними активами, ринками, біржами та графіками(Рис.3.7.). Клас має приватне поле _httpClient типу HttpClient, яке буде використовуватися для надсилання HTTP-запитів. Конструктор ініціалізує поле _httpClient і встановлює базову адресу HTTP-клієнта в "https://api.coincap.io/v2/". Це означає, що всі наступні запити, зроблені цим клієнтом, будуть відносно цієї базової адреси. Клас надає декілька загальнодоступних методів для отримання різних типів даних з API, і кожен метод повертає Task, що представляє асинхронну операцію:

- GetAssetsAsync() надсилає HTTP GET-запит до кінцевої точки "assets" і отримує список криптовалютних активів. Відповідь десеріалізується за допомогою JsonConvert.DeserializeObject в об'єкт типу DataAssets і повертається.
- GetAssetsIdAsync(рядок Id) надсилає HTTP GET-запит на кінцеву точку "assets/{Id}", де "{Id}" є параметром, що вказує конкретний ідентифікатор

активу. Він отримує детальну інформацію про конкретний криптовалютний актив, ідентифікований за його ідентифікатором. Відповідь десеріалізується в об'єкт типу DataAssetsId і повертається.

- GetMarketsAsync() надсилає HTTP GET-запит до кінцевої точки "markets" і отримує список криптовалютних ринків. Відповідь десеріалізується в об'єкт типу DataMarkets і повертається.
- GetExchangesAsync() надсилає HTTP GET-запит до кінцевої точки "exchanges" і отримує список криптовалютних бірж. Відповідь десеріалізується в об'єкт типу DataExchanges і повертається.
- GetMarketsIdAsync(string market) надсилає HTTP GET-запит до кінцевої точки "assets/{market}/markets", де "{market}" є параметром, що вказує ринок. Він отримує список ринків для певного криптовалютного активу, ідентифікованого за назвою ринку. Відповідь десеріалізується в об'єкт типу DataAssetsMarkets і повертається.
- GetChartsAsync(string assetId, string date) надсилає HTTP GET-запит до кінцевої точки "assets/{assetId}/history?interval={date}", де "{assetId}" - параметр, що вказує ідентифікатор активу, а "{date}" - параметр, що вказує інтервал. Він отримує історичні дані графіка для конкретного криптовалютного активу в межах заданого інтервалу. Відповідь десеріалізується в об'єкт типу DataCharts і повертається.

```
private readonly HttpClient _httpClient;

7 references
public DataServices()
{
    _httpClient = new HttpClient();
    _httpClient.BaseAddress = new Uri("https://api.coincap.io/v2/");
}

1 reference
public async Task<DataAssets> GetAssetsAsync()
{
    var response = await _httpClient.GetAsync("assets");
    response.EnsureSuccessStatusCode();

    var content = await response.Content.ReadAsStringAsync();
    return JsonConvert.DeserializeObject<DataAssets>(content);
}
```

Рис.3.7. Загальнодоступний метод та конструктор HTTP-запитів.

Перейдемо до папки ViewModels та створимо базовий клас BaseViewModel,

який реалізує інтерфейс INotifyPropertyChanged для повідомлення про зміни властивостей. Основні властивості класу включають:

- IsBusy: булеве значення, яке вказує, чи виконується в даний момент асинхронна операція.
- Title: рядок, що містить заголовок моделі.
- Asset: список об'єктів Assets, який зберігає активи.
- Exchange: список об'єктів Exchanges, який зберігає обмінники.
- Market: список об'єктів AssetsMarkets, який зберігає ринки активів.
- Charts: список об'єктів Charts, який зберігає діаграми.

У класі також є методи для асинхронного завантаження даних з використанням DataServices. Наприклад:

- LoadAssetsAsync(): завантажує список активів із використанням GetAssetsAsync і зберігає їх у властивості Asset.
- LoadAssetsAsync(string id): завантажує інформацію про актив за заданим ідентифікатором id і зберігає ці дані у властивості firstItem.
- LoadExchangesAsync(): завантажує список обмінників із використанням GetExchangesAsync і зберігає їх у властивості Exchange.
- LoadMarketsAsync(string id): завантажує список ринків для заданого ідентифікатора активу id і зберігає їх у властивості Market.
- LoadChartsAsync(string i, string b): завантажує діаграми для заданих параметрів i і b і зберігає їх у властивості Charts.

Клас також містить реалізацію методів, необхідних для інтерфейсу INotifyPropertyChanged, а саме SetProperty для встановлення значення властивості та виклику подій PropertyChanged та OnPropertyChanged для сповіщення про зміни властивостей. Конструктор BaseViewModel створює новий об'єкт DataServices, який його використовує для отримання даних. Це може бути клас або служба, яка надає методи для взаємодії з певним джерелом даних, наприклад, отримання

активів, обмінників, ринків та діаграм. Клас містить реалізацію інтерфейсу INotifyPropertyChanged, який дозволяє сповіщати про зміни значень властивостей, які можуть бути прив'язані до інтерфейсу користувача (UI). Це дозволяє оновлювати відображення даних при їх зміні. Код включає використання декоратора [CallerMemberName] для визначення імені властивості, коли властивість змінюється. Це дозволяє уникнути жорсткого кодування імен властивостей при виклику методу OnPropertyChanged. Усі властивості, які пов'язані з даними, використовують метод SetProperty для установки нового значення та сповіщення про зміни властивостей. Це допомагає підтримувати узгодженість даних між моделлю та інтерфейсом користувача.

Загалом, цей код представляє базову модель для управління даними та їх відображенням. Він надає спільні властивості та методи, які можуть бути успадковані та розширені в похідних класах для виконання конкретних завдань та обробки даних.

Розглянемо дві моделі представлення HomeViewModel та MarketsViewModel (Рис. 3.8.). На зображеннях видно представлені моделі які успадковуються від базової моделі BaseViewModel, а також мають в собі базові конструктори в якому встановлюється заголовок Title для сторінок та викликається асинхронний метод що завантажує дані представлення на сторінку. В MarketsViewModel ми можемо побачити додаткові конструктори що викликають пеший конструктор, вони потрібні задля створення сортування заготовлених даних. Таким чином, цей код дозволяє створювати екземпляри MarketsViewModel з різними конструкторами, які встановлюють заголовок та виконують асинхронне завантаження ринків з використанням певних параметрів.

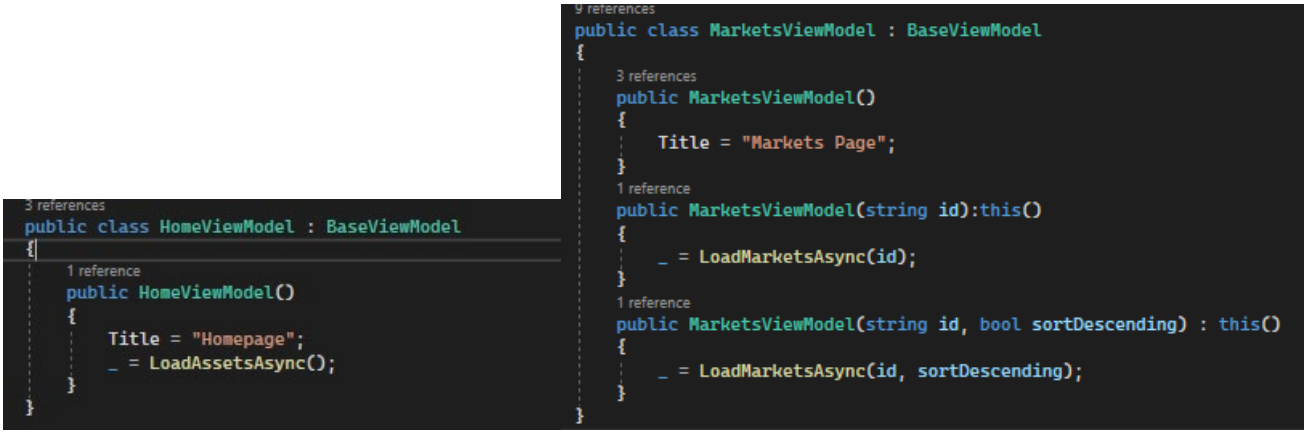

Рис. 3.8. *а* – HomeViewModel; *б* – MarketsViewModel

Точно так само створюємо й інші моделі представлення, але розглянемо додатково ще ProfileViewModel(сторінка налаштувань). Ця модель представлення має наступні елементи:

- Команди:
  - BackCommand: ICommand, використовується для обробки команди повернення назад.
  - DarkModeToggleCommand: ICommand, використовується для обробки команди перемикання між темною та світлою темами.
- Властивості:
  - Init: Task, представляє завдання для ініціалізації моделі представлення.
  - IsDarkMode: bool, вказує, чи ввімкнений темний режим.
- Конструктори:
  - ProfileViewModel(): Конструктор, в якому ініціалізуються команди BackCommand та DarkModeToggleCommand, а також викликається метод Initialize() для ініціалізації властивості IsDarkMode.
  - Initialize(): Асинхронний метод, який ініціалізує властивість IsDarkMode на основі поточної теми застосунку.
- Методи-обробники команд:
  - BackCommandHandler(): Обробник команди повернення назад, який викликає перехід до сторінки "HomeView".

- DarkModeToggleCommandHandler(): Обробник команди перемикання теми, який змінює тему застосунку на основі властивості IsDarkMode і зберігає вибрану тему у налаштуваннях за допомогою Preferences.

Отже, цей код визначає модель представлення ProfileViewModel, яка містить команди для обробки подій та властивості для збереження стану. Вона також має конструктори для ініціалізації та методи-обробники команд для виконання відповідних дій.

Перейдемо до самого відображення сторінок в папку Views. Для початку нам потрібно буде створити інтерфейс програми, тому застосуємо XAML та створимо робочу область та розмістимо кнопки та місця відображення списків (Рис. 3.9.).

```
<ScrollView>
    <StackLayout
        BackgroundColor="{AppThemeBinding Light={StaticResource LightPageBackgroundColor}, Dark={StaticResource DarkPageBackgroundColor}}">
        <Grid
            ColumnDefinitions="*, Auto"
            RowDefinitions="Auto, Auto"
            RowSpacing="0"
            Margin="20, 10, 20, 5">
            <Label
                Grid.Column="0"
                Grid.Row="0"
                Style="{StaticResource Body1FontSize_ExoBold}"
                Text="{Binding Title}"
                TextColor="{AppThemeBinding Light={StaticResource LightSecondaryTextColor}, Dark={StaticResource DarkSecondaryTextColor}}"></Label>
            <buttons:SfButton Text="I" Clicked="ProfileButton_Click" CornerRadius="20" WidthRequest="40" BorderWidth="2"
                TextColor="{AppThemeBinding Light={StaticResource LightSecondaryTextColor}, Dark={StaticResource DarkSecondaryTextColor}}"
                BorderColor="{AppThemeBinding Light={StaticResource DarkPageBackgroundColor}, Dark={StaticResource LightPageBackgroundColor}}"
                BackgroundColor="{AppThemeBinding Light={StaticResource LightPageBackgroundColor}, Dark={StaticResource DarkPageBackgroundColor}}"
                Grid.Column="1"
                Grid.Row="0"
                Grid.RowSpan="2">
            </buttons:SfButton>

        </Grid>
        <Grid>
            <Grid.RowDefinitions>
                <RowDefinition Height="Auto" />
                <RowDefinition Height="*" />
            </Grid.RowDefinitions>

            <!-- Column headers -->
            <Grid.ColumnDefinitions>
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
                <ColumnDefinition Width="*" />
            </Grid.ColumnDefinitions>
```

Рис. 3.9. Вигляд розробки стартової сторінки HomePage.xaml

Розробивши відображення для кожної сторінки потрібно реалізувати контекст прив’язки даних на екземпляр та методи натискання на кнопку або вибір елемента з списку. Ось приклад розробки back end для HomeView:

HomeView який успадковує від класу ContentPage і має атрибут [XamlCompilation(XamlCompilationOptions.Compile)].

Основний функціонал цього класу:

- Визначається приватна змінна _viewModel типу HomeViewModel, яка

представляє модель представлення для цієї сторінки.

- В конструкторі HomeView() виконується налаштування інтерфейсу користувача шляхом виклику методу InitializeComponent(). Також встановлюється контекст прив'язки даних BindingContext на екземпляр HomeViewModel, який створюється.
- Метод ProfileButton_Click() викликається при натисканні на кнопку "ProfileButton". Він виконує перехід до сторінки LoginPage1 за допомогою методу Navigation.PushAsync().
- Перевизначений метод OnAppearing() викликається при з'явленні сторінки. Викликається відповідний метод _viewModel.OnAppearing() для виконання додаткових дій при з'явленні сторінки.
- Метод OnButtonClicked() викликається при натисканні на кнопку, яка має подію Clicked. У цьому методі перевіряється наявність збереженого ключа UserAlreadyloggedIn за допомогою Xamarin.Essentials.Preferences.Get(). Якщо ключ існує, то отримується поточний об'єкт RelatedObject з контексту прив'язки даних і зберігається в базі даних. Якщо ключ не існує, виконується перехід до сторінки LoginPage1.
- Метод OnItemSelected() викликається при виборі елемента зі списку. Він отримує обраний елемент Assets і передає його ідентифікатор до сторінки ChartsView для відображення графіків.

По такій же технології створимо можливість реєстрації та логування користувача (Рис.3.2). В моделі Account описую два класи, Account і RelatedObject, які представляють таблиці бази даних в Xamarin проекті з використанням ORM (Object-Relational Mapping) підходу. Клас Account відповідає таблиці "Accounts" у базі даних. Він має наступні властивості:

- Id: Це первинний ключ таблиці, який автоматично інкрементується при вставці нового запису.
- FirstName, LastName, Email, UserName, Password: Ці властивості

відповідають стовпцям таблиці "Accounts" і містять інформацію про ім'я, прізвище, електронну пошту, ім'я користувача та пароль відповідно.

- RelatedObjects: Це властивість, яка вказує на зв'язок один до багатьох між таблицями "Accounts" і "RelatedObject". Кожен об'єкт Account може мати декілька об'єктів RelatedObject. Зв'язок встановлюється за допомогою зовнішнього ключа.

Клас RelatedObject відповідає таблиці "RelatedObject" у базі даних. Він має наступні властивості:

- Id: Це первинний ключ таблиці, який автоматично інкрементується при вставці нового запису.
- SomeData: Ця властивість відповідає стовпцю "SomeData" у таблиці "RelatedObject" і містить деякі дані.
- UserId: Це зовнішній ключ, який вказує на зв'язок з таблицею "Accounts". Значення цього ключа відповідає значенню стовпця Id у таблиці "Accounts", до якої відноситься даний об'єкт RelatedObject.

Ці класи описують структуру таблиць бази даних та встановлюють зв'язок між ними за допомогою властивостей RelatedObjects у класі Account і UserId у класі RelatedObject. LoginViewModel є моделлю представлення для сторінки входу в систему (login page) в Xamarin проекті. Деякі ключові аспекти коду включають:

- Властивість MyloginRequestModel: Це властивість, яка представляє модель запиту на вхід, яка містить дані користувача, такі як ім'я користувача та пароль. Властивість має встановлення, яке спрацьовує при зміні значення і викликає подію PropertyChanged, що вказує на зміну властивості.
- Властивість LoginCommand: Це команда, яка виконується при натисканні кнопки входу. Вона викликає метод PerformLoginOperation.
- Метод PerformLoginOperation: Цей метод виконує операцію входу, яка може бути API-запитом або операцією з базою даних. Він отримує дані введені користувачем з моделі MyloginRequestModel і перевіряє їх зі списком

облікових записів, отриманих з бази даних за допомогою App.Database.GetAccountItemsAsync(). Якщо знайдено відповідні облікові дані, встановлюється прапорець UserAlreadyloggedIn в налаштуваннях (Preferences), встановлюється значення UserLogin з імені користувача та навігація відбувається на сторінку "DashboardPage".

Також що б це все працювало зпотрібно створити базу даних (Рис. 3.10.).

```
public partial class App : Application
{
    public const string DATABASE_NAME = "accounts.db";
    public static AccountAsyncRepository database;
    7 references
    public static AccountAsyncRepository Database
    {
        get
        {
            if (database == null)
            {
                string dbPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.LocalApplicationData), DATABASE_NAME);

                database = new AccountAsyncRepository(dbPath);
            }
            return database;
        }
    }
    2 references
```

Рис.3.10. Створення бази даних

Загалом, цей код забезпечує доступ до бази даних AccountAsyncRepository через статичну властивість Database. Це дозволяє іншим частинам програми отримати доступ до бази даних для зберігання та отримання об'єктів типу Account.

3.3. Демонстрація роботи застосунку

Додаток має чотири основні сторінки: Головна, Зміни, Ринок та Біржі:

1. На сторінці Home відображається список всіх криптовалют, які в даний момент відстежуються додатком. Користувач може сортувати список за ціною, ринковою капіталізацією або обсягом. Користувач також може переглянути графік історії цін для кожної криптовалюти.
2. На сторінці Markets відображається список всіх цін на обрану криптовалютю в різних криптових біржах. Користувач може сортувати список за зміною ціни.
3. На сторінці Exchanges відображається список всіх бірж, на яких можна купити і продати криптовалюту.
4. На сторінці Changes користувач може переглянути відносну ціну однієї

криптовалюти до іншої.

Основні сторінки зображені на рис.3.11.

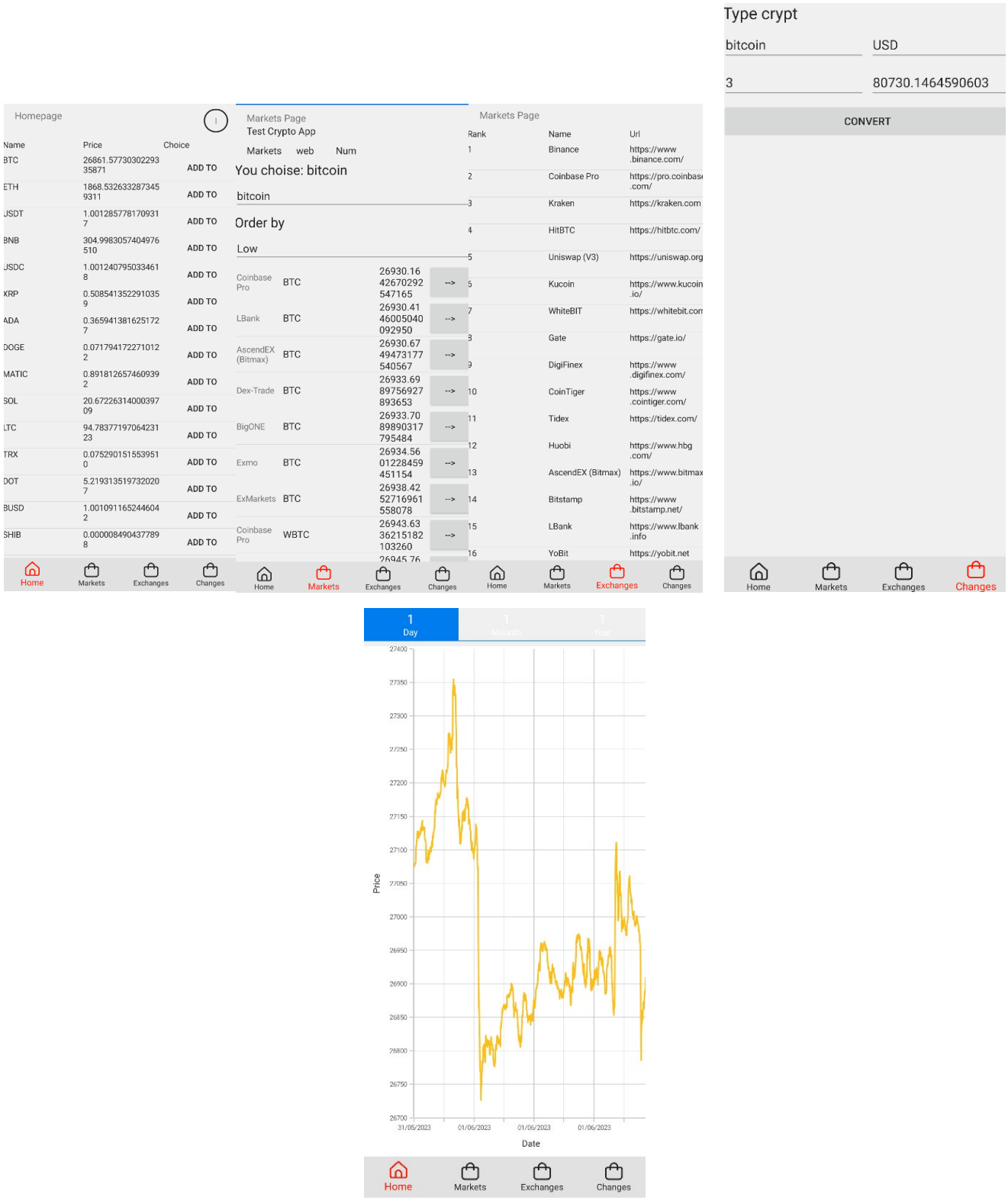

Рис.3.11. *а* – Home; *б* – Markets; *в* – Exchanges; *г* – Changes; *д* – Charts

На додаток до чотирьох основних сторінок, у додатку також є сторінка входу, сторінка реєстрації, сторінка "Вибране" і сторінка налаштувань.

- Сторінка входу дозволяє користувачам створити обліковий запис або увійти в існуючий обліковий запис.
- Сторінка реєстрації дозволяє користувачам створити новий обліковий запис.
- Сторінка "Вибране" дозволяє користувачам зберігати список улюблених криптовалют.
- Сторінка Налаштування дозволяє користувачам змінювати зовнішній вигляд і поведінку додатку.

Додаткові сторінки зображені на рис.3.12

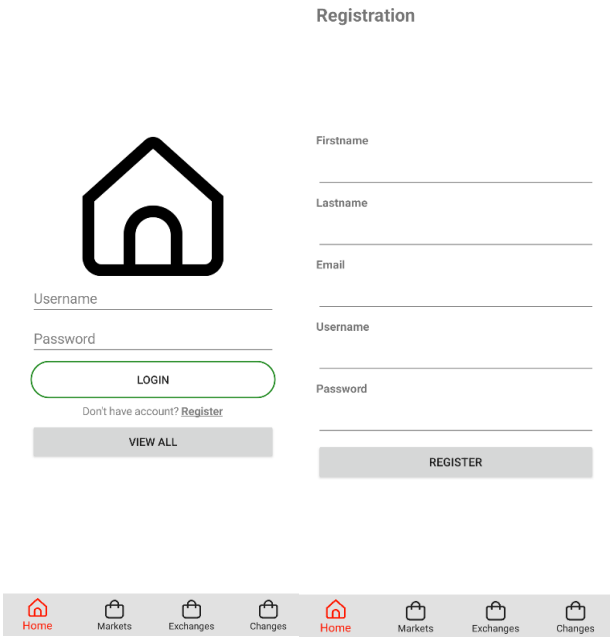

Рис.3.12. *а* – Login; *б* – Registration;

ВИСНОВКИ

Розроблено мобільний додаток для відображення даних по крипто валюті з використанням фреймворків .Net та Xamarin. Додаток надає користувачам, які слідкують за ринком криптовалют, платформу для доступу до великої кількості інформації з бірж по різним криптовалютам.

Додаток використовує API, що дало можливість інтегрувати зовнішні джерела даних та отримувати з них інформацію про: біржі, криптовалюти, ціни, капіталізацію. Це дозволило забезпечити додаток всією потрібною інформацією в реальному часі, що підвищує цінність та важливість додатку.

Використання фреймворку Xamarin для розроблення мобільного додатку дало можливість створити застосунок з простим та інтуїтивним користувацьким інтерфейсом, та достатнім функціоналом, що покращує клієнтський досвід. Для аутентифікації та збереження налаштувань користувачів використовувалася база даних SQLite. Використання бази даних дало можливість зберігати вподобану криптовалюту та біржі, що потрібні користувачу.

Аналіз існуючих додатків по виводу криптовалюти виявив їхні недоліки: погана оптимізація, пристуність платного контенту за основний функціонал, наявність реклами, відсутність сортувань та огляд цін на інших біржах крім основних. У розробленому додатку було враховано всі ці недоліки.

В майбутньому планується покращити інтерфейс, додати можливість перегляду останніх новин по крипто валюті, впровадження різних методів аутентифікації, та можливість налаштовувати сповіщення за вподобаними криптовалютами, покращення графіків з можливістю оновлення похвилинно та можливість локалізації на потрібні мови.

Розроблений мобільний додаток для відображення даних ринку криптовалют може бути важливим внеском у сферу криптовалют та фінансових технологій, надаючи користувачам зручний і надійний інструмент для відстеження та аналізу даних ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Jon Skeet. C# in Depth , 3rd Edition. Вид-во Manning Publications, 2013. 616 с.
2. Charles Petzold. Microsoft Press eBook Creating Mobile App with Xamarin Forms. Вид-во Microsoft Press, 2016. 1161 с.
3. Jon Skeet. Xamarin in Action: Creating native cross-platform mobile apps First Edition. Вид-во Manning, 2018. с. 608
4. Mark J. Price. C# 10 and .NET 6 – Modern Cross-Platform Development: Build apps, websites, and services with ASP.NET Core 6, Blazor, and EF Core 6 using Visual Studio 2022 and Visual Studio Code, 6th Edition. Вид-во Packt Publishing, 2022. 826 с.
5. Visual Studio [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://visualstudio.microsoft.com/
6. CoinCap API [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://docs.coincap.io/
7. JSON [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: www.json.org/
8. Git [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://git-scm.com/about
9. Xamarin.Forms [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://learn.microsoft.com/en-gb/xamarin/get-started/what-is-xamarin-forms
10. Xamarin [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://dotnet.microsoft.com/en-us/apps/xamarin
11. XAML [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://wpf-tutorial.com/xaml/what-is-xaml/
12. Java [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://www.java.com/en/download/help/whatis_java.html
13. Python [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу:

https://www.python.org/about/

14. Android Studio [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://developer.android.com/studio
15. UI/UX Design for Android [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://mockitt.wondershare.com/ui-ux-design/android-ui-design.html
16. UI/UX Design Tools [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://www.uxdesigninstitute.com/blog/user-interface-ui-design-tools/
17. C# [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://docs.microsoft.com/en-us/dotnet/csharp/
18. Blockfolio [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.blockfolio.cryptotracker&hl=uk&gl=US
19. Delta [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=io.getdelta.android&hl=uk&gl=US
20. CoinMarketCap [Електронний ресурс]: [Веб-сайт] – Режим доступу до ресурсу: https://play.google.com/store/apps/details?id=com.coinmarketcap.android&hl=uk&gl=US