МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет водного господарства та природокористування
Навчально-науковий інститут кібернетики, інформаційних технологій та інженерії
Кафедра комп'ютерних технологій та економічної кібернетики

Допущено до захисту:

Завідувач кафедри

_____ д. е. н., проф. П. М. Грицюк

«_____» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня «бакалавр»
за освітньо-професійною програмою «Інформаційні системи і технології»
спеціальності 126 «Інформаційні системи та технології»

на тему: **«Створення мобільного додатку для моніторингу ринку криптовалют»**

Виконав:
здобувач вищої освіти 5 курсу заочної форми навчання, групи ІСТз-51
Цилія Павло Тарасович

Керівник:
канд. техн. наук, доцент Гладка О. М.

Рецензент:
канд. техн. наук, доцент Джоші О. І.

Рівне – 2025

Зміст

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

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

У нашому столітті цифрові технології стали ключовим рушієм глобальних трансформацій, що охоплюють як економічну, так і соціальну сфери. Однією з найдинамічніших та інноваційних галузей, яка за останні десятиліття набула глобального масштабу та значного впливу на фінансові системи світу, є криптовалюти. Із зростанням популярності цифрових активів, таких як Bitcoin, Ethereum, Ripple та інших, виникла потреба у швидкому, надійному та доступному способі моніторингу ринку криптовалют у режимі реального часу. Ця потреба особливо актуальна для інвесторів, трейдерів, аналітиків та користувачів, які приймають рішення, базуючись на швидкозмінних даних ринку.

У сучасних умовах фінансова інформація оновлюється щосекундно. Користувачі криптовалютного ринку потребують інструментів, які дозволяють їм отримувати найсвіжіші дані щодо курсів валют, обсягів торгів, ринкової капіталізації, змін у динаміці цін, аналітики та інших ключових показників. Традиційні способи отримання такої інформації, зокрема через десктопні платформи чи вебсайти, не завжди є зручними, особливо у ситуаціях, коли потрібно оперативно реагувати на зміни ринку. У зв’язку з цим спостерігається зростаючий попит на мобільні додатки, які забезпечують оперативний доступ до актуальної інформації з будь-якої точки світу.

Мобільні технології та пов’язане з ними програмне забезпечення стрімко розвиваються, відкриваючи нові можливості для реалізації ефективних, зручних та персоналізованих інструментів моніторингу фінансових ринків. Особливої популярності набули кросплатформенні засоби розробки, що дозволяють створювати універсальні рішення для різних операційних систем – Android, iOS тощо. У цьому контексті фреймворки типу Xamarin, Flutter, React Native надають розробникам широкі можливості для побудови функціональних додатків із сучасним дизайном та високою продуктивністю.

Проте, попри наявність великої кількості застосунків для моніторингу крипторинку, більшість із них орієнтовані на англомовного користувача, не враховують локальні особливості, містять надлишкову рекламу, не забезпечують необхідної конфіденційності або мають складний інтерфейс.

Таким чином, виникає потреба у створенні мобільного застосунку, що відповідає актуальним вимогам українських користувачів – зручного, надійного, ефективного та безпечного.

Метою цієї роботи є розроблення кросплатформенного мобільного додатку для моніторингу ринку криптовалют, що забезпечить зручний інтерфейс користувача, інтеграцію з відкритими API криптобірж, відображення актуальних даних у режимі реального часу та забезпечить надійність і безпеку збереження персональних налаштувань користувача.

Об’єктом дослідження є процес моніторингу ринку криптовалют у реальному часі за допомогою мобільних застосунків.

Предметом дослідження є програмні засоби, методи та технології створення кросплатформенних мобільних додатків для відображення динаміки криптовалютних ринків.

Для досягнення поставленої мети необхідно вирішити такі основні завдання:

1. Провести аналіз ринку криптовалют, визначити його структуру, ключові тренди та потреби користувачів у сфері інформаційного моніторингу.

2. Дослідити існуючі мобільні додатки для моніторингу крипторинку, визначити їх функціональні переваги та недоліки.

3. Визначити вимоги до майбутнього застосунку: функціональні, нефункціональні, безпекові, інтерфейсні та технологічні.

4. Провести порівняльний аналіз технологій розробки мобільних застосунків: Xamarin, Flutter, React Native, Kotlin, Swift.

5. Обґрунтувати вибір засобів реалізації мобільного додатку та архітектурного підходу.

6. Розробити макети інтерфейсу користувача, відповідно до принципів UX/UI дизайну.

7. Реалізувати клієнтську частину додатку з використанням обраної технології.

8. Інтегрувати відкриті API криптовалютних бірж (наприклад, Binance API) для отримання реальних даних.

9. Реалізувати механізми збереження налаштувань користувача, кешування та захисту даних.

10. Провести тестування працездатності, продуктивності та безпеки додатку.

11. Розробити інструкцію користувача та оцінити перспективи розвитку проєкту.

У роботі використовуються методи системного аналізу, проєктування програмного забезпечення, порівняльного аналізу технологій, об’єктно-орієнтованого програмування, UI/UX дизайну, тестування, а також методи моделювання архітектур мобільних застосунків.

Yовизна цієї роботи полягає у створенні комплексного підходу до розроблення кросплатформеного мобільного застосунку для криптовалютного ринку, що поєднує ефективне оновлення даних у реальному часі, зручний користувацький інтерфейс, високу продуктивність і безпеку.

Результати кваліфікаційної роботи можуть бути використані для: створення персональних мобільних фінансових асистентів; впровадження у банківські або біржові мобільні сервіси; подальшого навчання і викладання дисциплін, пов’язаних з мобільною розробкою, інформаційною безпекою та криптографією; стартап-проєктів у сфері FinTech в Україні.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел і додатків.

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

1.1. Аналіз ринку криптовалют

Аналіз ринку криптовалют є комплексним дослідженням і оцінкою поточного стану ринку цифрових активів, що проводиться за допомогою різноманітних методів і підходів. Такий аналіз дозволяє глибше зрозуміти особливості функціонування криптовалютного ринку, оцінити його стабільність, тенденції розвитку та можливі ризики. Завдяки систематичному аналізу можна отримати детальні відомості про динаміку змін на ринку та виявити закономірності, що допоможуть прогнозувати його майбутній стан.

У межах дослідження, присвяченого розробці мобільного додатку для відображення даних криптовалютного ринку, аналіз криптовалютного ринку відіграє важливу роль. Він сприяє глибшому розумінню потреб користувачів у подібних мобільних додатках, що дозволяє створити ефективний і зручний інструмент для роботи з ринковими даними. Крім того, аналіз допомагає визначити, які показники та інформаційні джерела слід інтегрувати в додаток, щоб забезпечити його користувачів найбільш актуальною, точною та повною інформацією.

Існує багато методів аналізу ринку криптовалют, кожен з яких дозволяє отримати конкретні відомості про певний аспект ринку. Найбільш популярними та ефективними методами є:

Аналіз ринкової капіталізації. Даний підхід дає можливість оцінити загальну ринкову капіталізацію ринку криптовалют, що є ключовим показником його розміру та стабільності. Завдяки цьому методу можна оцінити загальну вартість всіх цифрових активів на ринку, а також порівняти різні криптовалюти між собою. Аналіз ринкової капіталізації дозволяє простежити зміни у вартості криптовалютного ринку та зробити висновки щодо його поточного стану та перспектив розвитку.

Аналіз цінових трендів. Даний метод спрямований на виявлення тенденцій зміни цін на криптовалюту. Використовуючи графічні дані, технічні індикатори та інші інструменти, можна визначити моменти змін у динаміці цін.

Це дозволяє трейдерам прогнозувати подальший розвиток ситуації на ринку та ухвалювати відповідні рішення щодо купівлі або продажу активів. Технічний аналіз є одним із найважливіших інструментів, що використовуються у криптотрейдингу.

Аналіз новин та інформації. Новини та різноманітні події можуть значною мірою впливати на курс криптовалют. Зміни у законодавстві, регулювання ринку, партнерства між великими компаніями, злам бірж або запуск нових технологій можуть кардинально змінити ринкову ситуацію. Для аналізу цього аспекту використовуються соціальні мережі, офіційні новинні ресурси та спеціалізовані аналітичні платформи. [4]

Криптовалютний ринок є динамічним і нестабільним, що вимагає від інвесторів і трейдерів постійної уваги та здатності швидко реагувати на зміни. Саме тому вивчення особливостей цього ринку є важливим завданням для всіх, хто працює з цифровими активами. Для проведення якісного аналізу необхідно враховувати кілька ключових аспектів, серед яких динаміка цін, рейтинги криптовалют, технічний аналіз, новини та події, що можуть вплинути на вартість активів.

Динаміка цін є одним із основних параметрів, що дозволяє визначити поточний стан криптовалютного ринку. Аналіз змін вартості активів допомагає виявити закономірності та тенденції, які можуть свідчити про можливі коливання цін у майбутньому. Дослідження динаміки вартості дозволяє вчасно виявляти як позитивні, так і негативні тренди, що є важливими для трейдерів та інвесторів.

Рейтинги криптовалют є ще одним важливим критерієм аналізу ринку цифрових активів. Вони допомагають визначити, які криптовалюти мають найбільші перспективи для зростання, а які можуть бути надмірно ризикованими для інвестицій. Оцінка рейтингів базується на таких показниках, як капіталізація, рівень ліквідності, рівень прийняття ринку та технологічна основа кожного проєкту. [11]

Технічний аналіз використовується для глибшого вивчення ринку та побудови прогнозів на основі історичних даних про рух цін. Для цього

застосовуються різноманітні технічні індикатори, патерни графіків та статистичні методи. Використання цього підходу дає можливість інвесторам і трейдерам визначати оптимальні точки входу та виходу з ринку, а також мінімізувати ризики фінансових втрат.

Новини та події відіграють ключову роль у визначенні ринкових трендів. Зміни у регуляторній політиці, заяви впливових осіб, впровадження нових технологій та партнерства між великими компаніями можуть суттєво впливати на криптовалютний ринок. Своєчасний аналіз інформаційного фону дозволяє інвесторам оперативно реагувати на зміни та ухвалювати зважені рішення.

Комплексний аналіз ринку криптовалют дозволяє отримати розширене уявлення про його поточний стан та тенденції розвитку. Це дає можливість ухвалювати більш обґрунтовані інвестиційні рішення, мінімізуючи ризики та збільшуючи потенційний прибуток.

Розробка мобільного додатку для відображення ринкових даних про криптовалюту є важливим етапом у створенні ефективного інструменту для трейдерів та інвесторів. Додаток повинен забезпечувати доступ до актуальних фінансових даних, аналізу ринку, новин та прогнозів, що дозволить користувачам своєчасно отримувати важливу інформацію. Інтеграція сучасних аналітичних інструментів у додаток допоможе трейдерам краще розуміти ринок та ухвалювати правильні рішення, ґрунтуючись на точних даних та об'єктивному аналізі. [23]

Таким чином, розгляд криптовалютного ринку та його аналіз є важливим фактором для ефективного інвестування. Урахування ключових показників ринку, використання надійних джерел інформації та правильне застосування методів аналізу дозволяють отримувати об'єктивну картину стану ринку та ухвалювати раціональні рішення. Створення мобільного додатку для відображення ринкових даних криптовалют сприятиме підвищенню рівня доступності інформації для користувачів, що працюють із цифровими активами..

Крім основних методів аналізу, існує низка додаткових факторів, які можуть відігравати важливу роль у формуванні динаміки криптовалютного

ринку. Врахування цих аспектів дозволяє отримати більш комплексне розуміння ринкових процесів і забезпечує більш точний прогноз змін у вартості цифрових активів.

Одним із ключових факторів є регулювання криптовалют у різних країнах. Політика держав щодо цифрових активів може суттєво впливати на їхню вартість і рівень популярності. Уряди деяких країн запроваджують законодавчі ініціативи, спрямовані на регулювання використання криптовалют, що може як підтримувати, так і стримувати їхній розвиток. Обмеження щодо операцій із криптовалютами, заборона їх використання в певних регіонах або навпаки — сприяння інтеграції блокчейн-технологій у фінансову систему країни — безпосередньо впливають на довіру до ринку та інвестиційну активність.

Технологічні аспекти також є важливими для оцінки стану криптовалютного ринку. Розвиток нових технологій, зокрема технології блокчейн, може змінювати принципи роботи криптовалют, підвищувати їхню ефективність і забезпечувати додаткову безпеку. Інновації, такі як удосконалення смарт-контрактів, перехід до енергоефективних алгоритмів консенсусу, створення нових механізмів масштабованості мережі — все це може впливати на попит на конкретні цифрові активи та, відповідно, на їхню вартість.

Ще одним важливим фактором є суспільні настрої щодо криптовалют. Громадська думка може змінюватися під впливом новин, інформації в соціальних мережах, заяв відомих підприємців та економічних експертів. Позитивне інформаційне тло може сприяти зростанню вартості криптовалют, тоді як негативні повідомлення, зокрема новини про шахрайські проєкти, зломи бірж або регуляторні обмеження, можуть спричинити паніку серед інвесторів і різке падіння цін. [7]

Врахування всіх вищезазначених чинників дає змогу отримати більш точне уявлення про стан ринку криптовалют та прогнозувати можливі сценарії його розвитку. Комплексний підхід до аналізу дозволяє приймати більш обґрунтовані інвестиційні рішення та мінімізувати потенційні ризики, що є

надзвичайно важливим для успішної роботи на цьому нестабільному ринку.

1.2. Огляд існуючих мобільних додатків для відстеження ринку криптовалют

Огляд мобільних додатків для відстеження ринку криптовалют Аналіз існуючих мобільних додатків для моніторингу криптовалютного ринку є важливим етапом перед розробкою власного програмного продукту. Дослідження ринку допомагає виявити основні функціональні можливості, сильні та слабкі сторони конкурентних додатків, а також визначити, які аспекти варто включити у власну розробку для підвищення її ефективності та привабливості для користувачів.[5]

У цьому розділі буде проведено огляд трьох популярних мобільних додатків, які використовуються для відстеження ринку криптовалют:

1. Blockfolio
2. CoinMarketCap
3. Delta

Для кожного з додатків буде розглянуто такі аспекти:

- функціональність та основні можливості
- дизайн та зручність використання
- якість і наочність графіків
- точність та доступність ринкових даних
- цінова політика та доступність безкоштовної версії

Blockfolio

Blockfolio є одним із найбільш популярних мобільних додатків, призначених для моніторингу криптовалютного ринку. Його основна функція полягає у відстеженні понад 10 000 різних криптовалют і токенів, а також наданні користувачам можливості створення персоналізованих портфелів. Користувачі можуть додавати транзакції вручну та отримувати сповіщення про зміну цін. Додаток також пропонує систему новин та інтеграцію з біржами для отримання ринкової інформації в режимі реального часу.

Переваги:

- зручний та інтуїтивно зрозумілий інтерфейс

- широкий набір функцій, включаючи створення портфелів та сповіщення
- безкоштовний доступ до більшості основних функцій

Недоліки:

- графіки можуть бути складними для читання
- відсутність вбудованих інструментів технічного аналізу
- немає можливості змінювати тему на темний режим

Blockfolio є гарним вибором для тих, хто шукає простий та ефективний додаток для базового моніторингу криптовалютного ринку. Однак, для користувачів, які потребують глибшого аналізу або розширених інструментів для прогнозування ринку, цей додаток може бути недостатньо функціональним.

CoinMarketCap

CoinMarketCap є одним із найбільш відомих сервісів для аналізу криптовалютного ринку, який також має власний мобільний додаток. Він пропонує доступ до інформації про понад 8 000 криптовалют і токенів, дозволяючи користувачам шукати монети за назвою, символом або ринковою капіталізацією. Додаток забезпечує можливість відстеження змін цін у реальному часі, перегляду історичних даних, а також ознайомлення з біржами, що торгують тими чи іншими криптовалютами.

Переваги:

- широкий спектр інформації про криптовалюти
- висока точність ринкових даних
- безкоштовний доступ до основних функцій

Недоліки:

- інтерфейс може бути перевантаженим великою кількістю інформації
- відсутній темний режим

CoinMarketCap підходить користувачам, які шукають комплексний інструмент для глибокого аналізу ринку та бажають отримувати максимальну кількість ринкових даних. Проте його складний інтерфейс може бути незручним для новачків, а відсутність темного режиму може впливати на комфорт використання додатку в умовах низького освітлення.[12]

Delta є ще одним популярним додатком для моніторингу криптовалют,

який дозволяє користувачам стежити за понад 6 000 різними цифровими активами. Окрім стандартного відстеження змін цін, додаток пропонує можливість створення портфелів, отримання сповіщень про зміни вартості та перегляду графіків у реальному часі. Він також має вбудовані інструменти технічного аналізу, що є важливою перевагою для трейдерів, які бажають аналізувати ринок більш глибоко.

Переваги:

- сучасний та привабливий інтерфейс
- наявність темного режиму
- вбудовані інструменти технічного аналізу
- можливість вибору між безкоштовною та платною версіями

Недоліки:

- безкоштовна версія має обмежений функціонал
- відсутні інструменти для автоматичного ребалансування портфеля

Delta є хорошим вибором для користувачів, які шукають стильний, функціональний додаток із сучасним дизайном та можливістю технічного аналізу. Безкоштовна версія може мати певні обмеження, але платна підписка відкриває доступ до розширених функцій, що робить його привабливим для досвідчених трейдерів.[2]

Кожен із розглянутих додатків має свої унікальні особливості, які можуть бути корисними залежно від потреб користувачів.

Blockfolio є чудовим вибором для тих, хто шукає простий і зручний додаток із базовими функціями моніторингу криптовалют.

CoinMarketCap підходить для користувачів, яким потрібен комплексний інструмент із максимальною кількістю ринкових даних та можливістю порівняння різних цифрових активів.

Delta пропонує сучасний дизайн, темний режим та вбудовані інструменти технічного аналізу, що робить його оптимальним варіантом для активних трейдерів.

Вибір найкращого мобільного додатку для моніторингу криптовалютного ринку залежить від індивідуальних потреб користувачів. Ті, хто шукають

простоту та зручність, можуть віддати перевагу Blockfolio. Для тих, хто бажає отримувати найбільш повну інформацію, кращим вибором стане CoinMarketCap. Якщо ж необхідні розширені аналітичні можливості та сучасний дизайн, варто звернути увагу на Delta. Кожен із цих додатків має свої особливості, і оптимальний вибір залежить від конкретних запитів користувача..

1.3. Опис функціоналу та вимог до мобільного додатку

Вимоги та особливості розробки мобільного додатку для відображення даних криптовалютного ринку. На основі проведеного аналізу існуючих мобільних додатків та сучасних тенденцій дизайну і функціональності, необхідно визначити ключові вимоги до розробки власного мобільного додатку. Основна мета проєкту – створення інтуїтивно зрозумілого, функціонального та сучасного додатку, що забезпечить користувачів актуальною інформацією про криптовалютний ринок та надасть інструменти для аналітики і моніторингу.

Основні функціональні можливості

Щоб мобільний додаток відповідав потребам користувачів, він має включати наступні функції:

- Відображення детальної інформації про криптовалюти, включаючи історію цін, відкриті та закриті ордери, капіталізацію ринку та інші фінансові показники.[21]

- Можливість моніторингу курсів криптовалют у різних валютних парах, що дозволить трейдерам та інвесторам оцінювати динаміку змін вартості активів у різних фіатних валютах та криптовалютах.

- Наявність графіків, що відображають рух курсів криптовалют за різні часові періоди та містять інструменти для технічного аналізу.

- Відстеження портфеля криптовалют, що включає можливість додавати активи, переглядати зміну їхньої вартості в режимі реального часу, а також отримувати статистичні дані про прибутковість інвестицій.

- Сповіщення про зміни цін, важливі новини та події, що можуть вплинути на криптовалютний ринок.

- Інтеграція з соціальними мережами, яка дозволить користувачам ділитися своїми фінансовими показниками, аналітичними графіками та іншою інформацією. [7]

Дизайн та користувацький інтерфейс

Для забезпечення зручності використання та естетичної привабливості додатку, його інтерфейс повинен відповідати сучасним стандартам UI/UX-дизайну. Основні вимоги до дизайну:

- Простий та зрозумілий інтерфейс, що дозволяє користувачам швидко знаходити необхідну інформацію та виконувати основні операції без зайвих труднощів.

- Гнучка система налаштувань, яка дає змогу користувачам адаптувати інтерфейс додатку під власні вподобання.

- Підтримка темного режиму, що забезпечує комфорт при використанні додатку в умовах низького освітлення та допомагає зменшити навантаження на очі.

- Адаптивний дизайн, що дозволяє коректне відображення додатку на різних мобільних пристроях із різними розмірами екранів.[16]

Візуалізація даних

Щоб користувачі могли ефективно аналізувати ринок, додаток повинен включати якісну візуалізацію фінансових даних.

- Використання різних типів графіків для аналізу динаміки цін криптовалют за різні часові періоди (лінійні графіки, свічкові графіки, гістограми тощо).

- Висока якість графіків, що дозволить користувачам чітко аналізувати коливання вартості активів та виконувати технічний аналіз.

- Можливість інтерактивного аналізу даних з використанням масштабування та вибору окремих часових інтервалів.

Сучасні технології дизайну та UI/UX

При створенні інтерфейсу важливо дотримуватися найновіших тенденцій у дизайні мобільних додатків. Основні концепції, які необхідно враховувати:

- Використання принципів Material Design – стандарту, розробленого

Google, що дозволяє створювати інтуїтивно зрозумілі інтерфейси з чіткою візуальною ієрархією.

- Використання Motion-дизайну для забезпечення плавних анімацій та інтерактивних елементів, які покращують користувацький досвід.

- Підтримка мікровзаємодій – невеликих інтерактивних ефектів, що надають додатку індивідуальності та допомагають користувачеві швидше адаптуватися до інтерфейсу.

- Інтеграція елементів Fluent Design, що дозволяє створювати елегантний та сучасний інтерфейс із використанням напівпрозорих ефектів, м’яких тіней та динамічного освітлення.

Додаткові можливості

Окрім базового функціоналу, мобільний додаток повинен мати кілька додаткових можливостей, що підвищують його корисність та конкурентоспроможність:

- Оновлення цін у реальному часі, що дозволить користувачам стежити за змінами вартості криптовалют без необхідності постійного оновлення сторінки.

- Можливість сортування криптовалют за різними параметрами, такими як ринкова капіталізація, обсяг торгів, відсоткові зміни за певний період часу тощо.

- Вбудований модуль новин, що агрегує важливі події та аналітичні статті з надійних джерел.

- Функція «улюблених активів», що дозволяє користувачам швидко отримувати доступ до даних про вибрані криптовалюти.

Розробка мобільного додатку для відображення даних криптовалютного ринку вимагає врахування багатьох факторів, включаючи функціональність, дизайн, швидкість роботи та інтеграцію сучасних технологій. Додаток повинен забезпечувати користувачам доступ до актуальної ринкової інформації, пропонувати зручний інтерфейс та інструменти для аналізу ринку, а також відповідати сучасним стандартам мобільної розробки.[19]

Використання таких підходів, як Material Design, Motion-дизайн, темний режим та інтеграція анімацій, дозволить створити сучасний, ефективний і

зручний у використанні додаток. Наявність розширених функцій, таких як управління криптовалютним портфелем, сповіщення про зміни ринку, вбудовані графіки та підтримка інтеграції з соціальними мережами, зроблять додаток корисним інструментом для трейдерів та інвесторів.

Таким чином, реалізація зазначених вимог дозволить створити якісний продукт, що відповідатиме сучасним потребам користувачів і стане надійним помічником для всіх, хто працює з криптовалютним ринком.

1.4. Аналіз методів розробки додатків для Android

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

Розробка мобільних додатків для Android може здійснюватися за допомогою різних мов програмування, кожна з яких має власні переваги та особливості. Найбільш поширеними серед них є C#, Java та Python. Кожна з цих мов використовується у певних випадках, залежно від вимог проєкту, рівня продуктивності, зручності розробки та наявних інструментів.[22]

Використання C# для розробки Android-додатків

C# є скомпільованою мовою програмування, що використовується для створення програмного забезпечення різного типу, включаючи мобільні додатки, веб-додатки та десктопні програми. Завдяки підтримці великою розробницькою спільнотою та розвиненій екосистемі бібліотек, C# є популярним вибором серед розробників.

Основні переваги C# у розробці мобільних додатків для Android:

- Висока швидкість виконання програм завдяки попередній компіляції коду в машинний код, що дозволяє отримати швидкодію вищу, ніж у мов з інтерпретованим виконанням.

- Простота у використанні та відносно легке навчання завдяки зрозумілому синтаксису та великій кількості навчальних матеріалів.

- Велика кількість інструментів та бібліотек, що значно спрощують процес розробки.

Завдяки платформі Xamarin, C# дозволяє створювати кросплатформні

мобільні додатки, які можуть працювати не лише на Android, а й на iOS.

Використання Java для розробки Android-додатків

Java є однією з основних мов програмування для створення Android-додатків і використовується в офіційному середовищі розробки Android Studio. Це об'єктно-орієнтована мова, яка має довгу історію розвитку та велику підтримку з боку спільноти.

Основні переваги Java у розробці мобільних додатків для Android:

- Портативність: код, написаний на Java, може працювати на будь-якому пристрої, що підтримує віртуальну машину Java (JVM), що робить його зручним для створення кросплатформних додатків.

- Високий рівень безпеки: Java має вбудовані механізми безпеки, які запобігають виконанню небезпечного коду та захищають додатки від атак.

- Об'єктно-орієнтований підхід, що спрощує організацію коду та його повторне використання.

Завдяки своїй стабільності та широкому використанню в мобільній розробці, Java залишається одним із провідних виборів для створення додатків на Android.

Використання Python для розробки Android-додатків

Python є мовою програмування з інтерпретованим виконанням, яка використовується для створення веб-додатків, мобільних програм, десктопних програм та інших рішень. Він відомий своєю простотою у використанні та широким набором інструментів для аналізу даних та штучного інтелекту. [1]

Основні переваги Python для розробки Android-додатків:

- Простота синтаксису, що дозволяє швидко опановувати мову та зменшує час розробки.

- Велика кількість бібліотек, які підтримують різні завдання, від обробки даних до розробки мобільних додатків.

- Гнучкість використання та можливість інтеграції з іншими мовами програмування.

Попри те, що Python не є основною мовою для розробки Android-додатків, його можна використовувати разом із такими інструментами, як Kivy

або BeeWare, для створення мобільних додатків. Однак продуктивність додатків на Python поступається рішенням, написаним на C# чи Java, через необхідність інтерпретації коду під час виконання. [30]

Інструменти для розробки Android-додатків

Щоб створювати якісні мобільні додатки, необхідно використовувати відповідні середовища розробки та фреймворки. Деякі з найбільш популярних інструментів для роботи з C#, Java та Python включають:

- Visual Studio (з підтримкою Xamarin або Unity) – це інтегроване середовище розробки (IDE), яке дозволяє створювати мобільні додатки на C#. Xamarin підтримує кросплатформену розробку, що дозволяє одночасно створювати додатки для Android та iOS.

- Android Studio – офіційне середовище розробки для Android-додатків, яке підтримує Java та Kotlin. Воно містить великий набір інструментів для тестування та оптимізації додатків.

- Eclipse – ще одне IDE, яке підтримує C#, Java та Python, що дає можливість створювати мобільні додатки з використанням різних мов програмування.

Чому C# та Xamarin є гарним вибором для розробки додатків для Android

C# разом із платформою Xamarin є привабливим вибором для розробників, які прагнуть створювати кросплатформні додатки з високою продуктивністю. Деякі з ключових переваг:

- Висока швидкість роботи програм завдяки компіляції в машинний код.

- Простота вивчення та велика кількість навчальних матеріалів.

- Велика спільнота розробників та доступність готових бібліотек.

Окрім цього, C# та Xamarin надають ряд додаткових можливостей, що роблять їх потужним рішенням для мобільної розробки:

- Xamarin.Forms – фреймворк для створення кросплатформних додатків із єдиним кодовим базисом для Android та iOS.

- Xamarin.Android – набір інструментів для створення нативних додатків на Android за допомогою C#.

- Xamarin.iOS – аналогічний набір інструментів для iOS.

Вибір мови програмування для розробки Android-додатків залежить від низки факторів, включаючи продуктивність, гнучкість, легкість розробки та підтримку спільноти.

- C# та Xamarin – чудовий вибір для кросплатформної розробки завдяки швидкості, зручності у використанні та великій кількості бібліотек.

- Java – ідеальне рішення для нативної розробки Android-додатків, оскільки ця мова підтримується безпосередньо Google.

- Python – більше підходить для розробки експериментальних мобільних додатків або у разі використання інструментів на базі штучного інтелекту. [9]

Залежно від цілей розробки, вибір між цими мовами може значно вплинути на кінцеву продуктивність, функціональність та можливості масштабування мобільного додатку.

РОЗДІЛ 2. ПЛАТФОРМИ РОЗРОБКИ .NET ТА XAMARIN

2.1. Засоби програмування для мобільної розробки

C# є сучасною, потужною та універсальною мовою програмування, яка використовується для розробки різних типів програмного забезпечення, зокрема мобільних застосунків, веб-додатків, десктопних програм та ігрових продуктів. Завдяки своїй продуктивності, широкій підтримці з боку спільноти та багатій екосистемі бібліотек і фреймворків, C# є одним із найбільш популярних інструментів серед розробників.

Оскільки C# є компільованою мовою програмування, вихідний код перед виконанням перетворюється у машинний код, що суттєво покращує продуктивність та ефективність роботи застосунків. Завдяки цьому програми, написані на C#, працюють швидше у порівнянні з тими, що використовують інтерпретовані мови, наприклад JavaScript. Це робить C# ідеальним вибором для розробки високопродуктивних додатків, де важливою є швидкість виконання та оптимізація ресурсів. [11]

Xamarin є кросплатформенною платформою розробки, яка дозволяє створювати мобільні додатки для операційних систем Android та iOS, використовуючи C#. Це означає, що розробник має змогу написати код один раз і використовувати його для кількох платформ, що значно зменшує витрати часу та ресурсів на розробку. Завдяки використанню спільної кодової бази, розробники можуть уникати дублювання коду, що підвищує продуктивність роботи та полегшує подальшу підтримку застосунку.

Ще однією вагомою перевагою Xamarin є можливість створення нативних інтерфейсів користувача. Це означає, що застосунки, розроблені за допомогою цієї технології, не лише виглядають, але й працюють так само, як і стандартні програми, створені безпосередньо для Android або iOS. Окрім того, Xamarin підтримує інтеграцію з нативними API платформ, що дозволяє розробникам отримати доступ до повного набору функціональності пристрою.

Visual Studio є потужним інтегрованим середовищем розробки (IDE), що широко використовується для створення різноманітних програмних продуктів.

Це інструмент, який забезпечує всі необхідні можливості для написання, налагодження, тестування та оптимізації коду. Visual Studio підтримує різні мови програмування, включаючи C#, Visual Basic та F#, а також має широкий набір розширень, які допомагають автоматизувати багато аспектів розробки.

Це середовище доступне для операційних систем Windows та macOS, що робить його гнучким вибором для розробників, незалежно від їхніх уподобань. Завдяки вбудованим засобам налагодження та підтримці інтеграції з системами контролю версій, Visual Studio дозволяє значно прискорити процес розробки, спрощуючи пошук та виправлення помилок.

Git є популярною системою контролю версій, яка широко використовується в розробці програмного забезпечення. Вона дозволяє ефективно відстежувати зміни у вихідному коді, співпрацювати з іншими розробниками та зберігати історію змін проєкту. Використання Git є стандартною практикою в сучасній розробці застосунків, оскільки ця система забезпечує надійний механізм керування змінами та дозволяє легко відкотити будь-які зміни у разі потреби. [6]

Завдяки інтеграції Git у більшість сучасних середовищ розробки, таких як Visual Studio, Android Studio та інші, розробники можуть працювати в команді, зручно керуючи версіями програмного коду, що є критично важливим у складних проєктах.

JSON є зручним і легким форматом обміну даними, який широко використовується у мобільних додатках для збереження та передачі інформації між клієнтськими та серверними частинами програми. Його основною перевагою є простота синтаксису та легкість у читанні, що робить його ідеальним для інтеграції з веб-сервісами та базами даних.

JSON активно використовується у розробці мобільних застосунків для Android, оскільки він дозволяє ефективно передавати дані між сервером та клієнтом у структурованому вигляді. Завдяки підтримці цього формату у багатьох мовах програмування, JSON став одним із найпоширеніших способів збереження та передачі інформації.

SQL є стандартною мовою для роботи з базами даних, яка

використовується для створення, зміни, читання та видалення даних. У розробці мобільних застосунків для Android SQL є важливим інструментом, що дозволяє ефективно керувати інформацією, збереженою у локальних або віддалених базах даних. [18]

Завдяки потужності та гнучкості SQL, розробники можуть оптимізувати зберігання та обробку великих обсягів даних, забезпечуючи швидку взаємодію користувачів із застосунком. Для Android найчастіше використовується SQLite – легковагова вбудована база даних, яка дозволяє зберігати та обробляти інформацію без потреби у зовнішньому сервері.

MVVM є патерном проєктування програмного забезпечення, що використовується для розділення логіки додатку від його інтерфейсу користувача. Він є важливим компонентом сучасної мобільної розробки, оскільки сприяє покращенню структури коду та підвищенню тестованості застосунку.

Завдяки MVVM розробники можуть організувати код таким чином, що зміни в інтерфейсі користувача не впливатимуть на основну бізнес-логіку програми, що полегшує підтримку, розширення та тестування застосунку.

XAML є мовою розмітки, що використовується для створення користувацьких інтерфейсів у застосунках, розроблених за допомогою Xamarin. Вона дозволяє визначати компоненти інтерфейсу у декларативному вигляді, що спрощує їхню розробку та зміну. [3]

Однією з основних переваг використання XAML є можливість чіткого відокремлення логіки застосунку від його візуальної частини, що дозволяє дизайнерам і розробникам працювати над проєктом незалежно один від одного.

Завдяки поєднанню технологій C#, Xamarin, SQL, JSON та XAML, мобільні застосунки для Android можуть бути ефективними, продуктивними та гнучкими. Розробники отримують можливість створювати додатки з високим рівнем продуктивності, нативною взаємодією з платформою та простотою у підтримці. Поєднання цих технологій робить їх ідеальним вибором для реалізації сучасних мобільних рішень.[22]

C# є об’єктно-орієнтованою мовою програмування загального

призначення, яку було створено компанією Microsoft. Вона є компільованою, що означає, що вихідний код перед виконанням перетворюється на машинний код, завдяки чому забезпечується висока швидкодія та ефективність роботи програм. Це відрізняє C# від інтерпретованих мов, таких як JavaScript, які виконуються повільніше через необхідність обробки коду під час запуску.

Однією з ключових особливостей C# є її сувора типізація, що вимагає явного оголошення типів змінних і виразів. Це дозволяє уникнути багатьох поширених помилок у коді та підвищує його читабельність, спрощуючи подальший супровід і модифікацію програмного забезпечення.

Основні особливості мови C#:

- Об’єктно-орієнтований підхід: C# побудований на основі концепції об’єктно-орієнтованого програмування, що дозволяє ефективно структурувати код за допомогою класів і об’єктів. Це спрощує розробку, підтримку та масштабування програм.

- Компіляція коду: Оскільки C# є скомпільованою мовою, він забезпечує швидке виконання програм, що важливо для додатків, які вимагають високої продуктивності.

- Сильна типізація: Усі змінні та об’єкти в C# повинні мати чітко визначений тип, що запобігає помилкам, пов’язаним із некоректним використанням даних.

- Автоматичне управління пам’яттю: C# використовує механізм збору сміття (Garbage Collection), який автоматично вивільняє пам’ять, що більше не використовується. Це знижує ризик витоків пам’яті та полегшує управління ресурсами програми.

- Сумісність з іншими мовами: C# дозволяє взаємодіяти з кодом, написаним на інших мовах, таких як Visual Basic або Java. Це забезпечує ширші можливості інтеграції у великі програмні системи. [31]

C# є багатофункціональною мовою, що використовується для розробки широкого спектра програмного забезпечення, включаючи:

- мобільні додатки

- веб-додатки

- десктопні програми

- ігрові проєкти

- серверні сервіси

Щоб полегшити розробку програмного забезпечення на C#, існує безліч фреймворків і бібліотек, які значно розширюють можливості мови та прискорюють процес створення застосунків. [25]

Популярні фреймворки та бібліотеки для C#:

- .NET Framework – це потужний набір інструментів і бібліотек, що надає можливості для розробки додатків різних типів, включаючи веб-застосунки, десктопні програми та серверні рішення.

- Xamarin – кросплатформенна платформа, що дозволяє створювати мобільні додатки для Android та iOS, використовуючи єдину кодову базу на C#. Це значно зменшує витрати часу на розробку та спрощує підтримку застосунків.

- Unity – популярний ігровий рушій, який широко застосовується для створення 2D та 3D ігор для різних платформ, таких як комп'ютери, мобільні пристрої та ігрові консолі. Unity використовує C# для написання скриптів, що дозволяє розробникам створювати складні ігрові механіки та інтерактивний контент.

- ASP.NET – фреймворк для створення веб-додатків та сервісів, що дозволяє розробникам працювати з веб-технологіями на базі C#. Він підтримує інтеграцію з базами даних, має вбудовані засоби безпеки та забезпечує високу продуктивність веб-додатків.

- Entity Framework – ORM-фреймворк для роботи з базами даних, який дозволяє легко взаємодіяти з джерелами даних, такими як SQL Server, PostgreSQL та інші. Він забезпечує зручну абстракцію для маніпулювання базами даних, спрощуючи роботу з ними та підвищуючи продуктивність розробки.

Крім цих інструментів, для C# існує багато інших бібліотек, що допомагають розширити можливості мови, підвищити продуктивність програм та забезпечити зручну інтеграцію з іншими технологіями. [31]

Завдяки своїй потужності, продуктивності та широкому вибору інструментів C# є одним із найкращих варіантів для створення сучасного програмного забезпечення. Він надає розробникам можливість працювати з різними платформами, що робить його універсальним рішенням для багатьох сфер програмування.

NET Framework є програмним фреймворком, створеним компанією Microsoft, який призначений для розробки та виконання різноманітних додатків, що працюють переважно на операційній системі Windows. Він містить велику бібліотеку стандартних класів і забезпечує мовну сумісність завдяки Common Language Infrastructure (CLI), що дозволяє програмістам використовувати різні мови програмування в межах однієї платформи. [7]

Додатки, розроблені для .NET Framework, виконуються у спеціальному середовищі під назвою Common Language Runtime (CLR), яке є віртуальною машиною для керованих програм. CLR забезпечує такі критично важливі функції, як автоматичне управління пам’яттю, контроль безпеки, обробка винятків і оптимізація продуктивності. Завдяки цьому розробники можуть зосередитися на створенні логіки застосунку, не турбуючись про низькорівневе управління ресурсами.

.NET Framework складається з двох основних компонентів:

- Common Language Runtime (CLR) – середовище виконання, яке відповідає за управління кодом під час його виконання, забезпечуючи захист даних, ефективний розподіл пам’яті та механізм обробки помилок.

- Бібліотека класів .NET Framework (FCL) – велика колекція готових до використання класів і методів, що дозволяють розробникам створювати функціональні додатки без необхідності розробки багатьох базових функцій з нуля. [6]

Ця платформа відзначається високою гнучкістю та універсальністю, що робить її популярним вибором серед розробників для створення різних типів програмного забезпечення.

Основні особливості .NET Framework:

- Кросплатформність – фреймворк підтримує можливість розробки

застосунків для різних операційних систем, включаючи Windows, macOS та Linux. Хоча початково він був орієнтований виключно на Windows, новіші версії, такі як .NET Core і .NET 5+, значно розширили його сумісність.

- Повторне використання коду – бібліотека класів містить широкий набір багаторазових компонентів, що дозволяє значно прискорити процес розробки та зменшити дублювання коду.

- Простота у використанні – фреймворк побудований на основі об'єктно-орієнтованої моделі програмування, що робить код структурованим, легким для розуміння та супроводу.

- Безпека – платформа включає вбудовані механізми безпеки, такі як контроль доступу, шифрування та захист від несанкціонованого виконання коду, що допомагає створювати надійні та безпечні додатки.

- Висока продуктивність – .NET Framework оптимізований для швидкого виконання програм, використовуючи різні механізми, такі як Just-In-Time (JIT) компіляція та автоматичне управління ресурсами. [22]

Завдяки своїй універсальності, .NET Framework активно використовується у різних сферах розробки, зокрема:

- Веб-розробка – ASP.NET дозволяє створювати високопродуктивні та безпечні веб-додатки, які можуть взаємодіяти з базами даних, веб-сервісами та API.

- Десктопні додатки – Windows Forms і WPF (Windows Presentation Foundation) забезпечують можливості для створення графічних інтерфейсів для настільних застосунків.

- Мобільна розробка – за допомогою Xamarin можна створювати мобільні додатки для Android та iOS, використовуючи єдину кодову базу на C#.

- Ігрова індустрія – рушій Unity використовує .NET та C# для створення відеоігор та інтерактивних програм.

- Розробка серверних додатків – .NET часто застосовується для створення серверних рішень, які можуть взаємодіяти з великими масивами даних та забезпечувати високу продуктивність. [19]

Окрім базових можливостей, фреймворк підтримує інтеграцію з багатьма

іншими технологіями та бібліотеками, що значно розширює його функціональність.

Середовище .NET Framework залишається одним із найпопулярніших інструментів для створення програмного забезпечення завдяки своїй продуктивності, простоті розробки та широкому набору готових бібліотек. Його можливості дозволяють створювати як прості, так і складні рішення, що відповідають сучасним вимогам програмування.

Xamarin є потужною кросплатформенною платформою, яка використовується для розробки мобільних додатків, що можуть працювати на Android, iOS та macOS. Вона дозволяє розробникам створювати нативні застосунки, використовуючи мову програмування C#, що значно полегшує процес розробки та скорочує витрати часу завдяки можливості використовувати спільну кодову базу. [32]

Ця технологія складається з набору інструментів та бібліотек, які дають змогу писати код, що може бути скомпільований та виконаний на різних операційних системах. Такий підхід дозволяє мінімізувати дублювання коду, що не лише прискорює розробку, але й полегшує підтримку застосунку в майбутньому.

Xamarin тісно інтегрується з Visual Studio , що надає розробникам зручне середовище для написання, тестування та розгортання додатків. Використання Visual Studio у поєднанні з Xamarin значно покращує ефективність роботи завдяки таким можливостям:

- IntelliSense – інтелектуальна система підказок, що допомагає швидше писати код, автоматично доповнюючи синтаксис та пропонуючи варіанти для завершення команд.
- Редактор коду – потужний та зручний інструмент, що полегшує написання програмного коду, а також дозволяє здійснювати його швидке редагування.
- Вбудований налагоджувач – забезпечує ефективний пошук і виправлення помилок, дозволяючи розробникам виявляти та усувати недоліки безпосередньо під час виконання програми.

- Модульне тестування – вбудований тестовий фреймворк дає можливість перевіряти окремі модулі програми, що допомагає запобігти помилкам та покращити стабільність застосунку.

- Розгортання додатків – Visual Studio містить вбудований інструмент для швидкого розгортання Xamarin-додатків у магазини застосунків, такі як App Store та Google Play.

Крім інтеграції з Visual Studio, Xamarin надає ще низку можливостей, які роблять його популярним вибором для створення мобільних застосунків:

- Висока продуктивність – додатки, створені за допомогою Xamarin, компілюються в нативний код для кожної окремої платформи. Це забезпечує високу швидкість роботи та продуктивність, що відповідає рівню додатків, розроблених нативними засобами.

- Єдина кодова база – значна частина програмного коду є спільною для всіх платформ, що дозволяє уникнути дублювання роботи, зменшуючи витрати на розробку та технічну підтримку.

- Кросплатформність – можливість одночасної розробки для Android, iOS та macOS без необхідності писати окремий код для кожної операційної системи.

- Глибока інтеграція з платформами – Xamarin надає доступ до нативних API кожної платформи, що дозволяє використовувати всі можливості пристроїв, такі як GPS, камера, сенсори тощо.

- Широка підтримка спільноти – велика кількість розробників по всьому світу працює з Xamarin, що сприяє швидкому пошуку рішень та розширенню документації.

Окремо варто згадати про Xamarin Android Player – вбудований емулятор, який дозволяє тестувати мобільні додатки без потреби використовувати фізичний пристрій. Цей інструмент значно спрощує процес розробки та налагодження, оскільки розробники можуть запускати свої програми безпосередньо на комп’ютері, перевіряючи їхню роботу в реальних умовах. [27]

Завдяки всім перерахованим особливостям Xamarin є чудовим вибором

для створення сучасних мобільних застосунків, що працюють на різних платформах, забезпечуючи високу продуктивність, спрощену розробку та ефективне тестування..

Visual Studio є потужним інтегрованим середовищем розробки (IDE), створеним компанією Microsoft, яке використовується для створення різних видів програмного забезпечення, включаючи веб-додатки, настільні програми, веб-сайти, мобільні застосунки та серверні рішення. Це багатофункціональний інструмент, який забезпечує зручне написання, тестування, компіляцію та розгортання програм, значно спрощуючи роботу розробників.

Visual Studio надає широкий набір можливостей, що роблять процес розробки швидшим, зручнішим та ефективнішим. Його основні функції включають:

- Редактор коду з вбудованою підтримкою IntelliSense – інтелектуальної системи автодоповнення коду, яка підвищує швидкість написання програм і зменшує кількість синтаксичних помилок. Також редактор містить механізми підсвічування синтаксису, що полегшує сприйняття коду.

- Налагоджувач коду з можливістю встановлення точок зупинки, моніторингу змінних у реальному часі та покрокового виконання програми для пошуку та усунення помилок.

- Провідник проєктів і рішень , що забезпечує зручне керування файлами та структурами проєкту, дозволяючи організувати розробку навіть у великих командах.

- Система збірки , яка автоматично компілює та компонуватиме код, забезпечуючи швидке створення робочих версій програм.

- Фреймворк для тестування , який дозволяє створювати, запускати та аналізувати модульні тести, що покращує якість програмного забезпечення та зменшує ризик помилок у фінальному продукті.

- Інструменти для розгортання додатків , що дозволяють швидко публікувати проєкти в різних середовищах, таких як локальні сервери, хмарні сервіси або магазини застосунків.

Visual Studio доступний як для операційної системи Windows, так і для

macOS, що забезпечує гнучкість у виборі платформи для розробки. Це комерційне програмне забезпечення, однак існує безкоштовна версія – Visual Studio Community , яка доступна для студентів, індивідуальних розробників та проєктів з відкритим вихідним кодом. [23]

Основні переваги Visual Studio:

- Висока продуктивність – середовище розробки оснащене розширеним набором функцій, які допомагають розробникам швидше писати код, знаходити помилки та тестувати свої програми. Завдяки IntelliSense та автоматичному завершенню коду зменшується кількість рутинних операцій, що підвищує ефективність роботи.

- Ефективне тестування та налагодження – вбудовані інструменти дозволяють виконувати глибоку перевірку коду, що значно зменшує кількість критичних помилок у фінальному продукті.

- Зручна спільна робота – Visual Studio підтримує інтеграцію з Git та іншими системами контролю версій, що дозволяє командам ефективно працювати над одним проєктом, відстежуючи зміни та зберігаючи історію редагувань.

- Гнучке масштабування – IDE може використовуватися як для малих проєктів, так і для розробки складних корпоративних рішень. Visual Studio забезпечує підтримку великих команд і дозволяє легко керувати складними програмними архітектурами.

- Інтеграція з хмарними сервісами – завдяки підтримці Microsoft Azure розробники можуть створювати та розгортати хмарні застосунки безпосередньо з Visual Studio, що розширює можливості програмного забезпечення.

Visual Studio підтримує широкий спектр мов програмування, серед яких C#, JavaScript, Python, F#, Visual Basic та багато інших . Це робить його універсальним середовищем розробки, що підходить для різних сфер, включаючи веб-програмування, мобільну розробку, створення ігор, серверні рішення та роботу з базами даних. [23]

Завдяки своєму широкому функціоналу, зручності використання та можливостям інтеграції Visual Studio є одним із найбільш популярних

інструментів серед розробників у всьому світі. Він дозволяє створювати високоякісне програмне забезпечення з мінімальними витратами часу, забезпечуючи стабільність, продуктивність та зручність роботи як для окремих програмістів, так і для великих команд.

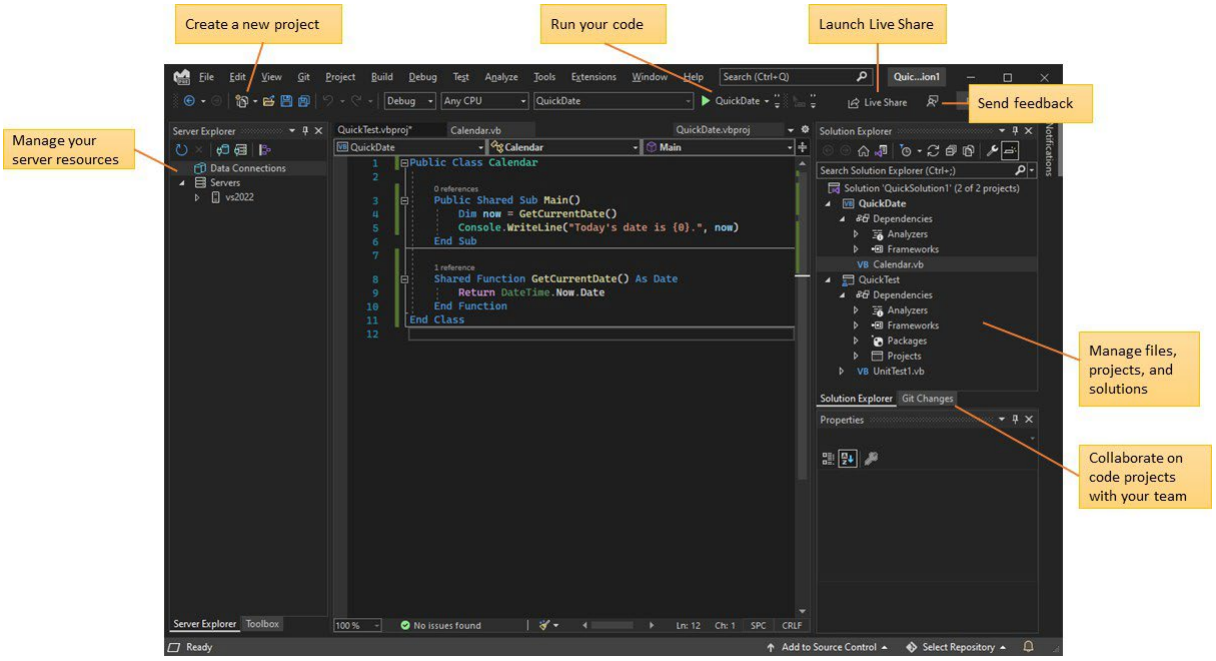

Рис. 2.1. Інтерфейс Visual Studio

Git – це розподілена система контролю версій, яка забезпечує ефективне відстеження змін у файлах та спрощує координацію роботи розробників програмного забезпечення. Вона широко використовується в розробці програм, особливо у великих командах, де потрібна синхронізація роботи між багатьма учасниками. Основна мета Git – забезпечити високу швидкість роботи, збереження цілісності даних і підтримку розподілених, нелінійних робочих процесів, що дозволяє гнучко працювати над програмним кодом. [28]

Розробка Git була розпочата Лінусом Торвальдсом у 2005 році для потреб розробки ядра операційної системи Linux. У розробці брали участь й інші члени спільноти ядра Linux, що допомогло створити потужний інструмент, який сьогодні використовується в тисячах програмних проєктів по всьому світу. Починаючи з 2005 року, підтримкою та розвитком Git займається Junio C

Hamano. [16]

Git є безкоштовним програмним забезпеченням з відкритим вихідним кодом, що розповсюджується на умовах Стандартної публічної ліцензії GNU (GNU General Public License). Це означає, що будь-хто може використовувати Git без фінансових витрат, а також модифікувати його відповідно до власних потреб.

Однією з найважливіших особливостей Git є його доступність на різних операційних системах. Git може працювати на всіх основних платформах, включаючи Linux, macOS, Windows і навіть Solaris , що робить його зручним для будь-якого розробника, незалежно від вибору операційної системи. [19]

Основні переваги використання Git:

- Ефективний контроль версій

Git дозволяє вести історію змін у коді, що дає змогу розробникам переглядати, які зміни були внесені, ким і коли. Це значно полегшує процес налагодження, дозволяє порівнювати різні версії файлів та, за потреби, повертатися до попередніх версій коду.

- Зручна командна робота

Завдяки розподіленій архітектурі Git розробники можуть одночасно працювати над одним проєктом, не заважаючи одне одному. Кожен учасник має власну локальну копію репозиторію, що дозволяє працювати автономно, а потім синхронізувати зміни з головною гілкою, об'єднуючи оновлення без конфліктів.

- Гнучке розгалуження (branching)

Git дозволяє легко створювати нові гілки коду для тестування нових функцій або виправлення помилок, не впливаючи на основний код проєкту. Це дає можливість експериментувати з різними підходами до реалізації функціоналу без ризику порушити роботу основної версії застосунку.

- Підтримка розподіленої роботи

На відміну від централізованих систем контролю версій, Git не потребує постійного підключення до сервера. Кожен розробник має повну копію репозиторію, що дозволяє працювати офлайн, а потім синхронізувати зміни,

коли це необхідно.

- Наявність віддалених репозиторіїв

Репозиторії Git можуть зберігатися не лише локально, а й у віддалених хостингових сервісах, таких як GitHub, GitLab, Bitbucket , що дає змогу легко співпрацювати над проєктами, публікувати зміни та отримувати внески від інших розробників.

- Безпека та надійність

Git забезпечує високу цілісність даних завдяки криптографічному хешуванню SHA-1, яке гарантує, що жодні зміни в історії комітів не будуть випадково або навмисно змінені без виявлення.

- Кросплатформність та портативність

Завдяки підтримці всіх основних операційних систем Git може використовуватися розробниками незалежно від їхнього робочого середовища. Це забезпечує максимальну гнучкість у виборі інструментів для роботи над проєктами.

- Безкоштовний доступ та відкритий вихідний код

Оскільки Git є вільним програмним забезпеченням, його можуть використовувати як незалежні розробники, так і великі компанії без жодних фінансових витрат. Більш того, можливість змінювати та адаптувати Git відповідно до потреб конкретного проєкту робить його ще привабливішим для професійного використання.

Додаткові можливості Git:

- Розширена система злиття змін

Git має потужний механізм об'єднання коду (merge), що дозволяє легко поєднувати зміни з різних гілок без ризику втрати даних.

- Інтеграція з популярними середовищами розробки

Git підтримується у більшості сучасних IDE, таких як Visual Studio, IntelliJ IDEA, Eclipse, PyCharm , що робить його зручним для інтеграції у робочий процес розробників.

- Автоматизація процесів за допомогою Git Hooks

Git дозволяє налаштовувати автоматичне виконання скриптів перед або

після певних операцій (наприклад, перед комітом або перед злиттям змін), що спрощує контроль якості коду. [19]

Завдяки своїй швидкості, надійності та підтримці розподілених робочих процесів Git став стандартним інструментом у сучасній розробці програмного забезпечення. Незалежно від розміру команди та масштабу проєкту, Git дозволяє ефективно організувати роботу над вихідним кодом, забезпечуючи простоту спільної розробки, безпеку змін та гнучкість у керуванні версіями.

2.2. Засоби взаємодії з даними у мобільних додатках

JSON (JavaScript Object Notation) є простим і зручним форматом обміну даними, який набув широкого поширення в сучасному програмуванні. Він використовується для передачі, зберігання та обробки структурованої інформації між різними програмами та сервісами. Основна перевага JSON – це його текстова природа, що робить дані легкими для читання як людиною, так і машиною. Завдяки своїй універсальності він може бути використаний у поєднанні з будь-якою мовою програмування, що підтримує синтаксичний аналіз та обробку JSON. [31]

Хоча JSON спочатку був розроблений на основі JavaScript, він не є прив’язаним до цієї мови й активно використовується у багатьох інших мовах програмування, включаючи Python, Java, C#, PHP, Ruby та багато інших. Це робить його ефективним інструментом для обміну даними між різними програмними платформами та системами.

Основною структурною одиницею JSON є об'єкт , який представлений у вигляді набору пар ключ-значення . Об'єкти JSON беруться у фігурні дужки `{}`, а кожен ключ записується у подвійних лапках " " , після чого ставиться двокрапка, за якою йде значення. Значення можуть бути представлені різними типами даних:

- Рядки (текстові значення)
- Числа (цілі та з плаваючою комою)
- Булеві значення (true або false)

- null (відсутність значення)
- Масиви (списки значень, укладені в квадратні дужки `[]`)
- Об'єкти (вкладені структури у фігурних дужках `{}`)

Використання JSON у веб-розробці

JSON є стандартним форматом обміну даними у веб-додатках, особливо у взаємодії між клієнтом і сервером. Його часто використовують у REST API та GraphQL API , де він слугує проміжним форматом передачі даних між веб-додатком і серверною частиною. JSON дозволяє швидко передавати дані між браузером і сервером у сучасних веб-додатках, що базуються на технологіях AJAX або Fetch API.

Приклад використання JSON у запитах до API:

1. Клієнтський застосунок (наприклад, веб-додаток) надсилає запит на сервер
2. Сервер обробляє запит та повертає JSON-об'єкт із відповіддю
3. Клієнт отримує JSON-дані та використовує їх у своїй логіці (наприклад, виводить на екран, зберігає у змінних, модифікує)

Також JSON застосовується для роботи з локальним сховищем (localStorage) у веб-додатках, дозволяючи зберігати невеликі обсяги даних у браузері без необхідності використання серверної бази даних. [4]

JSON у мобільних додатках

JSON відіграє важливу роль у мобільних застосунках, оскільки він дозволяє легко взаємодіяти з серверними API та отримувати або надсилати дані у структурованому вигляді. Багато мобільних платформ мають вбудовану підтримку JSON, що дозволяє конвертувати його у нативні структури даних (масиви, словники, об'єкти) без додаткових складнощів.

Завдяки простоті формату JSON, він використовується у популярних фреймворках для мобільної розробки, таких як React Native, Flutter, Xamarin , а також у мовах, які підтримують мобільну розробку, наприклад Swift (iOS) та Kotlin (Android) .

JSON у базах даних та Інтернеті речей (IoT)

Окрім веб-розробки та мобільних застосунків, JSON активно

використовується у сфері зберігання даних . Деякі бази даних, наприклад MongoDB та Firebase , використовують JSON або його розширену версію BSON (Binary JSON) для збереження інформації. Це забезпечує гнучкість роботи з неструктурованими даними та дозволяє легко масштабувати системи, що обробляють великі обсяги інформації. [22]

В екосистемі IoT (Інтернет речей) JSON використовується для передачі даних між пристроями та серверами, що дозволяє забезпечувати швидку та ефективну комунікацію між сенсорами, пристроями та хмарними сервісами.

Основні переваги JSON

1. Легка читабельність

JSON є текстовим форматом, який легко розуміється як розробниками, так і машинами, що дозволяє без труднощів обробляти дані та зберігати їх у зрозумілому вигляді.

2. Гнучкість і незалежність від платформи

JSON не прив’язаний до конкретної мови програмування, що робить його універсальним форматом для використання у різних середовищах.

3. Мала вага та висока швидкість обробки

JSON займає менше місця у порівнянні з XML, що дозволяє швидше передавати дані через мережу та забезпечує менше навантаження на сервери.

4. Зручність для API

JSON є стандартним форматом для обміну інформацією через API, оскільки він легко обробляється мовами програмування та підтримується багатьма бібліотеками.

5. Вбудована підтримка в більшості мов

JSON має вбудовану підтримку в багатьох популярних мовах, включаючи Python, JavaScript, Java, C#, PHP та інші.

JSON є універсальним, легким у використанні та потужним форматом обміну даними, який широко застосовується в різних сферах програмування. Його гнучкість, простота та ефективність роблять його незамінним інструментом для взаємодії між клієнтськими та серверними застосунками, роботи з базами даних та побудови IoT-систем. Завдяки своїм перевагам, JSON

залишається одним із найбільш популярних форматів збереження та передачі інформації у сучасному світі програмного забезпечення.[14]

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

SQL, або мова структурованих запитів, є стандартним інструментом для роботи з реляційними базами даних. Вона дозволяє отримувати доступ до даних, керувати ними, змінювати їх та організовувати зберігання інформації у вигляді структурованих таблиць. Використання SQL дає змогу взаємодіяти з базами даних у різних сферах, включаючи веб-розробку, фінанси, наукові дослідження та аналітику. Завдяки своїй гнучкості та ефективності SQL широко застосовується в системах управління базами даних, таких як MySQL, PostgreSQL, Oracle та Microsoft SQL Server.

Основна особливість SQL полягає у його декларативному характері, що означає, що користувач визначає лише результат, якого він хоче досягти, а система самостійно оптимізує процес отримання даних. Такий підхід значно полегшує взаємодію з базами даних, оскільки не вимагає від користувача знань про внутрішню організацію їхньої роботи.

За допомогою SQL можна створювати нові таблиці та видаляти непотрібні, додавати, оновлювати та видаляти записи, здійснювати вибірку інформації відповідно до певних умов, об’єднувати дані з кількох таблиць, застосовувати сортування та фільтрацію, а також керувати доступом до бази даних, надаючи чи відкликаючи дозволи для користувачів. Завдяки такій багатофункціональності SQL став основним інструментом для роботи з реляційними базами даних.

Популярність цієї мови пояснюється її простотою та ефективністю. SQL використовується не лише розробниками програмного забезпечення, а й аналітиками, адміністраторами баз даних та іншими фахівцями, які працюють з

великими масивами даних. Завдяки SQL можна швидко отримати доступ до потрібної інформації, обробити її та представити у зручному форматі.

Середовище застосування SQL є надзвичайно широким. У бізнесі вона використовується для керування базами даних клієнтів, аналізу фінансової інформації та формування звітності. У наукових дослідженнях SQL допомагає обробляти великі масиви даних та здійснювати складні аналітичні обчислення. У сфері веб-розробки SQL забезпечує взаємодію між сайтами та базами даних, зберігаючи інформацію про користувачів, товари, замовлення та інші елементи веб-застосунків.[19]

Ще однією важливою перевагою SQL є його кросплатформність. Він підтримується на різних операційних системах, що робить його універсальним інструментом для роботи з базами даних незалежно від середовища, в якому працює розробник. Крім того, SQL активно використовується у хмарних технологіях, що дозволяє працювати з базами даних віддалено, не прив’язуючись до конкретного пристрою.

Сучасні системи управління базами даних надають додаткові можливості для роботи з SQL. Наприклад, багато з них підтримують збереження та обробку JSON-об’єктів, що робить SQL ще більш гнучким інструментом для розробників, які працюють із сучасними веб-технологіями.

Завдяки своїй структурованості, простоті використання та широкому спектру можливостей SQL залишається основним інструментом для управління реляційними базами даних. Його застосування охоплює різні галузі, від малого бізнесу до великих корпоративних систем, наукових досліджень та аналітики великих даних. Постійний розвиток технологій баз даних та підтримка нових функцій роблять SQL невід’ємною частиною сучасного програмування та роботи з інформацією. [6]

2.3. Архітектура та патерни мобільних застосунків

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.

XAML, що розшифровується як Extensible Application Markup Language , є декларативною мовою розмітки, створеною компанією Microsoft для ініціалізації об'єктів і структурованих значень. Вона базується на XML і забезпечує зручний спосіб опису графічних інтерфейсів користувача та логіки взаємодії компонентів у програмних застосунках. Відкритість специфікації XAML надає розробникам можливість використовувати її в межах Microsoft Open Specification Promise , що сприяє широкому застосуванню цієї мови в різних програмних рішеннях. [32]

XAML активно використовується в таких технологіях, як Windows

Presentation Foundation (WPF), Silverlight, Windows UI Library (WinUI), Workflow Foundation (WF) та Universal Windows Platform (UWP) . У середовищах WPF та UWP вона слугує основним інструментом для визначення графічного інтерфейсу користувача, опису розташування елементів, їхніх властивостей, прив'язки до даних, а також обробки подій. Водночас у Workflow Foundation (WF) ця мова застосовується для створення робочих процесів , що дозволяє зручно керувати бізнес-логікою в програмних системах.

Головною перевагою XAML є її декларативний характер , що дозволяє описувати графічний інтерфейс без необхідності використання імперативного коду . Це робить її зручною для розробників, які можуть швидко створювати складні інтерфейси, працюючи лише з розміткою. Крім того, XAML підтримується широким спектром інструментів розробки , таких як Visual Studio та Blend for Visual Studio , що значно полегшує процес проєктування та налаштування інтерфейсу користувача. [25]

Завдяки використанню XAML можна легко розділити логіку програми та візуальну складову . Це особливо важливо для побудови архітектури MVVM (Model-View-ViewModel) , яка широко застосовується у WPF та UWP. Такий підхід спрощує тестування, підтримку та повторне використання коду, що робить XAML чудовим інструментом для розробки масштабованих застосунків.

Важливо зазначити, що XAML не є самостійною мовою програмування, а використовується разом із C# або іншими мовами .NET , які відповідають за реалізацію логіки взаємодії користувача з інтерфейсом. У коді XAML визначаються елементи інтерфейсу , їхні властивості та поведінка, а за допомогою C# можна додавати складні алгоритми та обробники подій.

Серед ключових особливостей XAML слід виділити гнучкість у роботі з компонентами , можливість створення ресурсів та стилів , підтримку анімацій та графічних ефектів , а також інтеграцію з мультимедійними елементами . Використання контейнерів компоновки дозволяє легко адаптувати інтерфейс під різні розміри екранів, що особливо актуально для розробки сучасних додатків із підтримкою динамічного масштабу.

Однією з важливих функцій XAML є підтримка прив'язки даних (Data

Binding) , яка дозволяє легко зв’язувати UI-елементи з джерелами даних . Це забезпечує динамічну зміну вмісту інтерфейсу відповідно до змін у внутрішній логіці програми без необхідності додаткового коду. Такий підхід значно підвищує продуктивність розробки та зменшує кількість помилок, пов’язаних із оновленням графічного інтерфейсу.

Ще однією важливою перевагою XAML є можливість створення стилів та шаблонів для елементів інтерфейсу. Це дозволяє стандартизувати вигляд програми, що особливо корисно при розробці великих проєктів. Оскільки стилі можуть бути винесені в окремі файли , це робить їх зручними для повторного використання та змін без потреби переписування коду в кожному окремому випадку.

XAML також підтримує роботу з анімацією та графікою , що дозволяє створювати привабливі візуальні ефекти без використання додаткового програмного коду. Завдяки інтеграції з DirectX та іншими графічними технологіями , XAML дає змогу створювати гладкі переходи, масштабування, ефекти затемнення та відображення рухомих об'єктів .

Щодо взаємодії з користувачем, XAML дозволяє опрацьовувати події безпосередньо в розмітці або передавати їх у логіку, написану мовою C#. Наприклад, кнопки, текстові поля, списки та інші елементи можуть бути налаштовані так, щоб виконувати певні дії у відповідь на дії користувача, такі як клацання, введення тексту або наведення миші. [29]

Важливо зазначити, що XAML розвивається разом із платформами Microsoft, і його використання розширюється завдяки новим версіям Windows, які підтримують Windows UI Library (WinUI) . Це дозволяє створювати сучасні користувацькі інтерфейси з використанням найновіших можливостей операційної системи.

Таким чином, XAML є потужним і гнучким інструментом для створення інтерфейсів користувача у застосунках на базі технологій Microsoft. Його декларативний підхід дозволяє швидко та ефективно розробляти графічний інтерфейс, відокремлюючи логіку програми від її візуальної складової. Завдяки широкій підтримці інструментів розробки, інтеграції з .NET та можливості

роботи з прив’язкою даних, стилями, анімаціями та графічними ефектами XAML залишається одним із ключових інструментів для створення сучасних застосунків у середовищі Windows.[8]

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

При розробці мобільного додатка на основі Xamarin критично важливо мати чітке уявлення про його архітектуру, дизайн та функціональність. Це дозволяє уникнути зайвих складнощів у процесі розробки, а також забезпечує ефективну реалізацію всіх необхідних функцій. В основі будь-якого мобільного застосунку в Xamarin лежить структура сторінок, де кожна виконує певну роль у навігації, обробці даних та взаємодії користувача з додатком.

Основною сторінкою є домашній екран , який виступає першим візуальним контактом користувача з програмою. Саме тут можна розміщувати ключові дані про криптовалютний ринок, такі як актуальні тренди, аналітику або новини. Ця сторінка також може слугувати навігаційним центром, забезпечуючи швидкий доступ до інших розділів програми. [27]

Наступною важливою частиною додатка є сторінка налаштувань , де користувач може змінювати параметри програми відповідно до власних потреб. Це може включати зміну теми інтерфейсу, налаштування сповіщень або вибір улюблених криптовалют для моніторингу. Додавання персоналізованих налаштувань підвищує комфорт використання програми та дає змогу користувачам адаптувати її під власні вимоги.

Окремий розділ програми – сторінка криптовалют , яка містить інформацію про різні цифрові активи, їхні поточні ціни, ринкову капіталізацію та інші ключові фінансові показники. Ця сторінка може мати детальні описи кожної криптовалюти, історичні графіки змін курсу, а також додаткову статистику, яка допомагає користувачам аналізувати ринок та приймати виважені рішення. [4]

Крім перегляду інформації, користувачам може знадобитися можливість торгівлі криптовалютами . Для цього необхідна сторінка обміну , де буде реалізований зручний інтерфейс для купівлі, продажу або обміну цифрових активів. Важливо передбачити можливість перегляду книг замовлень, історії транзакцій та динамічних оновлень вартості активів у режимі реального часу.

Це дозволить користувачам швидко реагувати на зміни ринку та ефективно управляти своїми фінансами.

Щоб забезпечити користувачів повним уявленням про ситуацію на ринку, важливо реалізувати сторінку ринків . Вона відображатиме загальні ринкові тенденції, рух цін та зміни індексів. Використовуючи цю сторінку, користувачі зможуть порівнювати криптовалютні активи, аналізувати їхню динаміку та слідкувати за станом ринку на різних біржах. [11]

Для детального аналізу вартості криптовалют доцільно реалізувати сторінку графіків . Вона міститиме інтерактивні засоби візуалізації, що допоможуть користувачам оцінювати історичні показники ринку. Графіки можуть включати свічкові діаграми, лінійні графіки та інші елементи технічного аналізу, що дозволяють визначати закономірності в зміні курсів.

Розробка цього додатка передбачає використання патерну MVVM (Model-View-ViewModel) . Це дозволить ефективно організувати код, відокремлюючи бізнес-логіку від інтерфейсу користувача. Кожна зі сторін додатка матиме три основні складові: модель (Model), подання (View) та модель відображення (ViewModel) .

Домашня сторінка складатиметься з моделі, яка зберігатиме дані про ключові криптовалютні активи або новини, макета XAML, що визначає візуальне представлення інформації, а також ViewModel, яка взаємодіє з API, забезпечуючи отримання актуальних даних.

Сторінка налаштувань міститиме модель для збереження даних користувача, XAML-подання з відповідними елементами управління та ViewModel, яка забезпечуватиме збереження та застосування змін. [7]

Сторінка криптовалют включатиме модель, що зберігатиме фінансові показники кожної криптовалюти, подання у XAML, яке відображатиме ці дані, та ViewModel, що забезпечить отримання та оновлення інформації.

Сторінка обміну працюватиме на основі моделі ордерів, подання для введення даних про купівлю та продаж, а також ViewModel, яка керуватиме виконанням торгових операцій.

Сторінка ринків матиме модель, що зберігатиме ринкові індекси та

аналітичні показники, подання, що їх відображатиме, та ViewModel, яка оновлюватиме інформацію та оброблятиме взаємодію з користувачем.

Сторінка графіків включатиме модель даних для побудови графічних візуалізацій, XAML-розмітку для графіків і ViewModel, яка керуватиме їхнім рендерингом та налаштуваннями.

Оскільки додаток передбачає взаємодію з зовнішніми API для отримання даних про криптовалютний ринок, важливо передбачити механізми обробки запитів, кешування даних і оптимізації роботи з мережею . Це дозволить зменшити навантаження на сервери та забезпечити плавну роботу програми навіть за нестабільного інтернет-з'єднання.

Дизайн додатка повинен бути інтуїтивно зрозумілим і зручним у використанні , а також підтримувати темний режим і адаптивність для різних розмірів екранів. Візуальні елементи мають відповідати Material Design або Fluent Design , що забезпечить відповідність сучасним стандартам мобільного UI/UX. [22]

Загалом, розробка додатка на основі Xamarin потребує продуманого підходу до організації коду, навігації між сторінками та інтеграції з зовнішніми сервісами. Реалізація патерну MVVM значно полегшить підтримку додатка та його подальше розширення. Чітко визначена структура додатка допоможе ефективно реалізувати всі необхідні функції, забезпечуючи якісний користувацький досвід..

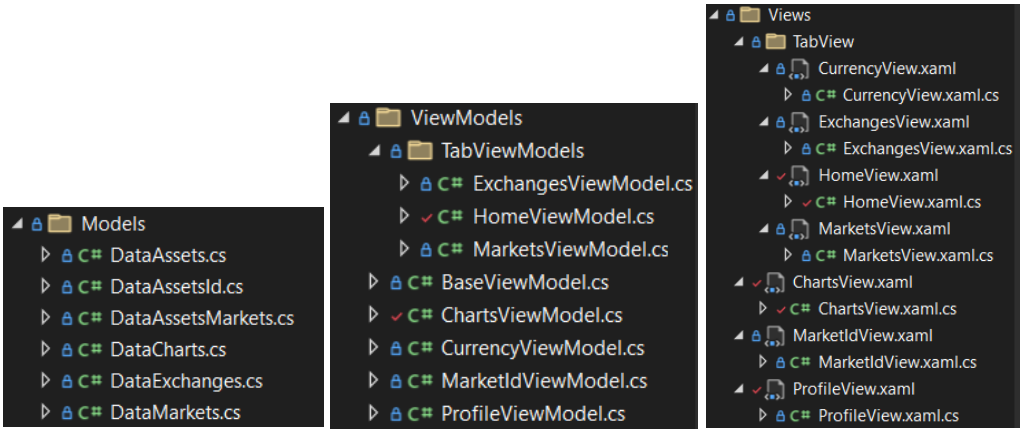

Рис. 3.1. Зображення розбивки додатку по патерну MVVM:

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

Частина з логуванням (Рис.3.2.)

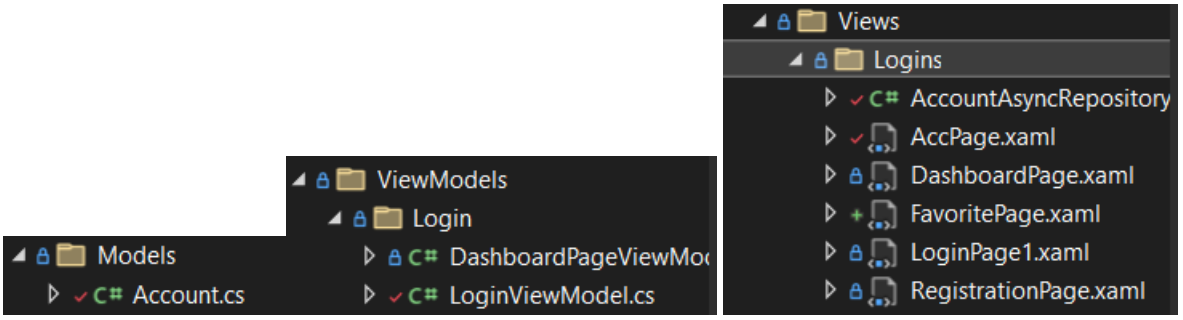

Рис. 3.2. Патерн розробки для створення профілю користувача

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.).

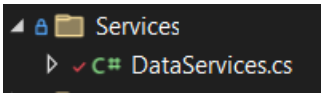

Рис. 3.3. Клас запитів для API

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

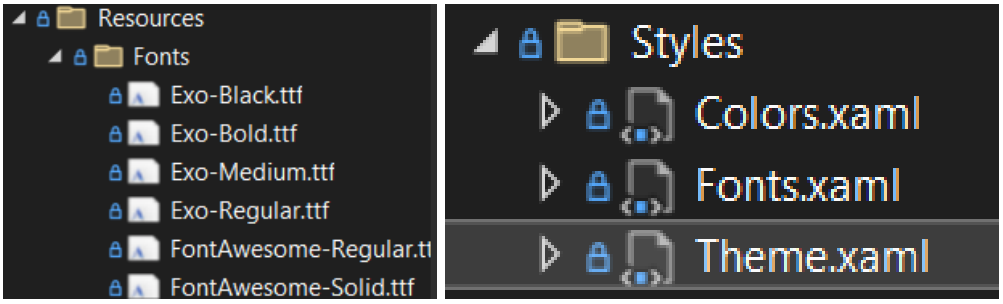

Рис.3.4. *а* – Шрифти; *б* – Теми та стилі

3.2. Розробка функціоналу для відображення даних та графічних елементів

Для початку необхідно розглянути вже створені компоненти, а саме моделі для зчитування та збереження даних, оскільки вони залишаться незмінними протягом усього процесу розробки. Ці моделі будуть слугувати основою для обробки отриманої інформації та її подальшого використання у

додатку.

Першим кроком є отримання початкових даних із зовнішнього API. Це можна зробити двома основними способами. Перший варіант передбачає використання безпосереднього запиту до API через URL, що має формат `api.coincap.io/v2/{тип даних}`. Цей підхід дозволяє напряму звертатися до сервісу, отримуючи структуровані дані у форматі JSON. Другий варіант – це використання консольного запиту, де потрібно вказати веб-адресу API, авторизаційний ключ та тип запиту. Використання цього підходу є зручним для тестування та аналізу отриманих даних ще до інтеграції в кодову базу. [17]

Якщо необхідно отримати інформацію про активи, можна виконати запит за наступним посиланням: `api.coincap.io/v2/assets`. Це дозволить отримати список криптовалютних активів із їхніми характеристиками, такими як поточна ціна, ринкова капіталізація, зміни за певний період та інші важливі фінансові показники. Отримані дані будуть використані для подальшого їх відображення в інтерфейсі додатка.

Процес роботи з API передбачає не лише отримання даних, а й обробку відповідей сервера, парсинг JSON-формату та збереження необхідної інформації у відповідних моделях. Це дає змогу організувати ефективну роботу з даними та забезпечити стабільне оновлення інформації у мобільному додатку. [3]

Далі необхідно реалізувати логіку для отримання та обробки цих даних, що дозволить динамічно оновлювати інформацію на сторінках додатка. (Рис.3.5.).

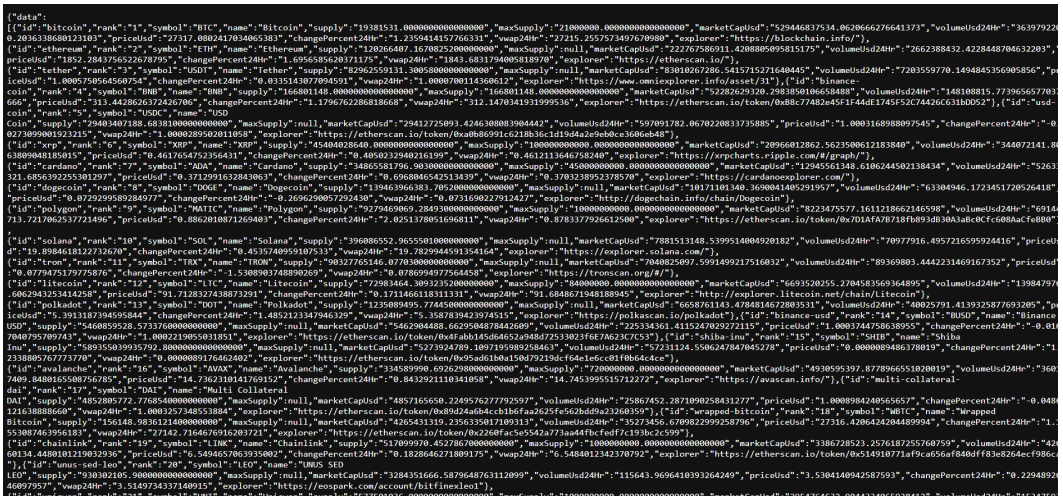
{"data":
[{"id":"bitcoin","rank":"1","symbol":"BTC","name":"Bitcoin","supply":"19381531.00000000000000000","maxSupply":"21000000.00000000000000000","marketCapUsd":"529446837534.06206662766413 73","volumeUsd24Hr":"36397922
0.20363386801231 03","priceUsd":"27317.080241703406538 3","changePercent24Hr":"1.23594141577663 31","vwap24Hr":"27215.255757349767 0980","explorer":"https://blockchain.info/"},
{"id":"ethereum","rank":"2","symbol":"ETH","name":"Ethereum","supply":"120266407.16708252000000000","maxSupply":null,"marketCapUsd":"222767586911.42088050958151 75","volumeUsd24Hr":"2662388432.42284487046322 03",
priceUsd":"1852.284375652267879 5","changePercent24Hr":"1.69565856203711 75","vwap24Hr":"1843.683179400581 8970","explorer":"https://etherscan.io/"},
{"id":"tether","rank":"3","symbol":"USDT","name":"Tether","supply":"83962559131.30053000000000000","maxSupply":null,"marketCapUsd":"83810267286.54157152716404 45","volumeUsd24Hr":"7203559770.14948453569058 56","p
iceUsd":"1.000575056456 0754","changePercent24Hr":"0.03351437070945 91","vwap24Hr":"1.000070011436061 2","explorer":"https://www.omniexplorer.info/asset/31"},{"id":"binance-
coin","rank":"4","symbol":"BNB","name":"BNB","supply":"166801148.00000000000000000","maxSupply":"166801148.00000000000000000","marketCapUsd":"52282629320.29838501066584 88","volumeUsd24Hr":"148108815.7739656577033
666","priceUsd":"313.44286263724262 706","changePercent24Hr":"1.17967622868186 68","vwap24Hr":"312.14703419131999536","explorer":"https://etherscan.io/token/0xB8c77482e45F1F44dE1745F52C74426C631bDD52"},{"id":"usd-
coin","rank":"5","symbol":"USDC","name":"USD
Coin","supply":"29403407188.68381000000000000","maxSupply":null,"marketCapUsd":"29412725093.42463088390444 2","volumeUsd24Hr":"59709178 2.067022083373 5885","priceUsd":"1.000316898809 7545","changePercent24Hr":"-0
027309900192 3215","vwap24Hr":"1.000028950211 058","explorer":"https://etherscan.io/token/0xa0b86991c6218b36c1d19d4a2e9eb0ce3606eb48"},
{"id":"xrp","rank":"6","symbol":"XRP","name":"XRP","supply":"45404028640.00000000000000000","maxSupply":"100000000000.00000000000000000","marketCapUsd":"20966012862.56235006121838 40","volumeUsd24Hr":"344072141.8
638090481850 15","priceUsd":"0.461765475235643 1","changePercent24Hr":"0.40502329402161 99","vwap24Hr":"0.461211364675820 40","explorer":"https://xrpcharts.ripple.com/#/graph/"},
{"id":"cardano","rank":"7","symbol":"ADA","name":"Cardano","supply":"34865581796.90300000000000000","maxSupply":"45000000000.00000000000000000","marketCapUsd":"12945561348.61062445020138434","volumeUsd24Hr":"5263
321.6856392255301297","priceUsd":"0.371299163284306 3","changePercent24Hr":"0.69680465425133 99","vwap24Hr":"0.370323895237857 0","explorer":"https://cardanoexplorer.com/"},
{"id":"dogecoin","rank":"8","symbol":"DOGE","name":"Dogecoin","supply":"139463966383.70520000000000000","maxSupply":null,"marketCapUsd":"10171101340.36900414052919 57","volumeUsd24Hr":"63304946.17234517205264 18"
priceUsd":"0.072929598928497 7","changePercent24Hr":"-0.26962900572924 30","vwap24Hr":"0.073169022791242 2","explorer":"http://dogechain.info/chain/Dogecoin"},
{"id":"polygon","rank":"9","symbol":"MATIC","name":"Polygon","supply":"9279469069.284930000000000000","maxSupply":"10000000000.00000000000000000","marketCapUsd":"8223475577.16112186621465 98","volumeUsd24Hr":"6914
713.721706253772149 6","priceUsd":"0.886201087126940 3","changePercent24Hr":"2.02513780516968 11","vwap24Hr":"0.878337792661250 0","explorer":"https://etherscan.io/token/0x7D1AFA7B718fb893dB30A3aBc0Cfc608AaCfeB80"
},
{"id":"solana","rank":"10","symbol":"SOL","name":"Solana","supply":"396086552.965550100000000","maxSupply":null,"marketCapUsd":"7881513148.539951400492 0182","volumeUsd24Hr":"70977916.495721659592 4416","priceUs
d":"19.898461812273 670","changePercent24Hr":"0.453574095910 7533","vwap24Hr":"19.782994459135 4164","explorer":"https://explorer.solana.com/"},
{"id":"tron","rank":"11","symbol":"TRX","name":"TRON","supply":"90327765146.07703000000000000","maxSupply":null,"marketCapUsd":"7040825097.59914992175160 32","volumeUsd24Hr":"89369803.444223146916 7352","priceUsd
":"0.077947517977587 6","changePercent24Hr":"-1.53089037488902 69","vwap24Hr":"0.078699497756 4458","explorer":"https://tronscan.org/#/"},
{"id":"litecoin","rank":"12","symbol":"LTC","name":"Litecoin","supply":"72983464.30932352000000000","maxSupply":"84000000.00000000000000000","marketCapUsd":"6693520255.270458356936 4895","volumeUsd24Hr":"13984797
.66029432534142 58","priceUsd":"91.712832743887 3293","changePercent24Hr":"0.17134661183311 331","vwap24Hr":"91.684867194818 8945","explorer":"https://explorer.litecoin.net/chain/Litecoin"},
{"id":"polkadot","rank":"13","symbol":"DOT","name":"Polkadot","supply":"1235089495.774450000000000","maxSupply":null,"marketCapUsd":"6658761143.478481467280 3531","volumeUsd24Hr":"40025791.413932587769 3205","p
iceUsd":"5.391318739459 5844","changePercent24Hr":"1.48521233479463 29","vwap24Hr":"5.358783942397 45155","explorer":"https://polkascan.io/polkadot"},{"id":"binance-usd","rank":"14","symbol":"BUSD","name":"Binance
USD","supply":"5460859528.573376000000000","maxSupply":null,"marketCapUsd":"5462904488.662950487844 2689","volumeUsd24Hr":"225334361.411524702927 2115","priceUsd":"1.000374475863 8955","changePercent24Hr":"-0.010
704075097 743","vwap24Hr":"1.000221995031 851","explorer":"https://etherscan.io/token/0x4fabb145d64652a948d72533023f6E7A623C7C53"},{"id":"shiba-inu","rank":"15","symbol":"SHIB","name":"Shiba
Inu","supply":"589355039935792.800000000000000","maxSupply":null,"marketCapUsd":"5273924789.109719598925 8463","volumeUsd24Hr":"57231124.550624784704 5278","priceUsd":"0.000008948637 8019","changePercent24Hr":"1
233880576773 770","vwap24Hr":"0.000008917646 2402","explorer":"https://etherscan.io/token/0x95ad61b0a150d79219dcf64e1e6cc0110b64c4ce"},
{"id":"avalanche","rank":"16","symbol":"AVAX","name":"Avalanche","supply":"334589990.692629800000000 00","maxSupply":"720000000.00000000000000000","marketCapUsd":"4930595397.877896655102 0019","volumeUsd24Hr":"360
7409.848016550875 785","priceUsd":"14.736231014176 9152","changePercent24Hr":"0.84329211103410 58","vwap24Hr":"14.745399551571 2272","explorer":"https://avascan.info/"},{"id":"multi-collateral-
dai","rank":"17","symbol":"DAI","name":"Multi Collateral
DAI","supply":"4852805772.776854000000000","maxSupply":null,"marketCapUsd":"4857165650.224957627792597","volumeUsd24Hr":"25867452.287109025843 1277","priceUsd":"1.000898424056565 7","changePercent24Hr":"-0.048
121638888660","vwap24Hr":"1.000325734855 3884","explorer":"https://etherscan.io/token/0x89d24a6b4ccb1b6faa2625fe562bdd9a23260359"},{"id":"wrapped-bitcoin","rank":"18","symbol":"WBTC","name":"Wrapped
Bitcoin","supply":"156148.983612140000000","maxSupply":null,"marketCapUsd":"4265431319.235633501709313","volumeUsd24Hr":"35273456.670982299925 8796","priceUsd":"27316.420642424044 89994","changePercent24Hr":"1.1
553087463956 183","vwap24Hr":"27142.716467691620 721","explorer":"https://etherscan.io/token/0x2260fac5e5542a773aa44fbcfedf7c193bc2c599"},
{"id":"chainlink","rank":"19","symbol":"LINK","name":"Chainlink","supply":"517099970.452786700000000 00","maxSupply":"1000000000.00000000000000000","marketCapUsd":"3386728523.257618725576 0759","volumeUsd24Hr":"420
60134.448801012190329 36","priceUsd":"6.549465706393500 2","changePercent24Hr":"0.18286462718091 75","vwap24Hr":"6.548401234230792","explorer":"https://etherscan.io/token/0x514910771af9ca656af840dff83e8264ecf986ca
"},{"id":"unus-sed-leo","rank":"20","symbol":"LEO","name":"UNUS SED
LEO","supply":"930302105.900000000000000","maxSupply":null,"marketCapUsd":"3284351666.587964876311 2099","volumeUsd24Hr":"115643.969641039326 4249","priceUsd":"3.530414094258 7593","changePercent24Hr":"0.229489 46097957","vwap24Hr":"3.514973433714 0915","explorer":"https://eospark.com/account/bitfinexleo1"},

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

Скопійовані дані, отримані з API, необхідні для створення класу, який відповідатиме за їх зчитування та обробку. Щоб спростити процес роботи з JSON-структурами та уникнути можливих помилок при ручному створенні класів, можна скористатися вбудованими інструментами Visual Studio.

Для автоматичного генерування класів відповідно до отриманої JSON-структури потрібно виконати такі кроки. Спочатку у Visual Studio необхідно відкрити проєкт, у якому буде оброблятися API-запит. Далі слід перейти до класу DataAssets, де буде збережено отримані дані. Щоб швидко створити клас на основі JSON, необхідно використати наступний інструмент: View → Paste Special → Paste JSON As Classes. Ця функція дозволяє автоматично конвертувати скопійований JSON у відповідні C#-класи, що значно спрощує процес розробки.

Цей метод допомагає не лише уникнути помилок під час ручного написання класів, але й зберігає їхню коректну структуру, відповідну отриманим JSON-даним. Відповідно, всі інші класи, які будуть використовуватися для обробки аналогічних API-відповідей, можна створювати за цим же принципом. Це забезпечить узгодженість структури даних і пришвидшить розробку, дозволяючи одразу перейти до реалізації логіки взаємодії з API та відображення інформації в інтерфейсі додатка.[21]

Після створення класів можна буде налаштувати процес зчитування та серіалізації JSON, що дасть змогу отримувати та зберігати отримані дані у зручному форматі для подальшої обробки. Це є ключовим етапом у побудові мобільного додатка, який використовує зовнішні джерела даних, зокрема API для криптовалютного ринку.

Наступним етапом у розробці є написання скриптів для отримання даних з API та десеріалізації JSON-файлів. Це необхідно для обробки інформації про криптовалютні активи, ринки, біржі та графіки. Щоб реалізувати цю функціональність, у проєкті створено папку Services, в якій знаходиться клас DataServices. Його основним завданням є управління з'єднанням із REST API та забезпечення взаємодії з сервером для отримання необхідних даних.

У класі DataServices визначено приватне поле `_httpClient` типу `HttpClient`, яке використовується для надсилання HTTP-запитів до API. Конструктор класу ініціалізує цей об'єкт і встановлює його базову адресу як `"https://api.coincap.io/v2/"`, що означає, що всі наступні HTTP-запити будуть виконуватися відносно цієї адреси.

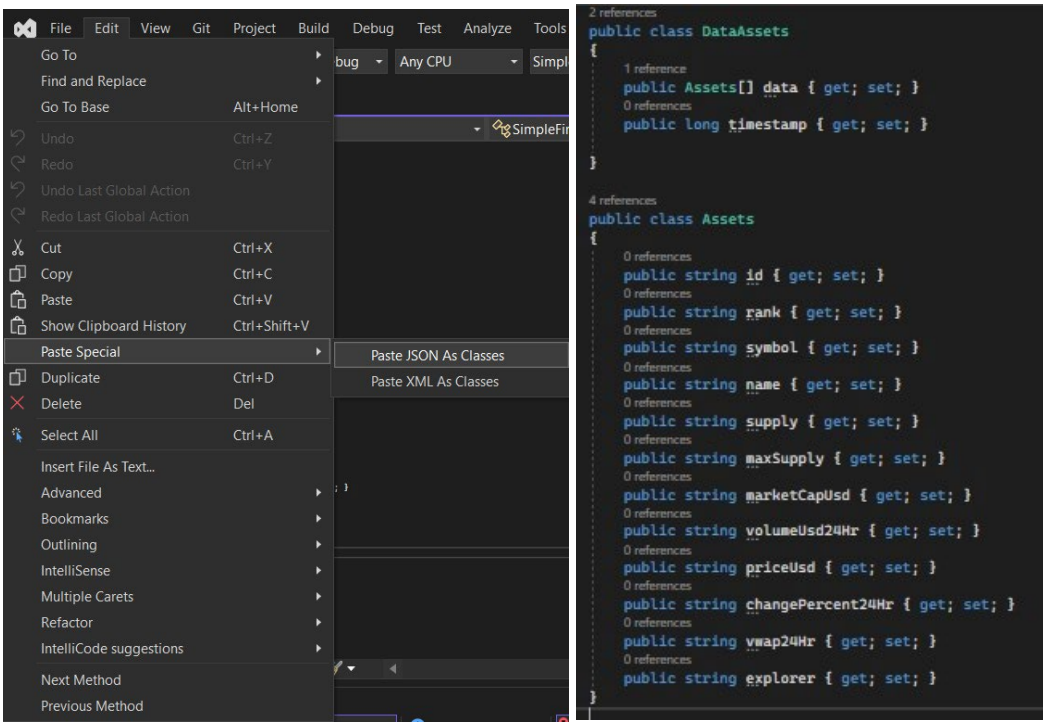

Рис.3.6. *а* – Функція; *б* – Результат генерації

Клас включає кілька загальнодоступних методів, які використовуються для отримання різних типів даних із сервера. Кожен метод виконує асинхронну операцію, використовуючи `Task`, що дозволяє обробляти запити без блокування основного потоку виконання програми.

Метод `GetAssetsAsync()` використовується для отримання списку криптовалютних активів. Він надсилає `HTTP GET`-запит до кінцевої точки "assets" і повертає отриману відповідь у вигляді об'єкта `DataAssets`. Десеріалізація даних здійснюється за допомогою `JsonConvert.DeserializeObject`, що дозволяє конвертувати JSON у відповідний C#-об'єкт.

Метод `GetAssetsIdAsync(string Id)` надсилає запит до кінцевої точки "assets/{Id}", де параметр `{Id}` відповідає конкретному ідентифікатору активу. Він отримує детальну інформацію про вказаний криптовалютний актив, після

чого десеріалізує JSON-відповідь у об'єкт `DataAssetsId`.

Метод `GetMarketsAsync()` відповідає за отримання списку криптовалютних ринків. Для цього він надсилає запит до кінцевої точки "markets", після чого отримані дані конвертуються в об'єкт `DataMarkets`.

Метод `GetExchangesAsync()` використовується для отримання переліку криптовалютних бірж. Він надсилає запит до кінцевої точки "exchanges", отримує відповідь і перетворює її у формат `DataExchanges`, що містить необхідну інформацію про біржі.[4]

Метод `GetMarketsIdAsync(string market)` призначений для отримання списку ринків для конкретного криптовалютного активу. Він надсилає `GET`-запит до кінцевої точки "assets/{market}/markets", де `{market}` – це параметр, що визначає назву ринку. Після отримання відповіді JSON-документ перетворюється в об'єкт `DataAssetsMarkets`.

Метод `GetChartsAsync(string assetId, string date)` використовується для отримання історичних даних про графіки конкретного криптовалютного активу. Для цього надсилається `HTTP GET`-запит до кінцевої точки "assets/{assetId}/history?interval={date}", де `{assetId}` визначає ідентифікатор активу, а `{date}` – інтервал, за яким потрібно отримати дані. Отримані дані десеріалізуються в об'єкт `DataCharts` і використовуються для відображення історичних змін вартості активів.

Реалізація цього класу дозволяє забезпечити ефективний зв'язок між клієнтською частиною програми та API, надаючи користувачам актуальні ринкові дані в режимі реального часу. Використання асинхронних методів гарантує швидке отримання інформації без затримок у роботі додатку, що робить його зручним та продуктивним. Наступним кроком є інтеграція цього функціоналу у візуальну частину програми, що дозволить користувачам переглядати дані та взаємодіяти з ними безпосередньо через інтерфейс мобільного додатка.

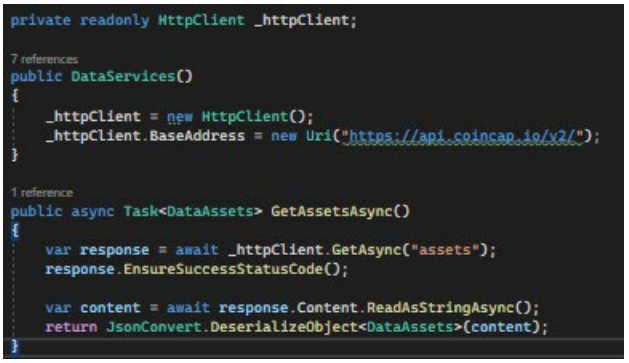

Рис.3.7. Загальнодоступний метод та конструктор HTTP-запитів.

У процесі розробки мобільного додатку на основі Xamarin важливо правильно організувати структуру коду, щоб забезпечити ефективне управління даними та їх відображення у користувацькому інтерфейсі. Для цього у папці ViewModels створено базовий клас BaseViewModel, який реалізує інтерфейс `INotifyPropertyChanged`, необхідний для автоматичного оновлення інтерфейсу при зміні даних.[34]

Ключовими властивостями цього класу є:

- IsBusy – булеве значення, що вказує, чи виконується в даний момент асинхронна операція.

- Title – рядок, що містить заголовок поточного представлення.

- Asset – список об'єктів `Assets`, які зберігають дані про криптовалютні активи.

- Exchange – список об'єктів `Exchanges`, що містять інформацію про біржі.

- Market – список об'єктів `AssetsMarkets`, що зберігають ринкові дані активів.

- Charts – список об'єктів `Charts`, у яких містяться історичні дані для побудови графіків.

Крім того, у класі визначені методи для асинхронного отримання даних за допомогою `DataServices`:

- `LoadAssetsAsync()` – завантажує список криптовалютних активів, використовуючи `GetAssetsAsync()`, і зберігає їх у властивості `Asset`.

- `LoadAssetsAsync(string id)` – отримує детальну інформацію про

конкретний актив за заданим ідентифікатором і зберігає результат у змінній `firstItem`.

- `LoadExchangesAsync()` – виконує запит `GetExchangesAsync()` для отримання переліку бірж та збереження у властивості `Exchange`.

- `LoadMarketsAsync(string id)` – завантажує список ринків для вказаного активу, викликаючи `GetMarketsAsync(id)`, і зберігає їх у `Market`.

- `LoadChartsAsync(string i, string b)` – отримує історичні дані про графіки для вказаних параметрів `i` та `b`, після чого результат зберігається у `Charts`.

Окрім роботи з даними, клас реалізує методи `SetProperty` та `OnPropertyChanged`, які необхідні для інтерфейсу `INotifyPropertyChanged`. Вони дозволяють автоматично оновлювати прив’язані до інтерфейсу користувача властивості, що допомагає динамічно змінювати відображення даних без додаткового втручання користувача. Використання `CallerMemberName` у методі `OnPropertyChanged` дозволяє автоматично отримувати ім'я властивості, яка змінюється, що зменшує кількість потенційних помилок, пов'язаних із прив’язкою даних.[12]

У конструкторі `BaseViewModel` створюється екземпляр `DataServices`, який використовується для отримання інформації про криптовалюти, біржі, ринки та історичні графіки. Завдяки цьому клас `BaseViewModel` є універсальним і може бути успадкований іншими моделями представлення, які працюють із конкретними типами даних.

Розглянемо дві моделі представлення: HomeViewModel та MarketsViewModel. Обидві моделі успадковують `BaseViewModel`, що дозволяє їм використовувати основні властивості та методи базового класу. Вони містять конструктори, в яких встановлюється заголовок `Title` для сторінок, а також викликається відповідний асинхронний метод для завантаження даних у момент ініціалізації сторінки.[9]

MarketsViewModel додатково містить кілька перевантажених конструкторів, які викликають основний конструктор, але з різними параметрами. Це дозволяє гнучко сортувати та фільтрувати отримані ринкові

дані за певними критеріями. Таким чином, ця модель представлення може створюватися з різними параметрами, що дає змогу адаптувати відображення ринків відповідно до вимог користувача.

Завдяки використанню базової моделі представлення `BaseViewModel`, структуру коду вдалося зробити більш модульною та підтримуваною. Це дозволяє легко розширювати функціонал додатку, додаючи нові сторінки та моделі представлення без необхідності дублювати код. Крім того, використання `INotifyPropertyChanged` забезпечує ефективне оновлення інтерфейсу користувача в режимі реального часу, що покращує досвід взаємодії з додатком.

Наступним кроком є розробка View – сторінок користувацького інтерфейсу, які будуть безпосередньо відображати отримані дані, а також реалізація механізмів навігації між цими сторінками для зручного перегляду інформації про криптовалютний ринок.

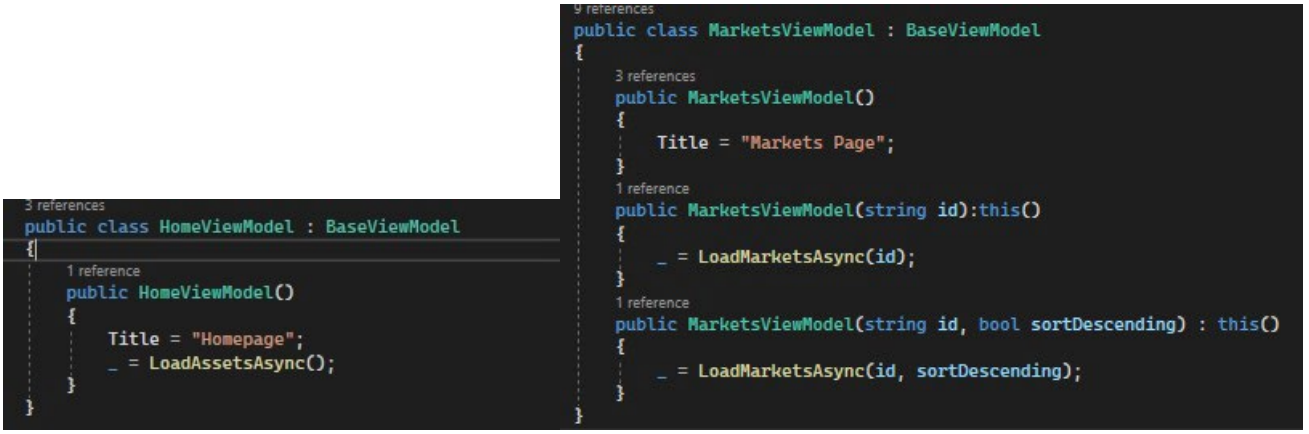

Рис. 3.8. *а* – HomeViewModel; *б* – MarketsViewModel

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.[27]

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

Для реалізації користувацького інтерфейсу програми в папці Views необхідно створити сторінки, які відповідатимуть за відображення різних розділів додатку. Оскільки додаток використовує MVVM (Model-View-ViewModel), важливо правильно організувати взаємодію між інтерфейсом користувача та моделями представлення. Для цього використовується XAML, що дозволяє декларативно визначати розміщення елементів та їх зв'язки з даними.

Першим кроком є створення головного інтерфейсу програми. Основний контейнер сторінки визначається за допомогою `ContentPage`, а всередині нього

розміщується головна робоча область. Вона може складатися з `Grid` або `StackLayout`, залежно від необхідної структури. Додавання кнопок, списків та інших елементів здійснюється шляхом визначення `Button`, `ListView`, `CollectionView` або інших відповідних компонентів.[11]

Головна сторінка (HomePage.xaml)

Головна сторінка виступає центральним елементом програми, з якої користувач може переходити до інших розділів. Вона може містити кнопки навігації, список останніх активів, а також інші важливі елементи, які допоможуть користувачу швидко отримати необхідну інформацію.

```xml
<ContentPage xmlns="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="CryptoApp.Views.HomePage">
    <StackLayout Padding="10">
        <Label Text="Огляд ринку криптовалют"
               FontSize="24"
               FontAttributes="Bold"
               HorizontalOptions="Center"/>
        <Button Text="Переглянути активи" Command="{Binding NavigateToAssetsCommand}"/>
        <Button Text="Відкрити біржі" Command="{Binding NavigateToExchangesCommand}"/>
        <Button Text="Ринки" Command="{Binding NavigateToMarketsCommand}"/>
        <CollectionView ItemsSource="{Binding Asset}">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <StackLayout Padding="5">
                        <Label Text="{Binding Name}" FontSize="18"/>
                        <Label Text="{Binding PriceUsd, StringFormat='Ціна:
```

```
{0:C}'}"/>
                    </StackLayout>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </StackLayout>
</ContentPage>
```

Ця сторінка містить заголовок, кнопки для переходу до інших розділів, а також список активів, який буде заповнюватися даними з моделі представлення `HomeViewModel`.

Сторінка налаштувань (SettingsPage.xaml)

Ця сторінка дозволяє користувачам змінювати параметри програми, такі як вибір теми, налаштування сповіщень тощо.

```xml
<ContentPage xmlns="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="CryptoApp.Views.SettingsPage">
    <StackLayout Padding="10">
        <Label Text="Налаштування" FontSize="24" FontAttributes="Bold"/>
        <Switch IsToggled="{Binding DarkModeEnabled}" />
        <Label Text="Темний режим"/>
        <Switch IsToggled="{Binding NotificationsEnabled}" />
        <Label Text="Сповіщення"/>
    </StackLayout>
</ContentPage>
```

Завдяки двосторонньому зв’язуванню (`Binding`), зміни, внесені на цій сторінці, будуть автоматично оновлюватися у відповідній моделі представлення `SettingsViewModel`.

Сторінка активів (AssetsPage.xaml)

Ця сторінка відображає список усіх доступних криптовалют з можливістю перегляду детальної інформації.

```xml
<ContentPage xmlns="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="CryptoApp.Views.AssetsPage">
    <StackLayout Padding="10">
        <Label Text="Список криптовалют" FontSize="24" FontAttributes="Bold"/>
        <SearchBar Placeholder="Пошук криптовалют" Text="{Binding SearchQuery}"/>
        <CollectionView ItemsSource="{Binding Asset}">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Frame Padding="10">
                        <StackLayout>
                            <Label Text="{Binding Name}" FontSize="18"/>
                            <Label Text="{Binding Symbol}" FontSize="16" TextColor="Gray"/>
                            <Label Text="{Binding PriceUsd, StringFormat='Ціна: {0:C}'}"/>
                            <Button Text="Детальніше" Command="{Binding ViewAssetDetailsCommand}" CommandParameter="{Binding}"/>
                        </StackLayout>
                    </Frame>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </StackLayout>
```

```
</ContentPage>
```

Ця сторінка включає пошуковий рядок, який допомагає користувачам швидко знаходити потрібні активи, та `CollectionView` для відображення списку криптовалют.

Сторінка обміну (ExchangePage.xaml)

На цій сторінці користувачі можуть переглядати біржі та здійснювати торгівлю.

```xml
<ContentPage xmlns="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="CryptoApp.Views.ExchangePage">
    <StackLayout Padding="10">
        <Label Text="Обмін криптовалют" FontSize="24" FontAttributes="Bold"/>
        <Picker Title="Оберіть валюту" ItemsSource="{Binding Asset}" ItemDisplayBinding="{Binding Name}" SelectedItem="{Binding SelectedAsset}"/>
        <Entry Placeholder="Введіть суму" Keyboard="Numeric" Text="{Binding ExchangeAmount}"/>
        <Button Text="Обміняти" Command="{Binding ExchangeCommand}"/>
        <Label Text="{Binding ExchangeResult}" FontSize="18" FontAttributes="Bold"/>
    </StackLayout>
</ContentPage>
```

Ця сторінка дозволяє користувачам обирати криптовалюту, вводити суму для обміну та отримувати результат через команду `ExchangeCommand`.

Сторінка ринків (MarketsPage.xaml)

Ця сторінка містить аналітичні дані про криптовалютні ринки.

```xml
<ContentPage xmlns="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             x:Class="CryptoApp.Views.MarketsPage">
    <StackLayout Padding="10">
        <Label Text="Криптовалютні ринки" FontSize="24" FontAttributes="Bold"/>
        <CollectionView ItemsSource="{Binding Market}">
            <CollectionView.ItemTemplate>
                <DataTemplate>
                    <Frame Padding="10">
                        <StackLayout>
                            <Label Text="{Binding ExchangeId}" FontSize="18"/>
                            <Label Text="{Binding PriceUsd, StringFormat='Ціна: {0:C}'}"/>
                        </StackLayout>
                    </Frame>
                </DataTemplate>
            </CollectionView.ItemTemplate>
        </CollectionView>
    </StackLayout>
</ContentPage>
```

Ця сторінка відображає список ринків та дозволяє користувачам переглядати ціни активів у різних біржах.

Реалізація навігації

Для переходу між сторінками використовується `Shell` або

`NavigationPage`. У AppShell.xaml можна налаштувати навігацію:

```xml
<Shell>
    <ShellContent Title="Головна" ContentTemplate="{DataTemplate local:HomePage}" />
    <ShellContent Title="Активи" ContentTemplate="{DataTemplate local:AssetsPage}" />
    <ShellContent Title="Обмін" ContentTemplate="{DataTemplate local:ExchangePage}" />
    <ShellContent Title="Ринки" ContentTemplate="{DataTemplate local:MarketsPage}" />
    <ShellContent Title="Налаштування" ContentTemplate="{DataTemplate local:SettingsPage}" />
</Shell>
```

Реалізація користувацького інтерфейсу у Views дозволяє створити зручну навігацію між сторінками, ефективно відображати необхідні дані та забезпечувати інтерактивну взаємодію користувача з додатком за допомогою MVVM. Оскільки MVVM дозволяє відокремити логіку від інтерфейсу користувача, це полегшує масштабування додатка, його подальшу підтримку та тестування.

Подальшим важливим етапом у розробці є покращення інтеграції з API для забезпечення більш ефективного отримання та обробки даних. Оновлення інформації про криптовалютні ринки в режимі реального часу є ключовою функцією, яка забезпечує користувачам доступ до актуальних даних про активи, ринки та біржі.

Для досягнення цього можна застосувати WebSocket-з'єднання або використовувати підписку на оновлення даних через API з використанням SignalR чи Polling. У разі застосування WebSocket, додаток зможе безпосередньо отримувати дані про зміну курсів у реальному часі без

необхідності постійно надсилати запити до сервера. Це значно зменшує навантаження на мережу та підвищує продуктивність роботи програми.

Також, потрібно оптимізувати асинхронні запити до API, використовуючи HttpClient та Task-based асинхронне програмування (async/await). Це дозволить не блокувати основний потік виконання програми та зробить взаємодію з користувачем плавною і швидкою.[14]

Окрім цього, варто додати механізми кешування даних, щоб уникнути надмірних запитів до API, зменшити затримку при оновленні інтерфейсу та забезпечити можливість роботи в офлайн-режимі. Це можна реалізувати за допомогою SQLite або Preferences API для тимчасового зберігання отриманих даних.

Таким чином, впровадження покращеної інтеграції з API та забезпечення оновлення інформації в реальному часі сприятиме більшій ефективності роботи додатка, дозволяючи користувачам отримувати найсвіжішу інформацію без затримок. Це зробить додаток зручнішим, продуктивнішим та надійнішим у використанні. (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

Після створення відображення для кожної сторінки наступним важливим кроком є реалізація механізмів зв’язку між інтерфейсом користувача та його функціональною частиною. Це передбачає встановлення контексту прив’язки

даних до конкретного екземпляра моделі представлення, а також створення методів для обробки подій, таких як натискання кнопок чи вибір елементів зі списку. [23]

Розглянемо приклад реалізації back-end для HomeView. Даний клас наслідує ContentPage і містить атрибут (XamlCompilation Options.Compile), який оптимізує компіляцію XAML-розмітки для покращення продуктивності додатка. Основною складовою класу є змінна _viewModel типу HomeViewModel, що виступає моделлю представлення для сторінки. У конструкторі HomeView() виконується ініціалізація інтерфейсу користувача, а також встановлення BindingContext для зв'язку з екземпляром HomeViewModel, що дає змогу автоматично оновлювати інтерфейс у разі зміни даних.

Однією з ключових функцій є метод ProfileButton_Click(), що викликається при натисканні на кнопку профілю. Він ініціює навігацію до сторінки авторизації LoginPage1 за допомогою Navigation.PushAsync(). Додатково реалізовано перевизначений метод OnAppearing(), який виконується при завантаженні сторінки. Виклик методу _viewModel.OnAppearing() дозволяє підготувати необхідні дані або оновити інтерфейс.

Метод OnButtonClicked() використовується для обробки натискання кнопки, яка викликає подію Clicked. Його логіка полягає в перевірці існування ключа UserAlreadyloggedIn через Xamarin.Essentials.Preferences.Get(). Якщо такий ключ є, отримується активний об'єкт RelatedObject із контексту прив'язки, після чого він додається в базу даних. У разі його відсутності відбувається перенаправлення на сторінку входу LoginPage1. [19]

Ще однією важливою функцією є OnItemSelected(), яка виконується під час вибору елемента зі списку. Вона отримує вибраний об'єкт Assets і передає його ідентифікатор до ChartsView, що дозволяє переглянути графічне представлення вибраного активу.

За аналогічним принципом реалізовано функціональність реєстрації та авторизації користувачів. У моделі Account створено два класи: Account та RelatedObject, які слугують представленням відповідних таблиць у базі даних у рамках ORM-підходу (Object-Relational Mapping). Клас Account відповідає

таблиці "Accounts" та містить такі поля, як Id, FirstName, LastName, Email, UserName та Password, що зберігають дані користувача. Крім того, у ньому є властивість RelatedObjects, яка визначає зв'язок "один до багатьох" між таблицями "Accounts" і "RelatedObject". Завдяки цьому кожен запис в Account може мати декілька пов'язаних елементів у таблиці RelatedObject.

Сам клас RelatedObject відображає таблицю з аналогічною назвою, а його структура включає поля Id, SomeData та UserId. Поле UserId виступає зовнішнім ключем, що зв'язує об'єкти RelatedObject з відповідними записами в таблиці Accounts.

Наступним кроком є реалізація моделі представлення LoginViewModel, що відповідає за логіку процесу авторизації користувачів у додатку. Основними її елементами є властивість MyloginRequestModel, яка містить інформацію, введену користувачем, а також команда LoginCommand, що запускає процес входу, викликаючи метод PerformLoginOperation. У цьому методі відбувається перевірка введених даних шляхом порівняння їх із записами, що містяться в базі даних, яку отримує метод App.Database.GetAccountItemsAsync(). Якщо знайдено відповідний обліковий запис, то у налаштуваннях Preferences встановлюється прапорець UserAlreadyloggedIn, що підтверджує успішний вхід, а також зберігається UserLogin з іменем користувача. Після цього здійснюється навігація до DashboardPage, яка виступає основною сторінкою після авторизації.

У процесі розробки мобільного додатку критично важливим є впровадження механізмів, які забезпечують коректну взаємодію користувача з системою, зокрема через налаштування прив'язки даних, процесу авторизації та роботи з базою даних. Реалізація цих механізмів сприяє не лише зручності використання додатку, а й значно підвищує його продуктивність, дозволяючи швидко та ефективно керувати інформацією. Завдяки правильно налаштованій інтеграції бази даних із додатком можна гарантувати безпечне збереження та обробку користувацьких даних, що особливо важливо для програм, що працюють із конфіденційною інформацією.[8]

Для забезпечення безперебійної роботи всіх функціональних елементів необхідно створити базу даних, яка зможе зберігати облікові записи користувачів

та інші важливі об'єкти додатку. Це дозволить реалізувати ефективну систему управління обліковими записами, включаючи можливості реєстрації, входу та зміни параметрів користувача. База даних повинна бути побудована таким чином, щоб забезпечувати швидкий доступ до інформації, гарантуючи високу продуктивність додатку навіть при значному обсязі збережених 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

Один із ключових аспектів у цій реалізації – використання асинхронного підходу для взаємодії з базою даних. Це дозволяє уникнути блокування основного потоку виконання програми, що покращує її стабільність і плавність роботи. Доступ до сховища даних здійснюється через спеціальний клас AccountAsyncRepository, який використовується для зберігання та отримання об’єктів відповідного типу. У коді програми реалізовано механізм, що забезпечує централізований доступ до бази даних за допомогою статичної властивості Database. Завдяки цьому інші частини додатку можуть безперешкодно звертатися до сховища даних, виконуючи необхідні операції, такі як збереження, оновлення або вилучення інформації.[26]

Загалом, впровадження системи прив’язки даних, авторизації та роботи з базою даних створює надійну основу для стабільної та ефективної роботи мобільного додатку. Це не лише покращує користувацький досвід, але й дозволяє розробникам легко масштабувати програму, додаючи нові функції та інтегруючи її з іншими сервісами. Використання правильної архітектури даних у поєднанні з оптимізованими алгоритмами обробки запитів до бази дозволяє значно підвищити продуктивність додатку, роблячи його більш зручним, швидким та

безпечним.

3.3. Демонстрація роботи застосунку

У додатку реалізовано чотири основні сторінки, кожна з яких виконує певну функцію та надає користувачам необхідну інформацію про криптовалютний ринок. До них належать: Головна сторінка, Ринок, Біржі та Зміни.

Головна сторінка слугує основним інформаційним центром додатку, де представлений перелік усіх криптовалют, які наразі відстежуються системою. Користувач має можливість здійснювати сортування списку за різними критеріями, зокрема за ринковою капіталізацією, поточною ціною або торговим обсягом. Крім того, додаток передбачає функціонал перегляду історичних даних у вигляді графіків, що дозволяє користувачам аналізувати зміни вартості обраної криптовалюти за певний проміжок часу.

Сторінка Ринок надає детальну інформацію про ціни на конкретну криптовалюту, залежно від біржі, на якій вона торгується. Користувач має змогу відстежувати актуальні дані з різних криптовалютних платформ та сортувати інформацію за показником зміни вартості активу, що дає змогу швидко оцінити динаміку ринку та вибрати найкращий варіант для купівлі чи продажу.

Сторінка Біржі містить перелік всіх криптовалютних бірж, на яких можна здійснювати операції з купівлі та продажу цифрових активів. Ця сторінка дозволяє користувачеві отримати уявлення про доступні торгові платформи, порівняти їх та обрати найбільш відповідну для своїх потреб.[22]

На сторінці Зміни користувач може здійснити порівняння курсу двох криптовалют, отримуючи інформацію про їхню відносну вартість. Це особливо корисно для трейдерів, які проводять операції обміну між різними цифровими активами та хочуть швидко визначити вигідні умови для конвертації.

Завдяки чіткій структурі та функціональним можливостям кожної сторінки, додаток забезпечує користувачам швидкий доступ до актуальної

інформації про криптовалютний ринок, сприяючи зручності у використанні та ефективному прийняттю фінансових рішень.

Основні сторінки зображені на рис.3.11.

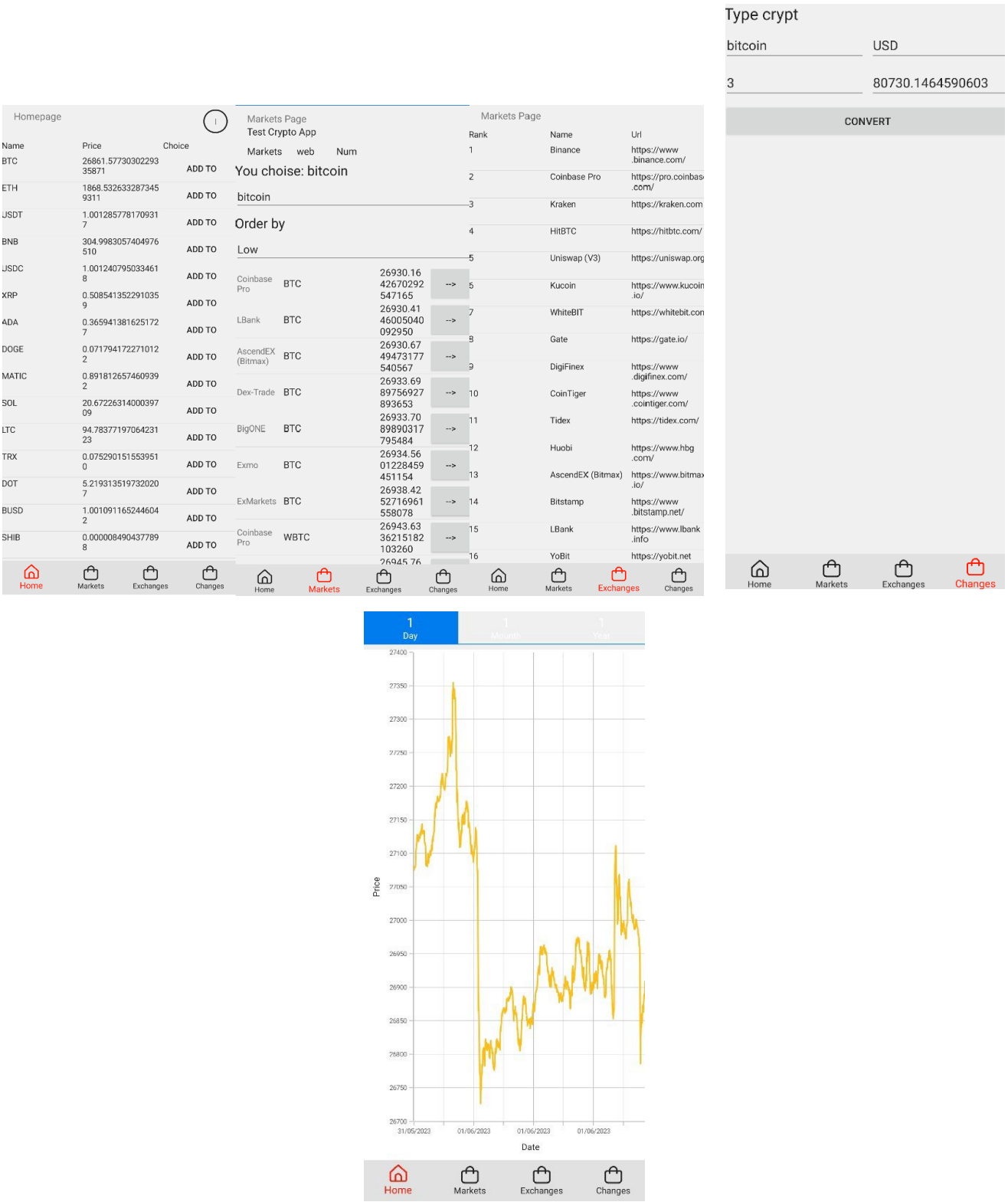

Рис.3.11. *а* – Home; *б* – Markets; *в* – Exchanges; *г* – Changes; *д* – Charts

Окрім чотирьох основних сторінок, додаток також містить додаткові функціональні розділи, які забезпечують користувачам персоналізований досвід

використання. До них належать Сторінка входу, Сторінка реєстрації, Сторінка "Вибране" та Сторінка налаштувань.

Сторінка входу відіграє ключову роль у забезпеченні доступу до персоналізованого функціоналу додатку. Вона надає користувачам можливість увійти до свого облікового запису, використовуючи збережені реєстраційні дані. Якщо користувач ще не має акаунту, він може одразу перейти до процесу створення нового облікового запису. Це дозволяє забезпечити збереження індивідуальних налаштувань, список обраних криптовалют та історію взаємодії з додатком.[4]

Сторінка реєстрації створена для нових користувачів, які бажають створити власний обліковий запис. Вона містить необхідні поля для введення персональних даних, таких як ім’я, електронна пошта, пароль та інші параметри, що можуть бути використані для ідентифікації та входу в систему. Після успішної реєстрації користувач отримує доступ до персоналізованого контенту та можливості синхронізації своїх даних на різних пристроях.

Сторінка "Вибране" надає користувачам можливість формувати власний список улюблених криптовалют. Це значно спрощує моніторинг змін вартості вибраних активів, дозволяючи користувачеві швидко переглядати інформацію про ті криптовалюти, які становлять для нього найбільший інтерес. Завдяки цій функції не потрібно щоразу переглядати весь ринок — обрані активи будуть доступні в окремому розділі для швидкого аналізу.

Сторінка налаштувань забезпечує гнучкість у використанні додатку, дозволяючи користувачам змінювати параметри інтерфейсу та функціональності відповідно до власних потреб. Вона може включати можливість вибору темного або світлого режиму відображення, зміни параметрів оновлення даних, налаштування сповіщень про зміну вартості криптовалют та інші корисні опції, що підвищують комфортність роботи з додатком.

Таким чином, ці додаткові сторінки розширюють можливості користувачів, дозволяючи їм не лише отримувати інформацію про криптовалютний ринок, а й персоналізувати взаємодію з додатком, зберігати

важливі активи у "Вибраному" та керувати загальними параметрами роботи системи.

Додаткові сторінки зображені на рис.3.12

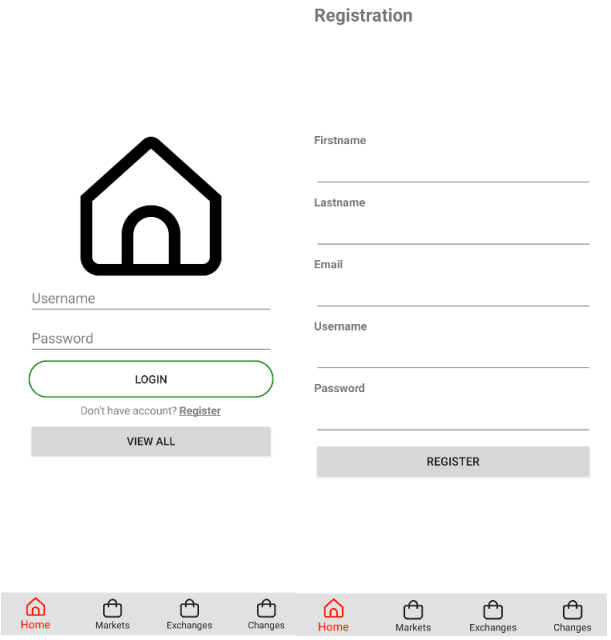

Рис.3.12. *а* – Login; *б* – Registration;

ВИСНОВКИ

У процесі виконання бакалаврської кваліфікаційної роботи на тему «Створення мобільного додатку для моніторингу ринку криптовалют» було реалізовано повний цикл прикладного дослідження – від вивчення теоретичних аспектів і аналізу ринку до створення функціонального програмного продукту. Основною метою проєкту було створення ефективного інструменту для оперативного доступу до ринкової інформації про криптовалюти, що є надзвичайно актуальним у контексті розвитку фінансових технологій та зростання популярності цифрових активів.

На першому етапі роботи було здійснено комплексний аналіз сучасного стану криптовалютного ринку. Було встановлено, що цей сегмент характеризується високою волатильністю, динамічним оновленням інформації та великою кількістю інструментів, які постійно вдосконалюються. Особливу увагу було приділено вивченню інформаційних потреб користувачів, зокрема інвесторів і трейдерів, які працюють із цифровими активами. Було з’ясовано, що мобільність, інтерактивність, точність даних та безпека є ключовими вимогами до додатків, які обслуговують ринок криптовалют.

У наступному етапі було проведено аналіз існуючих мобільних рішень, серед яких найбільш популярними виявилися Blockfolio, CoinMarketCap та Delta. У процесі їх дослідження було виявлено як сильні сторони, так і недоліки, серед яких надмірна складність інтерфейсів, недостатня гнучкість налаштувань, відсутність підтримки маловідомих активів або обмеження у безкоштовних версіях. Цей аналіз став основою для формування чітких технічних і функціональних вимог до розроблюваного додатку, які мали забезпечити конкурентоспроможність нового продукту.

У якості програмного інструментарію було обрано мову C# та платформу Xamarin, що дозволяє створювати кросплатформені застосунки з використанням єдиного кодового базису. Це рішення дало змогу ефективно реалізувати мобільний додаток для Android з перспективою масштабування на інші операційні системи. Використання Visual Studio як інтегрованого середовища розробки забезпечило комфортну роботу з кодом, а також надало

змогу швидко налагоджувати, тестувати й виводити застосунок на пристрої.

У процесі реалізації програмного продукту було створено адаптивний та інтуїтивно зрозумілий інтерфейс, побудований із використанням патерну MVVM. Це дозволило розділити логіку бізнес-процесів і відображення даних, що значно підвищило масштабованість, модульність і підтримуваність коду. Архітектура додатку також передбачає використання локальної бази даних SQLite для зберігання персональних налаштувань користувача, вподобаних активів та іншої допоміжної інформації. Для взаємодії з біржами криптовалют додаток використовує API, що забезпечує доступ до актуальної ринкової інформації в реальному часі.

Ключовими функціями створеного мобільного застосунку стали відображення актуальних курсів криптовалют, перегляд історичних графіків, створення персонального інвестиційного портфеля, налаштування повідомлень про зміну вартості активів, інтерактивна візуалізація даних та підтримка темного режиму для зручності використання у будь-яких умовах освітлення. Окремо було реалізовано можливість сортування активів за різними критеріями: обсягом торгів, динамікою цін, ринковою капіталізацією, що дозволяє користувачу ефективно фільтрувати та аналізувати інформацію.

У ході розробки значну увагу було приділено безпеці. Було впроваджено базові механізми автентифікації, зашифроване зберігання даних, захист від перехоплення запитів. Завдяки цьому користувач може бути впевнений у конфіденційності та захищеності власної інформації, що особливо важливо в контексті роботи з фінансовими сервісами.

Розроблений додаток успішно виконує всі функції, закладені в технічному завданні. У ході тестування було підтверджено його стабільність, швидкодію, коректне відображення інформації та зручність використання. Отримані результати свідчать про доцільність обраного підходу до архітектури, дизайну та програмної реалізації. Завдяки кросплатформеності рішення, у майбутньому можливе швидке розширення функціоналу та вихід на інші платформи без суттєвих витрат ресурсів.

У межах роботи також було визначено перспективні напрямки

подальшого вдосконалення застосунку. Зокрема, йдеться про інтеграцію модуля новин з аналітичних ресурсів, впровадження інструментів технічного аналізу на графіках, розробку механізму автоматичного оновлення портфеля, підтримку push-сповіщень, а також розширення функціоналу з урахуванням можливості взаємодії з NFT-активами або DeFi-сервісами. Усі ці можливості можуть бути поступово реалізовані на наступних етапах розвитку проєкту.

Таким чином, виконана бакалаврська робота є прикладом повноцінного проєктного підходу до вирішення актуального прикладного завдання в галузі інформаційних технологій. Вона демонструє здатність автора самостійно проводити аналітичне дослідження, формулювати вимоги до програмного продукту, приймати технічні рішення щодо вибору інструментарію, створювати програмну архітектуру, реалізовувати і тестувати програмне забезпечення, оцінювати його ефективність та обґрунтовувати напрями подальшого розвитку.

Загалом, створений мобільний додаток повністю відповідає сучасним вимогам до інтерфейсів, технічної реалізації та користувацького досвіду, що дозволяє використовувати його як базову платформу для комерційного використання, подальших досліджень або розширення в рамках більш масштабного проєкту. Робота відповідає критеріям бакалаврської кваліфікації, має практичне значення та свідчить про високий рівень професійної підготовки здобувача.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Боровик Т. І. Розробка мобільних застосунків. Київ: КНТ, 2017. 320 с.

2. Голубовський О. В. Мобільна аналітика: методи та інструменти. Львів: Вид-во ЛНУ, 2018. 256 с.

3. Дмитрук П. В. Фінансові мобільні додатки: практичний підхід. Харків: Фінансовий Університет, 2019. 288 с.

4. Журавська Л. П. Інтерфейси мобільних додатків. Київ: Наукова думка, 2020. 304 с.

5. Козак В. М. Основи криптографії в мобільних додатках. Одеса: ОНУ, 2021. 272 с.

6. Левченко Є. С. Аналітика даних у реальному часі. Дніпро: SAG, 2017. 240 с.

7. Мельник І. П. Криптовалютний ринок: теорія і практика. Київ: Либідь, 2018. 312 с.

8. Нечипорук Р. М. Blockchain: мобільна інтеграція. Харків: Сталіон, 2019. 296 с.

9. Павленко Т. О. UX/UI для фінтех-додатків. Львів: Центр навчання, 2020. 224 с.

10. Романчук С. Б. Безпека мобільних додатків. Київ: ДІТ, 2022. 264 с.

11. Савченко О. В. Мобільні технології та IoT. Полтава: ПНТУ, 2017. 280 с.

12. Ткаченко Н. М. Аналітичні платформи для фінансів. Дрогобич: Коло, 2018. 256 с.

13. Устименко Ю. В. Cryptocurrency exchange systems. Київ: Ліра-К, 2021. 320 с.

14. Федоренко А. І. Big data у фінансових додатках. Запоріжжя: ЗНУ, 2020. 288 с.

15. Цимбалюк М. О. Архітектура мобільних платформ. Київ: КНЕУ, 2019. 304 с.

16. Чорненко І. С. Машинне навчання в мобільних застосунках. Львів: Центр, 2022. 240 с.

17. Шевчук К. Д. Фінансові технології в Україні. Вінниця: НОВА, 2018. 272 с.

18. Якимчук О. П. Сучасні методи збору даних у мобільному середовищі. Київ: Артема, 2021. 256 с.

19. Антонюк Г. М. Розподілені системи для крипторинку. Луцьк: ТНУ, 2022.

248 с.

20. Борисенко В. Р. Аналітичне представлення фінансових даних. Ужгород: УжНУ, 2019. 280 с.

21. Гнатюк Т. І. Програмування мобільних додатків на Android та iOS. Одеса: ОНУ, 2016. 320 с.

22. Кондратюк С. В. Аналіз архітектурних патернів у мобільних крипто-додатках. Журнал прикладної інформатики. 2019. № 3. С. 45–60.

23. Лазаренко О. Е. Використання REST API для отримання курсів криптовалют. Інформатика і сучасність. 2021. № 2. С. 112–125.

24. Мамонова Н. В. Безпекові загрози у мобільних фінтех-додатках. Безпека інформаційних систем. 2020. № 1. С. 23–35.

25. Нікіфоров А. К. Оптимізація UI/UX мобільних додатків для торгівлі криптовалютами. Вісник НТУ. 2018. № 4. С. 98–110.

26. Павлюк Л. М. Гейміфікація у фінансових мобільних застосунках. Фінанси та інформ. технології. 2022. № 1. С. 55–69.

27. Семенчук О. Д. Аналітика ринку криптовалют у мобільних пристроях. Електронний вісник КНУ. 2021. № 5. С. 78–90.

28. Тарасюк І. О. Застосування WebSocket для live-даних крипторинку. Комп’ютерні системи і мережі. 2020. № 3. С. 135–147.

29. Удовенко В. Г. Моделі прогнозування курсу криптовалют. Прикладна математика і програмування. 2019. № 2. С. 67–82.

30. Філіпенко Г. П. Інтеграція блокчейн-сервісів у мобільні додатки. Сучасні проблеми інформатики. 2018. № 6. С. 24–36.

31. Харченко М. С. Використання кешування для швидкого відображення даних на мобільних платформах. ІТ-вектор. 2017. № 4. С. 18–29.

32. Шаповалова І. А. Забезпечення доступності мобільних крипто-застосунків. Accessible IT. 2022. № 2. С. 40–52.

33. Ярмолюк О. В. Синхронізація даних у реальному часі між сервером та мобільним додатком. Телекомунікаційні системи. 2021. № 1. С. 104–118.

34. Огляд популярних API для криптобірж. FinTech-UA. URL: https://fintech-ua.com/apis-crypto (дата звернення: 01.06.2025).

35. Коваленко С. Як створити мобільний додаток з відображенням криптовалют в реальному часі. DevUA. URL: https://devua.io/article/mobile-crypto-app (дата звернення: 12.05.2025).
36. Топ-5 SDK для взаємодії з WebSocket у Android. IT-Expert. URL: https://it-expert.ua/ws-sdk-android (дата звернення: 20.04.2025).
37. Петренко Л. П. Архітектурні рішення для фінтех-мобільних застосунків. FinDesign. URL: https://findesign.com/fintech-architecture (дата звернення: 03.03.2025).
38. Реалізація клієнта API Binance: приклад на Kotlin. StackUA. URL: https://stackua.com/kotlin-binance (дата звернення: 15.02.2025).
39. Іваненко М. М. Оцінка швидкодії мобільних криптовалютних додатків. MobileTest. URL: https://mobiletest.ua/crypto-perf (дата звернення: 28.01.2025).
40. Використання Firebase для оновлення ринку криптовалют у реальному часі. CloudUA. URL: https://cloudua.com/firebase-crypto (дата звернення: 05.01.2025).
41. Шевчук О. О. Забезпечення безпеки при роботі з криптовалютами на мобільних пристроях. CyberUA. URL: https://cyberua.com/mobile-crypto-security (дата звернення: 22.12.2024).
42. GraphQL vs REST для криптовалютних додатків. APIReview. URL: https://apireview.ua/graphql-crypto (дата звернення: 10.12.2024).
43. Гончаренко І. І. Порівняння UI-фреймворків для мобільних крипто-додатків. UI-Lab. URL: https://uilab.com/mobile-ui-crypto (дата звернення: 01.11.2024).