ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»
Факультет автоматизації виробництва, інформаційних
та управлінських технологій
Кафедра інформаційних технологій та аналітики даних

«Допущено до захисту»
Гарант ОПП
Ірина ГЕТЬМАН

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавра

за підсумками виконання
освітньо-професійної програми
«Комп'ютерні науки»
за спеціальністю 122 Комп`ютерні науки

на тему «Програмно методичний комплекс для аналізу рідкісності та ринкової вартості колекційних видань відеоігор»

Керівник роботи Ірина ГЕТЬМАН

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

Здобувач Микола ОЛІЙНИК

Підсумкова оцінка за атестацію			

Голова ЕК Антон КУДРЯВЦЕВ

ЗАПОРІЖЖЯ 2026

ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА»

Факультет	автоматизації виробництва, інформаційних та управлінських технологій
Кафедра	інформаційних технологій та аналітики даних
Ступінь вищої освіти	бакалавр
Спеціальність	122 Комп'ютерні науки
ОПП	Комп'ютерні науки

ЗАТВЕРДЖУЮ
Гарант ОПП

_____ Ірина ГЕТЬМАН

«26» лютого 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Олійнику Миколі Андрійовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема роботи *«Програмно-методичний комплекс для аналізу рідкісності та ринкової вартості колекційних видань відеоігор»*
керівник роботи, Гетьман Ірина Анатоліївна, доцент, канд. техн. наук
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом Університету від *23.02.2026 р. №41/23.02.2026*
2. Термін подання роботи *16.06.2026 р.*
3. Вихідні дані до роботи *Навчальна література, державні стандарти та методичні вказівки з дисциплін «Системний аналіз», «Інженерія програмного забезпечення», «Бази даних», «Управління IT-проєктами»; офіційна документація з розробки кросплатформених додатків мовою C# та на базі фреймворку .NET MAUI; технічні специфікації локальної СУБД SQLite та micro-ORM sqlite-net-pcl; ринкові маркери та історичні цінові дані з веб-платформи PriceCharting.com; стандарти грейдингу та оцінки фізичного стану відеоігор організацій WATA Games та VGA; результати власного об'єктно-орієнтованого проєктування, модульного тестування за допомогою фреймворку MSTest та техніко-економічного обґрунтування розробки.*
4. Зміст пояснювальної записки (перелік питань) *Реферат. Зміст. Вступ. Розділ 1. Аналіз предметної області оцінки рідкісності та ринкової вартості колекційних видань відеоігор. Розділ 2. Розробка моделі предметної області аналізу рідкісності та вартості колекційних видань відеоігор. Розділ 3. Проєктування та реалізація програмно-методичного комплексу аналізу рідкісності та вартості відеоігор. Розділ 4. Економічне обґрунтування розробки та використання програмно-методичного комплексу. Загальні висновки. Перелік використаних джерел. Додатки*
5. Перелік графічного (демонстраційного) матеріалу (з точним зазначенням обов'язкових креслень): *Актуальність, мета, об'єкт, предмет та завдання дослідження; ієрархія факторів ціноутворення та шкала оцінки фізичного стану відеоігор; логічна структура бази даних та ER-схема зв'язку 1:N між таблицями у СУБД SQLite; блок-схема детермінованого алгоритму обчислювального ядра PricingEngine; блок-схема алгоритму асинхронного веб-парсингу ринкових маркерів; діаграми UML (прецедентів, класів, послідовностей та розгортання); екранні форми*

користувацького інтерфейсу системи RareScore (MainPage, AddGamePage); приклади згенерованих аналітичних звітів у форматі PDF; результати розрахунків собівартості розробки та терміну окупності програмного комплексу; загальні висновки до роботи.

6. Консультанти по роботі, із зазначенням розділів роботи, що стосуються їх.

Розділ	Прізвище, ініціали та посада консультанта
1	Гетьман І.А., канд. техн. наук, доц. каф. ІНТЕХАД
2	Гетьман І.А., канд. техн. наук, доц. каф. ІНТЕХАД
3	Гетьман І.А., канд. техн. наук, доц. каф. ІНТЕХАД
4	Гетьман І.А., канд. техн. наук, доц. каф. ІНТЕХАД

7. Дата видачі завдання *26.02.2026*

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи
1	Розділ 1. Аналіз предметної області оцінки рідкісності та ринкової вартості колекційних видань відеоігор	26.02.2026 - 22.03.2026
2	Розділ 2. Розробка моделі предметної області аналізу рідкісності та вартості колекційних видань відеоігор	23.03.2026 - 05.04.2026
3	Розділ 3. Проєктування та реалізація програмно-методичного комплексу аналізу рідкісності та вартості відеоігор	06.04.2026 – 30.04.2026
5	Розділ 4. Економічне обґрунтування розробки та використання програмно-методичного комплексу	01.05.2026 - 08.05.2026
6	Висновки, перелік посилань, вступ, зміст, реферат	09.05.2026 – 25.02.2026
7	Подання завершеної роботи. Перевірка на академічний плагіат	26.05.2026 – 16.06.2026
8	Остаточне оформлення роботи, презентаційного матеріалу	16.06.2026 – 18.06.2026
9	Рецензування завершеної роботи. Захист	18.06.2026 – 23.06.2026

Здобувач (Микола ОЛІЙНИК)

Керівник роботи (Ірина ГЕТЬМАН)

РЕФЕРАТ

Олійник М. А. Програмно-методичний комплекс для аналізу рідкісності та ринкової вартості колекційних видань відеоігор. Кваліфікаційна робота на здобуття освітнього ступеня бакалавра комп'ютерних наук за спеціальністю 122 «Комп'ютерні науки». – ТОВ «ТЕХНІЧНИЙ УНІВЕРСИТЕТ «МЕТІНВЕСТ ПОЛІТЕХНІКА» МОН України, м. Київ, 2026.

Метою роботи було створення програмно-методичного комплексу «RareScore», який дозволить автоматизувати процеси каталогізації, грейдингу та прогнозування ринкової вартості колекційних видань відеоігор на основі інтегрованої моделі ціноутворення.

Об'єктом дослідження є процес ціноутворення та оцінки рідкісності на вторинному ринку фізичних носіїв відеоігор.

Предметом дослідження є математичні моделі, алгоритми та програмні засоби для автоматизованого аналізу ринкової вартості колекційних видань відеоігор з урахуванням їх технічного стану та характеристик комплектації.

У цій роботі проаналізовано специфіку вторинного ринку фізичних носіїв відеоігор, критерії стану примірників (Loose, CIB, Sealed) та існуючі системи незалежного грейдингу (WATA Games, VGA). Спроєктовано архітектуру системи та побудовано її логічну й інформаційну моделі. Програмний продукт «RareScore» реалізовано мовою C# із використанням кросплатформеного фреймворку .NET MAUI у середовищі Visual Studio та локальної бази даних SQLite. Реалізовано обчислювальне ядро PricingEngine для детермінованого розрахунку вартості з урахуванням покомпонентних штрафів за некомплектність та премій за грейд, асинхронний парсер маркерів з PriceCharting.com, підсистему сканування штрих-кодів у реальному часі через ZXing.Net.MAUI та конвеєр генерації PDF-звітів засобами QuestPDF. Було ведено модульне тестування алгоритмів та проведено техніко-економічне обґрунтування впровадження розробленого комплексу.

Розроблене програмне забезпечення дозволяє суттєво підвищити продуктивність оцінки лотів (прискорення у 20 разів), мінімізувати фінансові ризики через помилки оцінювання, забезпечити ведення локальної аналітики трендів (LiveChartsCore) та оптимізувати процеси обліку активів. Комплекс придатний для використання як індивідуальними колекціонерами, так і комерційними ретейлерами ретро-ігор.

КЛЮЧОВІ СЛОВА: .NET MAUI, C#, SQLITE, КОЛЕКЦІЙНІ ВІДЕОІГРИ, АВТОМАТИЗОВАНЕ ОЦІНЮВАННЯ, ГРЕЙДИНГ WATA, PRICING ENGINE, АСИНХРОННИЙ ПАРСИНГ, ЕКОНОМІЧНА ДОЦІЛЬНІСТЬ

ABSTRACT

Oliynyk M. A. Software and methodical complex for analyzing the rarity and market value of collectible video game editions. Qualification work for obtaining the degree of Bachelor of Computer Science in the specialty 122 Computer Science. – LLC «TECHNICAL UNIVERSITY «METINVEST POLYTECHNIC» MES of Ukraine, Kyiv, 2026.

The purpose of the work was to create the "RareScore" software and methodical complex designed to automate the processes of cataloging, grading, and forecasting the market value of collectible video games based on an integrated pricing model.

The object of the study is the process of pricing and rarity evaluation in the secondary market of physical video game media.

The subject of the study is mathematical models, algorithms, and software tools for the automated analysis of the market value of collectible video games, considering their physical condition and completeness characteristics.

In this work, the specifics of the secondary physical video game market, item condition criteria (Loose, CIB, Sealed), and existing independent grading systems (WATA Games, VGA) are analyzed. The architecture is designed, and the logical and informational models of the system are constructed. The "RareScore" software is implemented in the C# programming language using the cross-platform .NET MAUI framework within the Visual Studio environment and a local SQLite database. An isolated computational PricingEngine was developed to calculate the estimated value considering completeness penalties and WATA grading premiums. Additional implemented modules include asynchronous web parsing from PriceCharting.com, a real-time barcode scanning subsystem via ZXing.Net.MAUI, and a declarative PDF report generation pipeline using QuestPDF. Unit testing of the algorithms and a feasibility study for the implementation of the developed complex have been conducted.

The developed software significantly increases evaluation productivity (20x acceleration), minimizes financial risks caused by appraisal errors, provides local analytical trend tracking (LiveChartsCore), and optimizes asset inventory processes. The complex is suitable for individual collectors and commercial retro game retailers.

KEYWORDS: .NET MAUI, C#, SQLITE, COLLECTIBLE VIDEO GAMES, AUTOMATED APPRAISAL, WATA GRADING, PRICING ENGINE, ASYNCHRONOUS PARSING, ECONOMIC FEASIBILITY

ЗМІСТ

ВСТУП

Ринок колекційних видань відеоігор є динамічним і самостійним сегментом глобального ринку культурних та розважальних товарів. За даними аналітичної компанії Wata Games та платформи Heritage Auctions, у період 2018–2024 років зафіксовано стрімке зростання транзакційної активності: окремі екземпляри колекційних відеоігор були реалізовані на аукціонах за ціною, що перевищує 500 000 доларів США, а запечатаний примірник Super Mario Bros. для NES з оцінкою WATA 9.8 A++ досяг позначки 2 000 000 доларів у серпні 2021 року. Водночас відсутність стандартизованих автоматизованих інструментів для об'єктивної оцінки ринкової вартості та рідкісності залишається актуальною проблемою як для індивідуальних колекціонерів, так і для комерційних посередників.

Існуючі програмні рішення – PriceCharting, VGPC, CLZ Games – вирішують лише окремі аспекти задачі: переважно каталогізацію та моніторинг ринкових цін. Жодне з них не забезпечує комплексного аналізу з урахуванням грейдингових характеристик, типу видання та повноти комплектації. Це зумовлює необхідність розробки спеціалізованого програмно-методичного комплексу, що інтегрує алгоритмічну модель ціноутворення з аналітичними та звітними функціями.

Актуальність теми визначається: відсутністю програмних засобів, що автоматизують комплексну оцінку вартості колекційних відеоігор з урахуванням усіх релевантних факторів; зростаючим попитом на об'єктивні інструменти підтримки інвестиційних рішень у сфері колекціонування; потребою у стандартизації процесу оцінювання відповідно до систем грейдингу WATA та VGA.

Мета роботи: розробка та практична реалізація програмно-методичного комплексу RareScore для автоматизованого аналізу

рідкісності та ринкової вартості колекційних видань відеоігор на основі інтегрованої моделі ціноутворення.

Для досягнення поставленої мети визначено такі завдання:

– дослідити сучасний стан ринку колекційних видань відеоігор та виявити основні фактори, що визначають їх ринкову вартість;

– провести критичний аналіз існуючих програмних засобів та математичних моделей для оцінки вартості, виявити їх недоліки;

– розробити математичну модель (Pricing Engine) для розрахунку вартості колекційного видання на основі системи зважених коефіцієнтів;

– спроєктувати та реалізувати архітектуру програмного комплексу із застосуванням платформи .NET MAUI, гібридного підходу MVVM + Code-Behind та локальної бази даних SQLite з Micro-ORM sqlite-net-pcl;

– інтегрувати підсистему сканування штрих-кодів EAN/UPC на основі бібліотеки ZXing.Net.MAUI;

– реалізувати підсистему аналітики та підсистему генерації звітів у форматах PDF та CSV;

– провести тестування комплексу та верифікацію результатів оцінювання на реальних ринкових даних.

Об'єкт дослідження: процес оцінки ринкової вартості та рідкісності колекційних фізичних видань відеоігор.

Предмет дослідження: математичні моделі, алгоритми та програмні засоби для автоматизованого аналізу ринкової вартості колекційних видань відеоігор з урахуванням їх технічного стану та характеристик комплектації.

Практична цінність роботи полягає у створенні функціонального десктопного додатку RareScore, придатного для використання колекціонерами та ретейлерами в якості інструменту підтримки прийняття рішень при купівлі-продажу та оцінці колекцій.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ОЦІНКИ РІДКІСНОСТІ ТА РИНКОВОЇ ВАРТОСТІ КОЛЕКЦІЙНИХ ВИДАНЬ ВІДЕОІГОР

Перший розділ присвячено комплексному дослідженню предметної галузі системи RareScore. Розглядається поточний стан ринку колекційних відеоігор, аналізуються фактори ціноутворення, системи незалежного грейдингу та критерії фізичного оцінювання. Завершується розділ критичним аналізом наявних програмних засобів та постановкою задачі розробки.

1.1 Аналіз сучасного стану ринку колекційних відеоігор та систем їх грейдингу

Ринок колекційних фізичних видань відеоігор є самостійним сегментом вторинного ринку культурних цінностей. На відміну від первинного ринку програмного забезпечення, він функціонує переважно через спеціалізовані аукціонні платформи (Heritage Auctions, eBay), агрегатори цін (PriceCharting, VGPC) та дилерські мережі (DKOldies, Lukie Games).

Сегмент колекційних відеоігор демонструє значну цінову волатильність, зумовлену спекулятивним попитом та обмеженою пропозицією запечатаних (Sealed) примірників. Протягом 2020–2021 років спостерігалося пікове зростання транзакційної активності, обумовлене рейтинговою кампанією Wata Games та медійним висвітленням рекордних аукціонів. Після 2022 року ринок перейшов у фазу консолідації з тенденцією до стабілізації цін у масовому сегменті при збереженні преміальних оцінок на топові лоти.

Основними сегментами ринку є: (1) масовий – серійні видання, вартість яких корелює з тиражем і станом збереження; (2) преміальний – ліцензійні та колекторські видання (Collector's Edition, Limited Edition) з ексклюзивним контентом; (3) топ-сегмент – запечатані примірники з незалежним грейдом WATA або VGA категорії «gem mint» (9.4–10.0). Один і той самий тайтл може відрізнятися за вартістю на кілька порядків залежно від сегмента.

Ключовою платформою моніторингу ринкових цін є PriceCharting.com, що агрегує дані реалізованих транзакцій з eBay та інших майданчиків. База даних PriceCharting станом на 2024 рік охоплює понад 300 000 найменувань для більш ніж 100 ігрових платформ. Саме ця платформа використовується як джерело базових ринкових цін у Pricing Engine комплексу RareScore.

1.1.1 Система грейдингу WATA

Незалежний грейдинг – це процедура сертифікації колекційного видання спеціалізованою організацією, результатом якої є числова оцінка стану та автентичності примірника із запечатуванням у захисний акриловий футляр (slab). Грейдинг є обов'язковою умовою участі в аукціонних торгах у категорії «Sealed» на таких майданчиках, як Heritage Auctions та Rally.

Wata Games (засн. 2018 р., США) – провідна організація з грейдингу відеоігор. Застосовує двокомпонентну систему оцінки: числова оцінка за шкалою від 1.0 до 10.0 з кроком 0.5 та літерна оцінка цілісності заводського сілу (A++, A+, A, B+, B, C+, C). Оцінка 9.8 A++ є медіанною для топових аукціонних лотів, тоді як оцінки 10.0 формують найвищий

ціновий сегмент з мультиплікатором відносно 9.4 у діапазоні 2x–10x залежно від тайтлу.

1.1.2 Система грейдингу VGA

Video Game Authority (VGA) (засн. 2009 р., США) використовує виключно числову шкалу: VGA 85 (NM), VGA 90 (NM+), VGA 95 (NM/MT), VGA 100 (MT). VGA застосовує різні еталони оцінювання залежно від декади випуску гри та умови – «Uncirculated» для запечатаних, «Gold» для відкритих примірників. WATA домінує для постатарівських платформ (NES, SNES, N64, Genesis), тоді як VGA зберігає значну частку у сегменті Atari та ранніх PC-видань.

У Pricing Engine комплексу RareScore наявність верифікованого грейду реалізована як параметр з можливістю введення числового значення оцінки (поле WataGrade типу REAL, NULLABLE). Мультиплікативний коефіцієнт грейду застосовується виключно для видань у стані Sealed, що відповідає ринковій практиці.

1.2 Аналіз критеріїв оцінки рідкісності та фізичного стану колекційних видань відеоігор

Ціна конкретного примірника колекційного видання відеогри є функцією від сукупності кількісних та якісних параметрів. Аналіз транзакційних даних PriceCharting та Heritage Auctions дозволяє виокремити ієрархію факторів ціноутворення, представлену в таблиці 1.1.

Таблиця 1.1 – Фактори ціноутворення колекційних видань відеоігор

Фактор	Категорія	Вплив на вартість	Коефіцієнт у моделі
Базова ринкова ціна	Ринковий	Визначальний (база)	1.0 (базова)
Тип видання	Продуктовий	Суттєвий	до ×2.5 для Collector's Edition
Стан збереження (Condition)	Технічний	Визначальний	0.40–1.00 від бази
Наявність грейду (WATA/VGA)	Сертифікаційний	Суттєвий при Sealed	до ×1.50 залежно від балу
Повнота комплектації	Технічний	Значний	до ×1.30 за повної комплектації
Платформа та регіон випуску	Ринковий	Значний для ексклюзивів	визначається базовою ціною
Тираж та рідкісність	Ринковий	Суттєвий для лімітованих серій	визначається базовою ціною

Базова ринкова ціна (base market price) відображає медіану реалізованих транзакцій для конкретного тайтлу та платформи в умові Loose (картридж або диск без коробки та документації). Парсинг цього значення з PriceCharting є першим кроком Pricing Engine.

Стан збереження (Condition). Застосовується трирівнева класифікація: Sealed – фабричне пакування не розкривалось; CIB (Complete In Box) – наявна коробка, картридж/диск та всі документи; Loose – лише носій без коробки та документації. Різниця вартості між категоріями Sealed та Loose може варіюватися від 2x до 20x.

Повнота комплектації для CIB-видань включає наявність артбуку, стілбуку, мануалу, реєстраційної картки та рекламних вкладок. Відсутність

будь-якого з цих елементів фіксується ринком як «incomplete» з відповідним дисконтом 5–25%.

Таблиця 1.2 – Дескриптивна шкала оцінки фізичного стану колекційних видань відеоігор

Ступінь стану	Опис	Типові дефекти	Діапазон від Loose-базової
Mint (M)	Ідеальний стан, як нове	Відсутні	100–120%
Near Mint (NM)	Мінімальні сліди використання	Незначні подряпини на шрифтці	85–100%
Very Good Plus (VG+)	Очевидне дбайливе використання	Подряпини, незначне стирання	65–85%
Very Good (VG)	Помірні сліди використання	Помітні подряпини, часткове стирання	45–65%
Good (G)	Значні косметичні дефекти	Глибокі подряпини, ушкодження етикетки	25–45%
Poor (P)	Граничний функціональний стан	Тріщини корпусу, відсутність елементів	10–25%

Зазначені цінові діапазони є апроксимативними та характерними для масового сегменту. Для рідкісних та ліцензійних видань переходи між категоріями стану можуть мати значно вищі мультиплікативні ефекти. У практиці RareScore категорії стану картковані на внутрішні числові коефіцієнти відповідно до формули Pricing Engine.

1.3 Огляд існуючих програмних засобів для каталогізації та оцінки вартості відеоігор

Критичний аналіз наявних програмних засобів для обліку та оцінки колекційних відеоігор виявив ряд рішень, релевантних до предметної галузі.

PriceCharting.com є найбільшим веб-сервісом агрегації транзакційних цін: база даних охоплює понад 300 000 тайтлів для більш ніж 100 платформ. Проте сервіс надає лише три фіксовані цінові точки (Loose / CIB / Sealed) без можливості врахування стану конкретного примірника, грейду та повноти комплектації. Відсутня аналітика трендів та генерація PDF-звітів. Платформа є виключно веб-орієнтованою.

CLZ Games – десктопний та мобільний додаток для каталогізації та базової оцінки. Підтримує сканування штрихкодів та хмарну синхронізацію. Відсутня алгоритмічна модель ціноутворення з коефіцієнтами, немає підтримки WATA/VGA-грейдів та генерації PDF-звітів. Доступ до повного функціоналу вимагає платної підписки.

VGPC (Video Game Price Charting) пропонує сканування, моніторинг цін за категоріями Loose/CIB/Sealed та відстеження колекції. Gameye забезпечує базову каталогізацію та відображення загальної вартості колекції. Обидва рішення є виключно мобільними, що обмежує їх застосування у десктоп-орієнтованих сценаріях.

Таблиця 1.3 – Зведений порівняльний аналіз існуючих програмних засобів

Система	Тип	Переваги	Критичні недоліки
PriceCharting.com	Веб-сервіс	Найбільша БД цін (>300 000)	Немає розрахунку за станом/грейдом; нема офлайн; нема PDF
CLZ Games	Десктоп/мобільний	Зручний UI; багато платформ	Немає моделі з коефіцієнтами; не підтримує WATA/VGA; платна підписка
VGPC (мобільний)	Мобільний додаток	Швидке сканування; 3 цінових категорії	Немає деталізованої моделі; немає аналітики; немає PDF
Gameye	Мобільний додаток	Простота; безкоштовна базова версія	Немає моделі оцінки; немає підтримки WATA-грейдів
Excel / Google Sheets	Електронна таблиця	Гнучкість налаштування	Немає автопарсингу; немає PDF-звітів; висока трудомісткість

Проведений аналіз виявив системну прогалину у функціональному покритті існуючих рішень. Жодне з розглянутих рішень не реалізує одночасно: (1) автоматизованого парсингу базової ринкової ціни; (2) розрахунку вартості за системою зважених коефіцієнтів з урахуванням типу видання, стану, грейду та повноти комплектації; (3) підтримки стандартів WATA та VGA; (4) локальної аналітики трендів і KPI; (5) генерації структурованих PDF-звітів; (6) автономної роботи без постійного

підключення до мережі; (7) інтегрованого сканера штрих-кодів EAN/UPC для десктопного середовища.

1.4 Постановка задачі розробки програмно-методичного комплексу

На підставі проведеного аналізу предметної галузі та виявлених обмежень існуючих програмних рішень сформульовано постановку задачі розробки програмно-методичного комплексу RareScore.

Задачу розробки сформульовано таким чином: необхідно створити десктопний програмно-методичний комплекс RareScore, який реалізує: (1) автоматизований парсинг базових ринкових цін з платформи PriceCharting.com; (2) алгоритмічну модель ціноутворення (Pricing Engine) на основі мультиплікативно-адитивної формули з повним набором коефіцієнтів (стан, тип видання, комплектність, грейд); (3) сканування штрих-кодів EAN/UPC з камери пристрою через бібліотеку ZXing.Net.MAUI; (4) локальну базу даних для автономної роботи; (5) підсистему аналітики з побудовою графіків динаміки цін та розрахунком KPI; (6) функцію генерації структурованих PDF-звітів.

2 РОЗРОБКА МОДЕЛІ ПРЕДМЕТНОЇ ОБЛАСТІ АНАЛІЗУ РІДКІСНОСТІ ТА ВАРТОСТІ КОЛЕКЦІЙНИХ ВИДАНЬ ВІДЕОІГОР

Другий розділ присвячено розробці моделі предметної галузі та обґрунтуванню технологічних рішень. Сформовано глосарій ключових термінів, обґрунтовано вибір технологічного стеку, описано гібридну архітектурну модель, розроблено математичну модель Pricing Engine, спроєктовано логічну та інформаційну моделі програмного комплексу, а також сформульовано технічне завдання на розробку.

2.1 Обґрунтування вибору методів дослідження та програмного забезпечення

Вибір технологій для розробки комплексу RareScore здійснювався на основі аналізу функціональних і нефункціональних вимог. Основними критеріями відбору були: надійність і зрілість екосистеми, відповідність прагматичній гібридній архітектурі, можливість локальної роботи з даними без хмарної інфраструктури та наявність підтримки необхідних функцій – парсинг, сканування штрих-кодів, візуалізація, генерація PDF.

2.1.1 Платформа .NET MAUI та мова програмування C#

.NET Multi-platform App UI (.NET MAUI) – кросплатформний UI-фреймворк корпорації Microsoft [2], що є наступником Xamarin.Forms та входить до складу екосистеми .NET 8. Фреймворк забезпечує компіляцію

єдиної кодової бази у нативні додатки для платформ Windows, macOS, iOS та Android.

Для розробки RareScore платформу .NET MAUI обрано з таких технічних підстав. По-перше, Desktop-first архітектура: основним цільовим середовищем є Windows-десктоп; .NET MAUI надає повноцінну підтримку WinUI 3 як нативного UI-бекенду для Windows. По-друге, зрілість екосистеми .NET: мова C# та платформа .NET 8 забезпечують типобезпечне середовище виконання, розвинену систему NuGet-пакетів та повноцінну підтримку асинхронного програмування (async/await), що є критичним для реалізації неблокуючих HTTP-запитів у модулі парсингу. По-третє, кросплатформна перспектива: архітектурна сумісність з macOS та мобільними платформами є стратегічною перевагою для подальшого масштабування.

Таблиця 2.1 – Порівняльний аналіз альтернативних платформ для десктопної розробки

Платформа	Мова	Кросплатформність	Підтримка MVVM	Вибір
.NET MAUI (.NET 8)	C#	Windows, macOS, iOS, Android	Нативна	✓ Обрано
WPF	C#	Windows only	Нативна	Застаріла API
Electron.js	JS/TS	Windows, macOS, Linux	Стороння	Відхилено (надмірне RAM)
Flutter	Dart	Windows, macOS, Linux, Mobile	Provider/Bloc	Менш зріла екосистема

2.1.2 Гібридна архітектура: MVVM + Code-Behind (Event-Driven)

Архітектурне рішення для комплексу RareScore є прагматичним гібридним підходом, що поєднує два патерни: MVVM (Model-View-ViewModel) з прив'язкою даних через INotifyPropertyChanged – для підсистем аналітики та графіків, та Code-Behind (Event-Driven) – для всіх CRUD-форм і списків. Такий підхід відходить від механічного застосування «повного MVVM» на користь ефективності та відповідності реальній природі кожного компонента системи.

MVVM застосовується на сторінці MainPage (аналітика та KPI). AnalyticsViewModel реалізує INotifyPropertyChanged та експонує властивості типу IEnumerable<ISeries>, до яких прив'язані елементи CartesianChart і PieChart бібліотеки LiveChartsCore. При оновленні даних PriceSample у базі, ViewModel автоматично оновлює серії графіків через Data Binding без явного керування станом View.

Code-Behind (Event-Driven) застосовується для сторінок AddGamePage та CollectionPage. Вибір Event-Driven обґрунтовано трьома аргументами. По-перше, безпосередність обробки стану десятків чекбоксів: форма AddGamePage містить множину незалежних булевих прапорців (HasManual, HasArtbook, HasFigure, HasSteelbook, HasSoundtrack), стан яких потребує миттєвого оновлення UI та умовного відображення полів. Реалізація через Commands вносить надлишкові абстракції без функціонального виграшу. По-друге, прискорення UI: обробники подій виконуються синхронно в потоці UI. По-третє, спрощення коду: логіка вибору стану Condition і відображення панелей комплектації є суто UI-логікою.

Таблиця 2.2 – Розподіл архітектурних підходів за компонентами системи RareScore

Компонент / Сторінка	Архітектурний підхід	Обґрунтування вибору
MainPage (графіки, KPI)	MVVM (INotifyPropertyChanged)	Реактивне оновлення графіків LiveChartsCore через Data Binding; тестованість AnalyticsViewModel
AddGamePage (форма додавання)	Code-Behind (Event-Driven)	Миттєва обробка стану ~10 чекбоксів; умовне відображення полів без надлишкових Commands
CollectionPage (список каталогу)	Code-Behind (Event-Driven)	Швидка фільтрація та сортування списку; пряме оновлення ListView
PricingEngine (логіка розрахунку)	Model (чиста C#-логіка)	Повна відсутність UI-залежностей; ізольоване модульне тестування формули
PriceChartingParser (парсинг)	Model (async/await)	Неблокуючий HTTP-запит; незалежність від UI-шару

2.1.3 Система управління базою даних SQLite та Micro-ORM sqlite-net-pcl

Для локального зберігання даних колекції обрано реляційну СУБД SQLite у поєднанні з легкою бібліотекою об'єктно-реляційного

відображення sqlite-net-pcl (Micro-ORM) [1]. Вибір SQLite зумовлено serverless-архітектурою: база даних зберігає весь стан у єдиному файлі (rarescore_v4.db3), виключаючи залежність від хмарних серверів.

Класи предметної моделі (GameItem, PriceSample) анотуються атрибутами SQLite: [Table("GameItem")] пов'язує клас з таблицею; [PrimaryKey] та [AutoIncrement] на полі Id забезпечують автоматичну генерацію первинного ключа; [Ignore] виключає обчислювані властивості з маппінгу. Ініціалізація схеми виконується через асинхронний метод CreateTableAsync<T>(), що безпечно створює таблицю, якщо вона відсутня. На відміну від Entity Framework Core, що вимагає DbContext, реєстрацію в DI-контейнері та налаштування міграцій, підхід sqlite-net-pcl є суттєво простішим і забезпечує значно швидший холодний старт застосунку.

Таблиця 2.3 – Порівняльний аналіз підходів до доступу до даних у .NET MAUI

Характеристика	sqlite-net-pcl (Micro-ORM)	Entity Framework Core	Чистий ADO.NET
Складність налаштування	Мінімальна (атрибути)	Висока (DbContext, Migrations, DI)	Висока (ручний SQL)
Холодний старт	Швидкий	Повільний (рефлексія)	Дуже швидкий
Ініціалізація схеми	CreateTableAsync<T>()	Migrations (Add-Migration)	Ручний DDL SQL
Типобезпека запитів	LINQ (часткова)	LINQ (повна)	Відсутня
Потреба в DI-контейнері	Ні	Так (обов'язково)	Ні
Придатність для десктопу	Відмінна	Надмірна для малих схем	Прийнятна

2.1.4 Підсистема сканування штрих-кодів: ZXing.Net.MAUI

Підсистема сканування штрих-кодів реалізована на основі бібліотеки ZXing.Net.MAUI [3] – .NET MAUI-портуації відкритого рушія розпізнавання штрих-кодів ZXing (Zebra Crossing). Бібліотека забезпечує розпізнавання форматів EAN-13, EAN-8, UPC-A, UPC-E, Code 128, Code 39 та QR-кодів безпосередньо з відеопотоку камери пристрою.

Інтеграція реалізована через XAML-елемент CameraBarcodeReaderView, що передає розпізнані значення через подію BarcodeDetected. При отриманні EAN/UPC-коду система виконує HTTP GET-запит до зовнішнього API для автоматичного заповнення полів Title та Platform у формі AddGamePage.

Під час тестування модуля ZXing.Net.MAUI на декількох пристроях виникла характерна проблема: на деяких конфігураціях спостерігалася затримка ініціалізації камери від 1 до 3 секунд після виклику CameraBarcodeReaderView. Це потребувало оптимізації асинхронного виклику – зокрема, додаткового очікування готовності камери через механізм Task.Delay з подальшою повторною ініціалізацією. Через обмеження часу на практику повноцінне тестування на всьому спектрі цільових пристроїв не проводилось, тому зазначена поведінка фіксується як відома особливість, що потребує доопрацювання у фінальній версії.

2.1.5 Бібліотеки візуалізації та генерації звітів

Для підсистеми аналітики обрано бібліотеку LiveChartsCore (версія 2.x) [4] – відкриту .NET-нативну бібліотеку для побудови графіків, оптимізовану для .NET MAUI та WPF. Бібліотека надає XAML-елементи

CartesianChart та PieChart з повноцінною підтримкою Data Binding. Лінійні графіки (LineSeries) використовуються для відображення динаміки ринкової ціни тайтлу; стовпчасті (ColumnSeries) – для порівняльного аналізу вартості позицій колекції.

Функція генерації PDF-звітів реалізована за допомогою бібліотеки QuestPDF [5] – відкритого .NET-інструменту для програмної побудови PDF-документів із декларативним Fluent API, концептуально подібним до CSS Flexbox. Бібліотека iTextSharp відхилена через складну ліцензійну модель (AGPL), PdfSharp – через відсутність вбудованих засобів верстки таблиць та умовного форматування.

2.1.6 Глосарій ключових термінів предметної галузі

Таблиця 2.4 – Глосарій ключових термінів предметної галузі системи RareScore

Термін	Визначення
Екземпляр (примірник)	Конкретний фізичний об'єкт – картридж, диск або комплект у коробці – що є носієм відеогри й виступає одиницею обліку та оцінювання в системі RareScore.
Sealed (запечатаний)	Категорія стану, за якої оригінальне фабричне пакування залишається цілісним і нерозкритим. Обов'язкова умова для незалежного грейдингу.
CIB (Complete In Box)	«Повний комплект у коробці»: оригінальна коробка, носій та повна документація. Відсутність компонента фіксується дисконтом у моделі.
Loose	«Носій без упакування»: лише картридж або диск. Базова цінова категорія для Pricing Engine.

Продовження таблиці 2.4

Термін	Визначення
Грейдинг (Grading)	Незалежна сертифікація фізичного стану та автентичності видання організацією (WATA, VGA) із запечатуванням у акриловий футляр (slab).
WATA Games	Американська організація грейдингу відеоігор (засн. 2018). Двокомпонентна шкала: числова (1.0–10.0) + літерна (C–A++).
VGA (Video Game Authority)	Американська організація грейдингу (засн. 2009). Числова шкала 85–100. Домінує у сегменті Atari та ранніх PC.
Парсинг (Parsing)	Автоматизоване вилучення даних (базових цін) з вебсторінок PriceCharting.com за допомогою HTTP-запитів та аналізу HTML/JSON-відповіді.
Pricing Engine	Алгоритмічне ядро RareScore – модуль розрахунку вартості примірника на основі базової ціни та системи коефіцієнтів.
Micro-ORM (sqlite-net-pcl)	Легка бібліотека об'єктно-реляційного відображення, що маппить C#-класи на таблиці SQLite через атрибути [Table], [PrimaryKey], [AutoIncrement].
CreateTableAsync<T>()	Асинхронний метод Micro-ORM для створення таблиці БД на основі C#-класу. Замінює механізм міграцій EF Core.
MVVM	Архітектурний шаблон; розподіл між Model (бізнес-логіка), View (UI) та ViewModel (стан UI, INotifyPropertyChanged). У RareScore – тільки для аналітики.
Code-Behind (Event-Driven)	Підхід, за якого логіка UI обробляється безпосередньо в .xaml.cs-файлі сторінки через обробники подій. У RareScore – для форм CRUD та списків.
KPI (Key Performance Indicator)	Розрахункові аналітичні показники колекції: загальна вартість, середня вартість одиниці, розподіл за станом, динаміка ринкової вартості.

Продовження таблиці 2.4

Термін	Визначення
ZXing.Net.MAUI	Бібліотека розпізнавання штрих-кодів (EAN-13, UPC-A та ін.) з відеопотоку камери. Використовується для автозаповнення форми AddGamePage.
EAN/UPC	Стандарти штрихового кодування: EAN-13 (міжнародний, 13 цифр), UPC-A (американський, 12 цифр). Нанесені на коробки відеоігор.

2.2 Розробка математичної моделі процесу аналізу рідкісності та прогнозування ринкової вартості відеоігор

Pricing Engine є обчислювальним ядром комплексу RareScore та реалізує формалізовану математичну модель розрахунку оціночної вартості конкретного фізичного примірника колекційного видання відеогри. Модель базується на мультиплікативно-адитивній структурі: до базової ринкової ціни послідовно застосовуються коригувальні коефіцієнти та поправки.

2.2.1 Опис вхідних параметрів моделі

Вхідними параметрами Pricing Engine є атрибути конкретного запису GameItem. Визначимо позначення кожного компонента:

P_base – базова ринкова ціна стану Loose [USD], отримана парсингом або введена вручну.

K_cond – коефіцієнт стану збереження (Condition Multiplier). Безрозмірний мультиплікатор, що відображає ринкову різницю між категоріями стану.

K_ed – коефіцієнт типу видання (Edition Multiplier). Відображає додаткову ринкову цінність обмежених та колекторських видань відносно стандартного.

P_miss – сукупний штраф за некомплектність [частки від 1]. Застосовується виключно для стану CIB на основі булевих прапорців HasManual, HasArtbook, HasFigure, HasSteelbook, HasSoundtrack.

B_wata – грейдингова премія [частки від 1]. Застосовується виключно для стану Sealed за наявності верифікованого числового грейду (поле WataGrade ≠ NULL).

2.2.2 Значення коефіцієнтів моделі

Таблиця 2.5 – Значення коефіцієнта стану K_cond

Стан (Condition)	K_cond	Обґрунтування
Loose	1.00	Базова категорія; P_base визначена для стану Loose.
CIB (повна комплектація)	1.40	Медіана надбавки CIB-до-Loose за даними PriceCharting.
CIB (часткова комплектація)	1.20–1.35	Знижений мультиплікатор при відсутності компонентів; точне значення після P_miss.
Sealed (без грейду)	3.00	Мінімальне спостережуване значення мультиплікатора Sealed-до-Loose.
Sealed (з грейдом WATA/VGA)	3.00 + B_wata	Базовий Sealed-мультиплікатор із додаванням грейдингової премії.

Таблиця 2.6 – Значення коефіцієнта типу видання K_ed

Тип видання (Type)	K_ed	Обґрунтування
Standard	1.00	Базовий тип; без корекції.
Greatest Hits / Players Choice / Platinum	0.75	Перевидання зниженої ціни; ринковий дисконт 15–25%.
Limited Edition	1.30	Обмежений тираж з додатковим вмістом; надбавка 25–35%.
Collector's Edition	1.60	Повне колекторське видання; надбавка 50–70%.
Special / Anniversary Edition	1.20	Ювілейне видання з частковим додатковим вмістом.

Таблиця 2.7 – Значення штрафу за некомплектність P_miss (тільки для CIB)

Відсутній компонент (поле = false)	Дисконт, %	Примітки
Мануал (HasManual = false)	–10%	Базовий компонент CIB; відсутність суттєво знижує ліквідність.
Стілбук (HasSteelbook = false)	–15%	За умови, що стілбук є частиною оригінальної комплектації.
Артбук (HasArtbook = false)	–12%	Для видань, де артбук входить до комплектації Collector's Edition.
Фігурка / статуетка (HasFigure = false)	–20%	Найбільший штраф; фігурки є найціннішим елементом колекторських видань.
Саундтрек (HasSoundtrack = false)	–5%	Мінімальний штраф; вплив на ліквідність незначний.

Таблиця 2.8 – Значення грейдингової премії B_wata (тільки для Sealed)

Діапазон оцінки WATA / VGA	B_wata	Обґрунтування
Грейд відсутній (WataGrade = NULL)	0.00	Без сертифікації; базовий Sealed-мультиплікатор.
< 7.0 (Poor–Good)	0.05	Низький грейд; мінімальна цінність сертифікації.
7.0 – 8.5 (Very Good)	0.15	Задовільний стан; помірне преміювання.
8.5 – 9.2 (Near Mint)	0.30	Хороший стан; значна ліквідність на аукціонах.
9.2 – 9.6 (NM+)	0.50	Відмінний стан; входить до топ-сегменту ринку.
9.6 – 9.8 (Gem Mint)	0.75	Преміальний сегмент; суттєве зростання транзакційних цін.
9.8 – 10.0 (Perfect Mint)	1.00–1.50	Топ-сегмент; мультиплікатор визначається індивідуально для тайтлу.

2.2.3 Формалізована математична формула Pricing Engine

На основі визначених компонентів формулюється загальна формула розрахунку оціночної вартості EstimatedPrice.

Слід підкреслити, що коефіцієнти математичної моделі підбирались емпірично на основі аналізу транзакційних даних PriceCharting та Heritage Auctions. Модель слугує базовим орієнтиром для оцінки, а не претендує на абсолютну ринкову точність. Під час розробки виникла складність із верифікацією коефіцієнтів для рідкісних тайтлів: вибірка транзакцій для них є занадто малою для статистично значущих висновків, тому модель

може давати відхилення від реальних аукціонних цін – це відоме обмеження поточної версії [8].

$$P_{final} = P_{base} \times K_{cond} \times K_{ed} \times (1 - P_{miss}) \times (1 + B_{wata})$$

де:

P_{final} – фінальна оціночна вартість примірника [USD];

P_{base} – базова ринкова ціна стану Loose [USD];

$K_{cond} \in \{1.00; 1.20...1.40; 3.00\}$ – коефіцієнт стану;

$K_{ed} \in \{0.75; 1.00; 1.20; 1.30; 1.60\}$ – коефіцієнт типу видання;

$P_{miss} = \Sigma d_i$ – сукупний штраф за некомплектність;

$B_{wata} \in [0.00; 1.50]$ – грейдингова премія.

Умови застосування складових формули формалізуються логічними предикатами:

$P_{miss} = (Condition = «CIB») \; ? \; \Sigma d_i(HasX = false) : 0;$

$B_{wata} = (Condition = «Sealed» \wedge WataGrade \neq NULL) \; ? \; f(WataGrade) : 0,$

де $f(WataGrade)$ – кусково-лінійна функція відповідності числового значення грейду до премії B_{wata} відповідно до таблиці 2.8.

2.2.4 Приклад розрахунку

Розглянемо розрахунок для конкретного примірника: The Legend of Zelda: Breath of the Wild (Nintendo Switch), Collector's Edition, стан CIB,

відсутній артбук, саундтрек відсутній, решта компонентів присутні, грейд відсутній. Базова ціна Loose: P_base = 45.00 USD.

Вхідні параметри: K_cond = 1.40 (CIB); K_ed = 1.60 (Collector's Edition); P_miss = 0.12 (HasArtbook = false) + 0.05 (HasSoundtrack = false) = 0.17; B_wata = 0.

$$P_{final} = 45.00 \times 1.40 \times 1.60 \times (1 - 0.17) \times (1 + 0) \approx 83.79 \text{ USD}$$

Таким чином, оціночна вартість примірника Collector's Edition без артбуку та саундтреку становить приблизно 83.79 USD при базовій ціні Loose 45.00 USD. Мультиплікатор відносно базової ціни Loose становить 1.86x, що є коректним для CIB Collector's Edition із двома відсутніми компонентами.

2.3 Розробка логічної моделі програмного комплексу

Логічна модель системи описується двома типами поведінкових UML-діаграм: діаграмою прецедентів (Use Case Diagram), що визначає межі системи та взаємодію акторів з її функціями, та діаграмами діяльності (Activity Diagram), що деталізують алгоритми ключових процесів. Структурну основу логічної моделі складає інформаційна модель бази даних.

2.3.1 Інформаційна модель бази даних

Надійне та високопродуктивне збереження локальних даних у програмно-методичному комплексі «RareScore» реалізовано на базі реляційної системи керування базами даних SQLite, робота з якою здійснюється за допомогою асинхронного micro-ORM пакета SQLite-net-pcl. Інформаційна модель додатка спроєктована з урахуванням специфіки предметної області та містить дві основні взаємопов'язані сутності (таблиці): GameItem та PriceSample.

Сутність GameItem виступає центральним елементом моделі та інкапсулює повний набір атрибутів, необхідних для ідентифікації відеогри, опису її фізичного стану, поточного рівня комплектації за стандартом CIB, грейдингових міток WATA, а також результатів розрахунку вартості. Сутність PriceSample призначена для ведення логів та збереження історичних зрізів ринкової вартості кожного конкретного лоту, що дозволяє відстежувати динаміку цінових трендів у часі.

Між таблицями GameItem та PriceSample на рівні схеми даних встановлено відношення композиції (один до багатьох, 1:N). Зв'язок реалізовано за допомогою зовнішнього ключа ItemId у таблиці семплів, який посилається на унікальний ідентифікатор Id у таблиці ігор. Детальна специфікація полів, фізичних типів даних СКБД SQLite та їхнього маппінгу на класи мови C# наведена у таблицях 2.8 та 2.9.

Таблиця 2.9 – Специфікація таблиці GameItem

Поле (Column)	Тип SQLite	Атрибут C#	Обмеження	Призначення
Id	INTEGER	[PrimaryKey], [AutoIncrement]	PK, NOT NULL	Первинний ключ
Title	TEXT	[Column]	NOT NULL	Назва тайтлу; параметр пошуку при парсингу
Platform	TEXT	[Column]	NOT NULL	Ігрова платформа; ключ запиту до PriceCharting разом з Title
Type	TEXT	[Column]	NOT NULL	Тип видання; визначає K_ed у Pricing Engine
Condition	TEXT	[Column]	NOT NULL	Стан примірника; визначає K_cond та логіку штрафів
WataGrade	TEXT	[Column]	NULLABLE	Числова оцінка WATA або VGA. NULL = грейд відсутній
BasePrice	REAL	[Column]	DEFAULT 0	Базова ціна Loose з PriceCharting; вхідний параметр P_base

Продовження таблиці 2.9

Поле (Column)	Тип SQLite	Атрибут C#	Обмеження	Призначення
EstimatedPrice	REAL	[Column]	DEFAULT 0	Оціночна вартість, обчислена Pricing Engine
ImageUrl	TEXT	[Column]	NULLABLE	Шлях до зображення обкладинки. NULL = не прикріплено
PurchaseDate	TEXT	[Column]	NULLABLE	Дата придбання (ISO 8601). NULL = не вказано
Notes	TEXT	[Column]	NULLABLE	Довільні нотатки; не використовується у Pricing Engine
HasManual	INTEGER (bool)	[Column]	DEFAULT 1	Наявність мануалу (булевий тип); враховується тільки при CIB
HasArtbook	INTEGER (bool)	[Column]	DEFAULT 0	Наявність артбуку; релевантний для Collector/Limited
HasFigure	INTEGER (bool)	[Column]	DEFAULT 0	Наявність фігурки Collector's Edition
HasSteelbook	INTEGER (bool)	[Column]	DEFAULT 0	Наявність Steelbook-футляра
HasSoundtrack	INTEGER (bool)	[Column]	DEFAULT 0	Наявність диску або картки з саундтреком

Таблиця 2.10 – Специфікація полів сутності PriceSample

Поле (Column)	Тип SQLite	Атрибут C#	Обмеження	Призначення
Id	INTEGER	[PrimaryKey], [AutoIncrement]	PK, NOT NULL	Первинний ключ
ItemId	INTEGER	[Column]	NOT NULL, FK	Зовнішній ключ для зв'язку з GameItem.Id
Date	TEXT (ISO 8601)	[Column]	NOT NULL	Дата фіксації цінового зрізу; TEXT для лексикографічного сортування
Price	REAL	[Column]	NOT NULL	Базова ринкова ціна Loose [USD] на момент Date
Source	TEXT	[Column]	NOT NULL	Джерело: «PriceCharting» або «Manual»; для фільтрації при побудові графіків

Між таблицями GameItem та PriceSample встановлено відношення «один до багатьох» (1:N). Індексування таблиці PriceSample за полем ItemId забезпечує продуктивність запитів при N > 10 000 записів:

CREATE INDEX IF NOT EXISTS idx_pricesample_itemid ON PriceSample (ItemId);

Даний індекс скорочує час виконання запиту вибірки цінової історії з O(N) лінійного сканування до O(log N) через B-tree структуру індексу SQLite.

2.3.2 Діаграма прецедентів (Use Case Diagram)

Система RareScore має єдиного актора – Користувач (колекціонер або ретейлер). Функціональну структуру та межі взаємодії актора з компонентами комплексу декомпоновано на сім базових прецедентів (діаграма наведена у Додатку Б, рис. Б.1):

– UC-01 – Додавання лоту вручну: користувач відкриває форму AddGamePage, заповнює реквізити та прапорці комплектації, після чого система фіксує запис у базі;

– UC-02 – Додавання лоту за штрихкодом: модуль ZXing.Net.MAUI розпізнає код із камери та автоматично заповнює поля форми метаданими;

– UC-03 – Автоматизований парсинг маркерів: HttpClient ініціює GET-запит до PriceCharting.com для отримання базової ціни Loose;

– UC-04 – Розрахунок вартості (Pricing Engine): ізольоване обчислювальне ядро застосовує систему зважених коефіцієнтів для визначення EstimatedPrice;

– UC-05 – Управління ціновими семплами: фіксація історичних зрізів вартості (сутності PriceSample) для відстеження трендів;

– UC-06 – Генерація PDF-звіту: конвеєр верстки звітного документа засобами QuestPDF із вибором шляху через FileSavePicker;

– UC-07 – Перегляд аналітики та графіків: AnalyticsViewModel (шар MVVM) завантажує ряди даних для LiveChartsCore та прораховує KPI колекції через механізм Data Binding.

2.3.3 Діаграма діяльності AD-01: Генерація PDF-звіту

Логіка формування вихідних звітних документів у форматі кросплатформового представлення реалізована безпосередньо в обробнику подій інтерфейсного шару за допомогою декларативного Fluent API бібліотеки QuestPDF. Процес генерації PDF-документа (прецедент UC-06) відображено на відповідній діаграмі діяльності (рис. Б.2).

Алгоритм розгалужується на такі послідовні етапи: запит користувача на експорт звіту через графічний інтерфейс; асинхронне неблокуюче читання даних з локальної бази даних SQLite за допомогою методу LocalDatabase.GetItemsAsync(); логічна валідація колекції на наявність записів; лінійний прорахунок аналітичних KPI метрик у разі успішного отримання даних; декларативна верстка компонентів сторінки (Header, динамічна таблиця лотів, Footer) засобами бібліотеки QuestPDF та безпосередній бінарний запис PDF-файлу на диск пристрою за допомогою системного діалогу вибору шляху збереження.

2.3.4 Діаграма діяльності AD-02: Парсинг → Pricing Engine

Інтеграція підсистеми збору зовнішніх ринкових маркерів та обчислювального ядра ціноутворення формалізована на відповідній діаграмі діяльності (рис. Б.3). Даний алгоритм забезпечує автоматичний розрахунок вартості при додаванні або зміні стану комплектації примірника відеогри.

Процес запускається під час створення нового запису або коригування прапорців комплектації існуючої відеогри в інтерфейсі додатка. Програма виконує перевірку підключення до мережі Інтернет. У

разі його наявності ініціюється асинхронний HTTP GET запит через вбудований клас HttpClient до PriceCharting.com, звідки вилучається актуальна базова ціна, що автоматично фіксується в таблиці історії цін PriceSample та оновлює поле BasePrice сутності GameItem. За відсутності мережі додаток автоматично переходить на роботу з локально кешованими даними BasePrice з об'єкта. На наступному етапі параметри комплектації передаються до статичного методу PricingEngine.CalculateValue(), де через умовні оператори визначаються коефіцієнти стану K_cond та типу видання K_{ed}, вираховуються штрафи за відсутність мануалів або інших компонентів (P_{miss}), а також нараховується премія за наявності сертифікованого грейду WATA (B_{wata}). Фінальна оціночна вартість (P_{final}) присвоюється властивості EstimatedPrice та фіксується в SQLite за допомогою асинхронного виклику LocalDatabase.SaveItemAsync(item), після чого інтерфейс користувача миттєво оновлюється через механізм прив'язки даних.

2.4 Розробка технічного завдання на створення програмного засобу

Технічне завдання на розробку програмно-методичного комплексу RareScore формалізує вимоги до системи, визначені в ході аналізу предметної галузі та проєктування. Вимоги поділяються на функціональні та нефункціональні.

2.4.1 Функціональні вимоги

ФВ-01. Управління каталогом колекції: система повинна забезпечувати додавання нового лоту вручну через форму AddGamePage з полями: назва, платформа, тип видання, стан збереження, прапорці комплектації (HasManual, HasArtbook, HasFigure, HasSteelbook, HasSoundtrack), оцінка WATA; підтримувати додавання лоту шляхом сканування EAN/UPC-штрихкоду через ZXing.Net.MAUI з автоматичним пошуком метаданих; забезпечувати редагування та видалення існуючих записів; підтримувати прикріплення зображення обкладинки.

ФВ-02. Автоматизоване визначення базової ринкової ціни: система повинна здійснювати HTTP-запит до PriceCharting.com для отримання базової ринкової ціни Loose; результат зберігати у таблиці PriceSample з фіксацією дати та джерела; у разі відсутності мережевого з'єднання використовувати останнє відоме значення.

ФВ-03. Розрахунок оціночної вартості (Pricing Engine): система повинна автоматично розраховувати EstimatedPrice за математичною моделлю після кожного оновлення базової ціни або зміни атрибутів лоту з урахуванням P_base, K_cond, K_ed, P_miss (тільки CIB), B_wata (тільки Sealed).

ФВ-04. Аналітична підсистема (MainPage, MVVM): система повинна відображати динаміку базової ринкової ціни у вигляді лінійного графіка (LineSeries, LiveChartsCore) та розраховувати KPI колекції: загальна оціночна вартість, середня вартість одиниці, розподіл за станом збереження.

ФВ-05. Генерація аналітичних звітів: система повинна формувати структурований PDF-звіт із зведеною таблицею лотів, підсумковими KPI та датою генерації (QuestPDF); підтримувати експорт каталогу у форматі CSV.

2.4.2 Нефункціональні вимоги

Таблиця 2.11 – Нефункціональні вимоги до комплексу RareScore

ID	Категорія	Вимога	Метрика / Обмеження
НФВ-01	Платформа	Цільовою ОС є Windows 10 (версія 1903+) та Windows 11.	Windows 10 build 18362+; Windows 11
НФВ-02	Автономність	Всі функції перегляду, редагування та розрахунку доступні без підключення до мережі.	Повна офлайн-функціональність, крім парсингу
НФВ-03	Швидкодія	Час відображення каталогу та розрахунок Pricing Engine.	$t_{load} \leq 2$ с; $t_{calc} \leq 100$ мс
НФВ-04	Швидкодія	Час ініціалізації схеми БД. Холодний старт застосунку.	$t_{init} \leq 200$ мс; $t_{startup} \leq 3$ с
НФВ-05	Швидкодія	Генерація PDF-звіту для колекції до 500 лотів.	$t_{pdf} \leq 10$ с для $N \leq 500$
НФВ-06	Збереження даних	Дані зберігаються локально у єдиному файлі SQLite.	Файл: rarescore.db у AppData\Local\RareScore

Продовження таблиці 2.11

ID	Категорія	Вимога	Метрика / Обмеження
НФВ-07	Масштабованість	Коректна робота при обсязі до 10 000 лотів та 100 000 записів PriceSample.	≤ 10 000 GameItem; ≤ 100 000 PriceSample
НФВ-08	Надійність	Збій при парсингу не повинен призводити до втрати або пошкодження даних.	MTBF (витік даних при мережевій помилці) = 0
НФВ-09	Ресурси	Базове споживання оперативної пам'яті у стані спокою.	RAM idle ≤ 150 МБ
НФВ-10	Сканування	Час розпізнавання штрих-коду після активації сканера.	t_scan ≤ 3 с при достатньому освітленні

3 ПРОЄКТУВАННЯ ТА РЕАЛІЗАЦІЯ ПРОГРАМНО-МЕТОДИЧНОГО КОМПЛЕКСУ АНАЛІЗУ РІДКІСНОСТІ ТА ВАРТОСТІ ВІДЕОІГОР

Третій розділ присвячено безпосередньому проєктуванню та реалізації програмно-методичного комплексу RareScore. Описано архітектурну організацію проєкту у середовищі Visual Studio, наведено покроковий алгоритм роботи обчислювального ядра PricingEngine, розглянуто реалізацію інтерфейсу користувача на основі XAML-розмітки платформи .NET MAUI, описано конвеєр генерації PDF-звітів засобами QuestPDF та підсистему сканування штрих-кодів ZXing.Net.MAUI. Розділ завершується аналізом результатів тестування комплексу.

3.1 Проєктування структури програмного комплексу

Проєкт RareScore організовано у середовищі Microsoft Visual Studio 2022 як єдиний рішення (.sln), що містить один основний проєкт типу .NET MAUI Application. Гібридна архітектура, обрана у розділі 2, безпосередньо визначила поділ кодової бази на чітко розмежовані логічні шари, реалізовані через каталоги проєкту. Така організація забезпечує дотримання принципу розподілення відповідальностей (Separation of Concerns) та спрощує супровід коду.

Структура каталогів проєкту RareScore у Solution Explorer наведена на рисунку 3.1 та деталізована у таблиці 3.1.

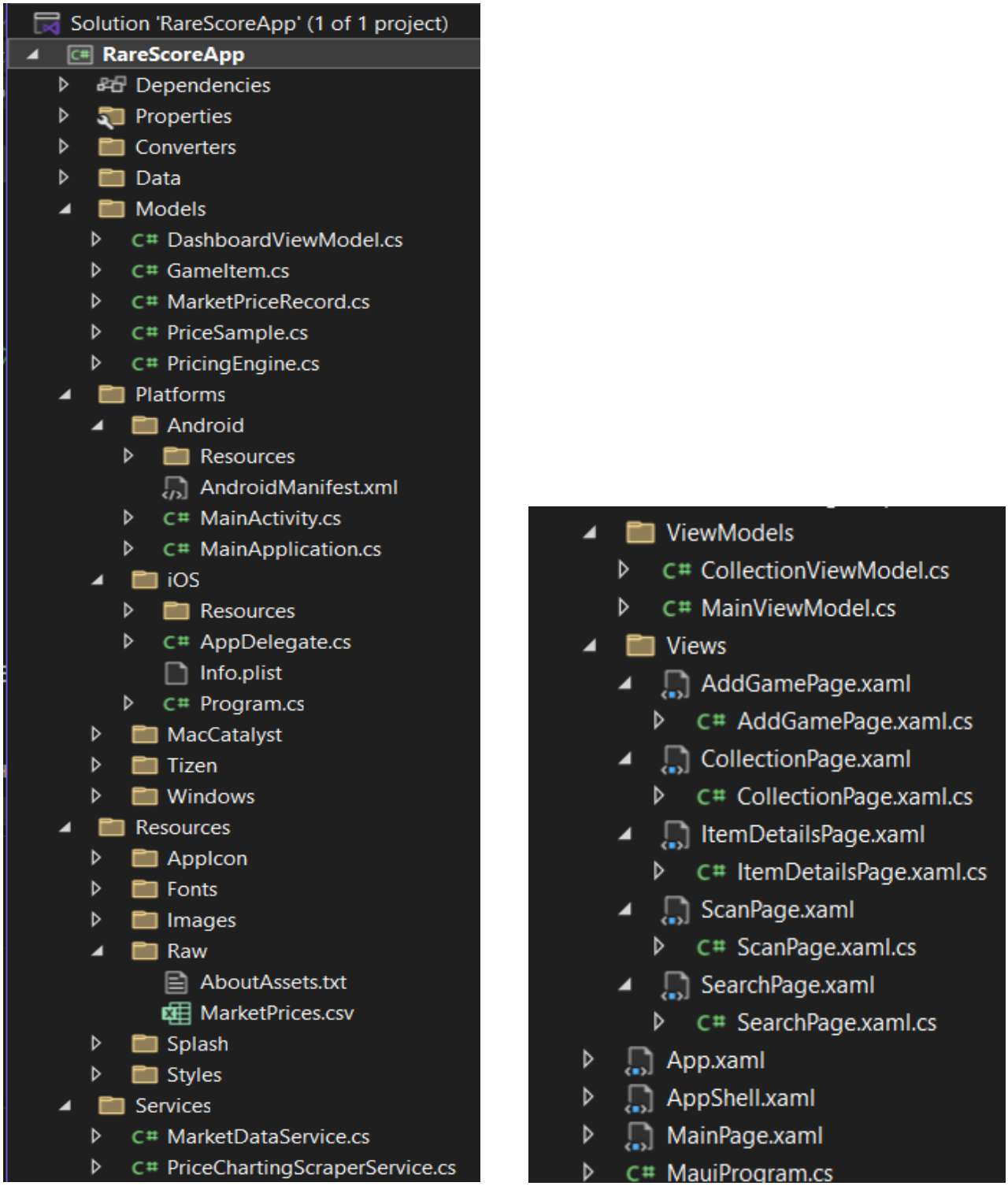

Рисунок 3.1 – Структура проєкту RareScore у вікні Solution Explorer Visual Studio 2022

Таблиця 3.1 – Структура каталогів проєкту RareScore

Каталог / Файл	Призначення	Ключові класи / файли
Models/	Класи предметної моделі – сутності БД та атрибути колекції	GameItem.cs (16 атрибутів), PriceSample.cs
Data/	Репозиторій доступу до даних SQLite через Micro-ORM	LocalDatabase.cs
ViewModels/	ViewModel-шар MVVM для аналітичної підсистеми	AnalyticsViewModel.cs
Views/Pages/	XAML-сторінки інтерфейсу користувача	MainPage.xaml, AddGamePage.xaml, CollectionPage.xaml, SearchPage.xaml
Core/	Ізольована бізнес-логіка: обчислювальне ядро та парсер	PricingEngine.cs (static), PriceChartingParser.cs (async)
Resources/Styles/	Глобальні XAML-стилі, палітра кольорів, шрифти	Styles.xaml, Colors.xaml
Platforms/Windows/	Специфічна конфігурація WinUI 3 (Package.appxmanifest)	app.manifest, Package.appxmanifest
MauiProgram.cs	Точка входу: реєстрація сервісів, ініціалізація ZXing	MauiProgram.cs
AppShell.xaml	Навігаційна оболонка Shell з маршрутами між сторінками	AppShell.xaml

Ключовими сутностями інформаційної моделі є класи GameItem та PriceSample. Клас GameItem є центральним елементом: він інкапсулює 16 атрибутів, що повністю описують фізичний примірник колекційного видання відеогри – від ідентифікаційних даних (Id, Title, Platform) до

характеристик стану (Condition, WataGrade), комплектності (HasManual, HasArtbook, HasFigure, HasSteelbook, HasSoundtrack), фінансових параметрів (BasePrice, EstimatedPrice) та мета-даних (PurchaseDate, Notes, ImageUrl). Тип поля WataGrade визначено як TEXT (NULLABLE), що забезпечує зберігання числового значення грейду у текстовому форматі та можливість перетворення у decimal під час розрахунку в PricingEngine.

Між таблицями GameItem та PriceSample встановлено відношення «один до багатьох» (1:N). Кожному запису GameItem може відповідати необмежена кількість записів PriceSample – цінових зрізів, що фіксуються щоразу під час парсингу або ручного введення. Зв'язок реалізовано через зовнішній ключ ItemId у таблиці PriceSample, що посилається на первинний ключ Id таблиці GameItem (рис. 3.2). Для забезпечення продуктивності запитів вибірки при N > 10 000 записів на таблицю PriceSample накладено B-tree індекс за полем ItemId:

```
CREATE INDEX IF NOT EXISTS idx_pricesample_itemid
    ON PriceSample (ItemId);
```

Наявність цього індексу скорочує часову складність запиту вибірки цінової історії конкретного лоту з O(N) лінійного сканування до O(log N) завдяки B-tree структурі індексу SQLite, що є критичним при масштабуванні бази до 100 000 записів PriceSample (нефункціональна вимога НФВ-07).

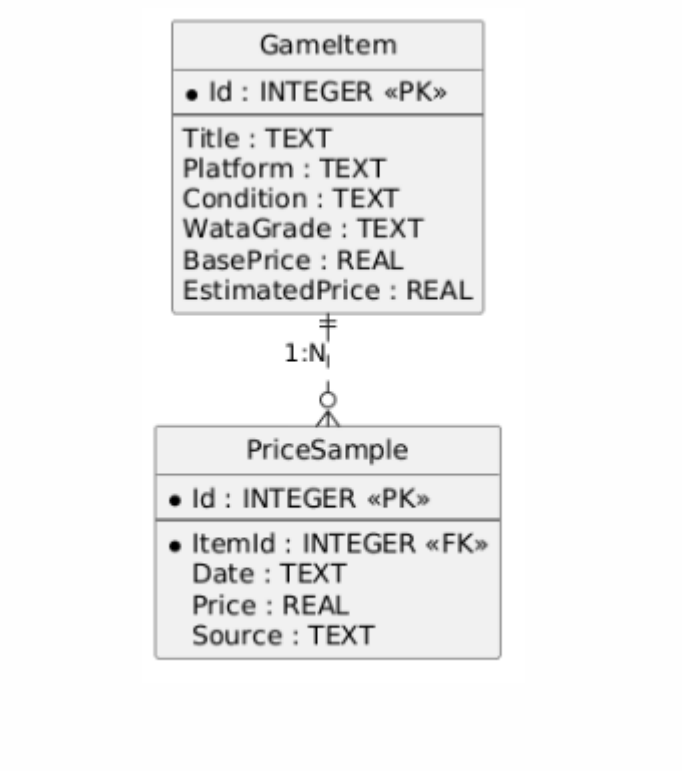

Рисунок 3.2 – ER-схема зв'язку 1:N між таблицями GameItem та PriceSample у базі даних SQLite

Репозиторій LocalDatabase є єдиним класом, що взаємодіє з рушієм SQLite через об'єкт SQLiteAsyncConnection. Він реалізує повний набір асинхронних CRUD-методів: GetItemsAsync() для завантаження каталогу, GetItemByIdAsync(int id), SaveItemAsync(GameItem item) для збереження або оновлення запису, DeleteItemAsync(GameItem item), GetPriceSamplesAsync(int itemId) для отримання цінової історії конкретного лоту, а також SavePriceSampleAsync(PriceSample sample). Всі методи позначені ключовим словом async та повертають об'єкти Task<T>, що забезпечує неблокуюче виконання в UI-потоці платформи .NET MAUI. Ініціалізація схеми бази даних викликається одноразово в MauiProgram.cs під час старту застосунку через виклик CreateTableAsync<T>() для кожного з двох класів-сутностей.

3.2 Розробка алгоритму роботи програмного комплексу

Алгоритмічну складову комплексу RareScore формують два незалежних модулі: статичний клас PricingEngine, що реалізує детерміновану математичну модель розрахунку вартості, та клас PriceChartingParser, що забезпечує асинхронний збір ринкових даних. Обидва модулі розташовані у каталозі Core/ та повністю відокремлені від UI-шару, що забезпечує їх ізольоване модульне тестування.

3.2.1 Алгоритм методу PricingEngine.CalculateValue()

Статичний метод PricingEngine.CalculateValue(GameItem item) є ядром системи оцінювання. Він приймає повністю заповнений об'єкт GameItem та повертає розраховане значення EstimatedPrice типу decimal. Метод є синхронним та детерміністичним: при однакових вхідних параметрах він завжди повертає ідентичний результат. Блок-схема алгоритму наведена на рисунку 3.3.

Алгоритм виконується у шість послідовних кроків.

Крок 1 – Валідація вхідних даних. На початку виконується перевірка: якщо значення BasePrice ≤ 0 або дорівнює null, метод негайно повертає 0, запобігаючи передачі некоректних вхідних даних до математичної формули. Ця перевірка є критичною у випадку, коли парсинг PriceCharting не дав результату та в об'єкті зберігається значення за замовчуванням.

Крок 2 – Визначення коефіцієнта стану K_cond. За допомогою конструкції switch на рядковому полі item.Condition визначається мультиплікатор стану. Значення «Loose» відповідає K_cond = 1.00 (базова

категорія); «CIB» – K_cond = 1.40; «Sealed» – K_cond = 3.00. Будь-яке нерозпізнане значення фолбечиться до 1.00 через гілку default.

Крок 3 – Визначення коефіцієнта типу видання K_ed. Аналогічна switch-конструкція на полі item.Type реалізує маппінг типів видань на мультиплікатори. «Greatest Hits» / «Players Choice» / «Platinum» → 0.75; «Standard» → 1.00; «Special Edition» → 1.20; «Limited Edition» → 1.30; «Collector's Edition» → 1.60.

Крок 4 – Розрахунок штрафу за некомплектність P_miss. Логічний предикат перевіряє умову item.Condition == "CIB". Якщо умова істинна, виконується лінійне підсумовування часткових штрафів: HasManual == false → +0.10; HasSteelbook == false → +0.15; HasArtbook == false → +0.12; HasFigure == false → +0.20; HasSoundtrack == false → +0.05. Якщо стан не є CIB, P_miss примусово встановлюється у 0. Таким чином, максимально можливий штраф за відсутність усіх п'яти компонентів становить P_miss = 0.62, тобто 62% від базового CIB-мультиплікатора.

Крок 5 – Розрахунок грейдингової премії B_wata. Застосовується подвійна умова: item.Condition == "Sealed" та !string.IsNullOrEmpty(item.WataGrade). Якщо обидві умови виконано, числове значення грейду парситься з текстового поля через decimal.TryParse(). Залежно від отриманого числа застосовується кусково-лінійна функція відповідно до таблиці 2.8. Якщо парсинг не вдається або стан не є Sealed, B_wata = 0.

Крок 6 – Застосування формули та округлення. Фінальне значення розраховується за формулою:

```
decimal p_final = p_base * k_cond * k_ed * (1 - p_miss) * (1 + b_wata);
return Math.Round(p_final, 2);
```

Результат округлюється до двох знаків після коми (точність центів USD) та присвоюється полю item.EstimatedPrice перед збереженням у

базу даних. Метод не містить жодних залежностей від UI-шару, що робить його повністю придатним для ізольованого модульного тестування.

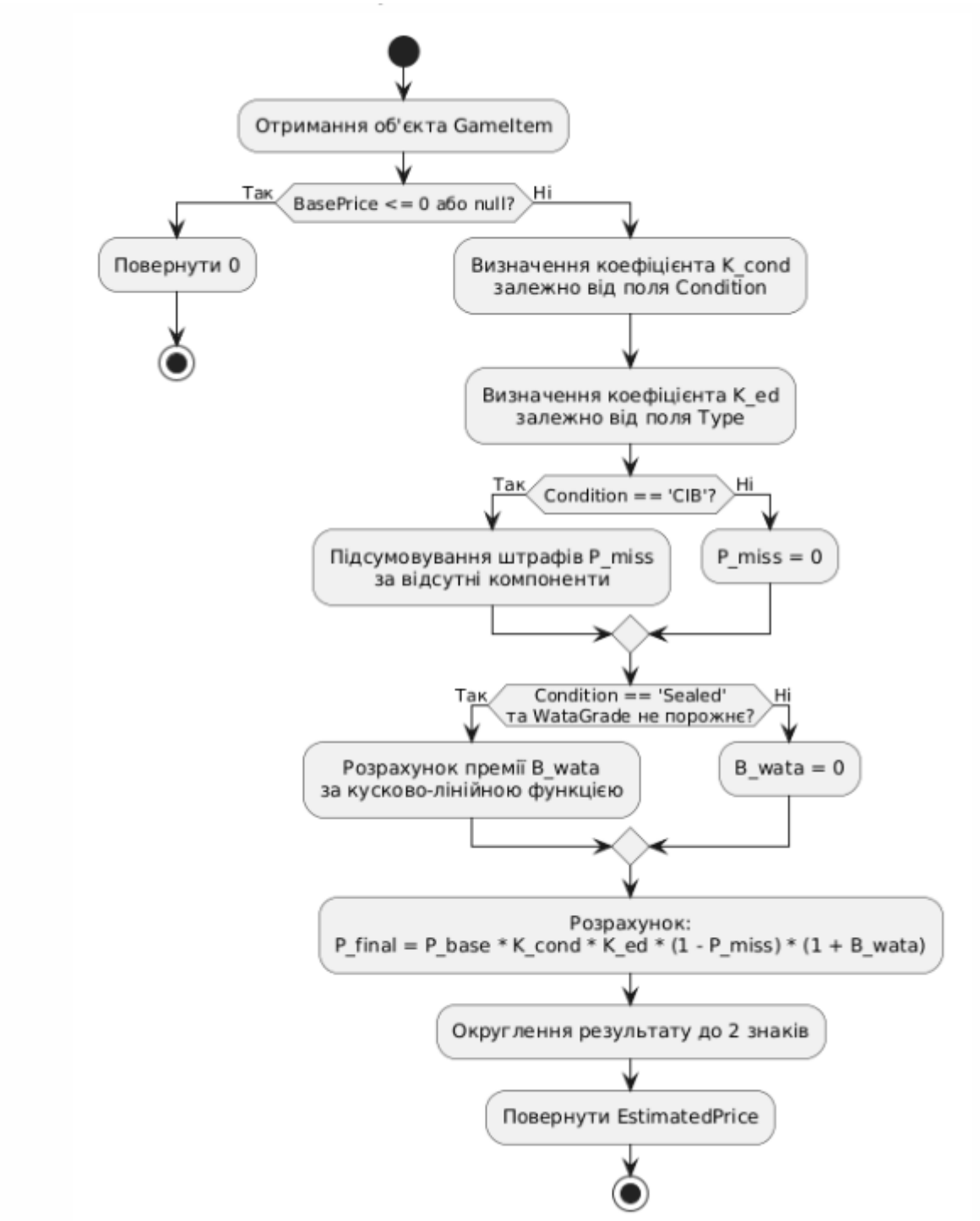

Рисунок 3.3 – Блок-схема алгоритму методу PricingEngine.CalculateValue()

3.2.2 Алгоритм асинхронного парсингу через HttpClient

Клас PriceChartingParser реалізує асинхронний метод FetchBasePriceAsync(string title, string platform), що повертає Task<decimal?>. Метод використовує єдиний статичний екземпляр HttpClient, оголошений на рівні класу – це усуває типову помилку «socket exhaustion» при багаторазовому створенні HttpClient. Алгоритм парсингу ілюструється рисунком 3.4.

Алгоритм складається з таких послідовних етапів.

Етап 1 – Формування запиту. Рядок запиту складається з назви тайтлу та платформи (title + " " + platform), URI-кодується через Uri.EscapeDataString() та підставляється у шаблон URL сервісу PriceCharting. До заголовків запиту додається User-Agent браузера для запобігання блокуванню з боку серверу.

Етап 2 – Виконання HTTP GET-запиту. Виклик await _client.GetStringAsync(url) виконується асинхронно без блокування UI-потоку. Весь блок обгорнутий у try-catch(HttpRequestException): у разі мережевої недоступності метод повертає null – це є сигналом для сторінки SearchPage використати кешоване значення BasePrice з об'єкта GameItem.

Етап 3 – Парсинг HTML-відповіді. З отриманого рядка HTML за допомогою регулярного виразу витягується числове значення ціни Loose. Обраний підхід на основі Regex є більш легким порівняно з повноцінним HTML-парсером (HtmlAgilityPack), достатнім для цільової структури сторінок PriceCharting.

Етап 4 – Збереження результату. У разі успішного парсингу повернене значення ціни зберігається у двох місцях: (а) у таблиці PriceSample через SavePriceSampleAsync() з фіксацією поточної дати та джерела «PriceCharting»; (б) у полі BasePrice об'єкта GameItem. Після

цього негайно викликається PricingEngine.CalculateValue() та оновлений GameItem зберігається у базі через SaveItemAsync(). Весь ланцюжок виконується асинхронно та не блокує UI-потік.

Рисунок 3.4 – Блок-схема алгоритму асинхронного парсингу PriceCharting.com

3.3 Реалізація програмного комплексу та інтерфейсу користувача

Інтерфейс користувача комплексу RareScore повністю реалізований засобами XAML-розмітки платформи .NET MAUI. Навігація між сторінками організована через Shell-навігацію AppShell.xaml, що дозволяє декларативно визначати маршрути та параметри переходу між сторінками. Застосунок містить чотири основних сторінки: MainPage (аналітика), CollectionPage (каталог колекції), AddGamePage (форма додавання лоту) та SearchPage (пошук та деталі лоту).

3.3.1 Головне вікно MainPage та прив'язка даних LiveChartsCore (MVVM)

Сторінка MainPage є аналітичним центром застосунку та єдиним компонентом, де застосовується архітектурний патерн MVVM. Прив'язка даних реалізується через властивість BindingContext сторінки, що встановлюється на екземпляр AnalyticsViewModel у Code-Behind: this.BindingContext = new AnalyticsViewModel(_database).

Клас AnalyticsViewModel реалізує інтерфейс INotifyPropertyChanged та надає UI такі прив'язувані властивості: ISeries[] PriceSeries – масив серій для лінійного графіка динаміки ціни тайтлу; ISeries[] CollectionSeries – масив серій для кругової діаграми розподілу за станом; string TotalValue – загальна оціночна вартість колекції; string AvgValue – середня вартість одиниці; int TotalCount – кількість лотів.

У XAML-розмітці MainPage.xaml графіки LiveChartsCore підключаються як стандартні XAML-елементи та прив'язуються до властивостей ViewModel через механізм Data Binding (рис. 3.5).

```
<lvc:CartesianChart
    Series="{Binding PriceSeries}"
    XAxes="{Binding XAxes}"
    HeightRequest="280" />
<lvc:PieChart
    Series="{Binding CollectionSeries}"
    HeightRequest="200" />
```

Рисунок 3.5 – Фрагмент XAML-розмітки підключення графіків LiveChartsCore

При кожному виклику методу LoadDataAsync() у ViewModel відбувається завантаження записів PriceSample із БД та побудова нових об'єктів LineSeries<DateTimePoint> для кожного тайтлу. Присвоєння значення властивості PriceSeries через сеттер автоматично викликає OnPropertyChanged("PriceSeries"), що змушує LiveChartsCore перерендерити графік без будь-якого явного втручання з боку View.

Загальний вигляд головного вікна MainPage з аналітичними графіками та панеллю KPI наведено на рисунку 3.6.

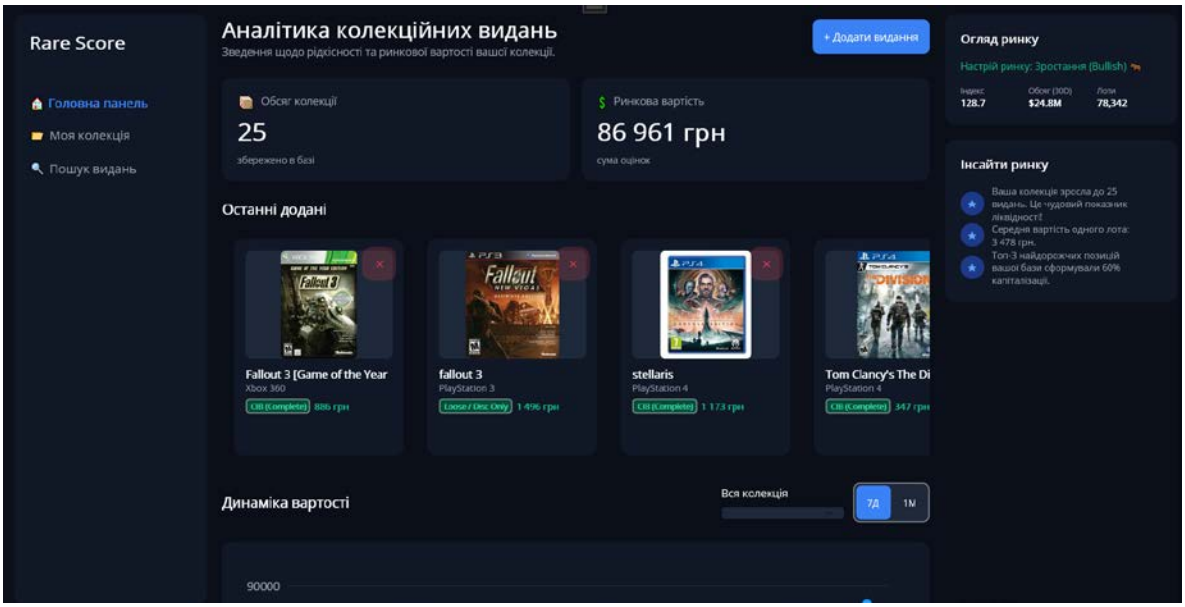

Рисунок 3.6 – Головне вікно MainPage з аналітичними графіками LiveChartsCore та панеллю KPI

3.3.2 Форма AddGamePage та обробка подій (Code-Behind)

Сторінка AddGamePage є формою введення даних про новий лот колекції та реалізована за паттерном Code-Behind (Event-Driven) без ViewModel. Вибір обумовлений наявністю десяти незалежних булевих чекбоксів комплектації (HasManual, HasArtbook, HasFigure, HasSteelbook, HasSoundtrack та п'яти відповідних допоміжних Label-елементів), стан видимості яких потребує миттєвого оновлення залежно від обраного стану Condition.

Зміна стану у Picker-елементі Condition викликає обробник Condition_SelectedIndexChanged(), що програмно перемикає видимість панелі комплектації (IsVisible = selectedCondition == "CIB") та панелі грейдингу (IsVisible = selectedCondition == "Sealed"). Такий підхід є синхронним і виконується миттєво у UI-потоці без накладних витрат механізму ICommand.

Обробник кнопки «Зберегти» SaveButton_Clicked() виконує такі операції: (1) формує об'єкт GameItem із поточного стану форми; (2) асинхронно викликає PriceChartingParser.FetchBasePriceAsync() (описаний у п. 3.2.2); (3) синхронно викликає PricingEngine.CalculateValue() та записує результат у item.EstimatedPrice; (4) зберігає об'єкт через _database.SaveItemAsync(item); (5) навігує назад на CollectionPage. Вигляд сторінки AddGamePage з формою додавання наведено на рисунку 3.7.

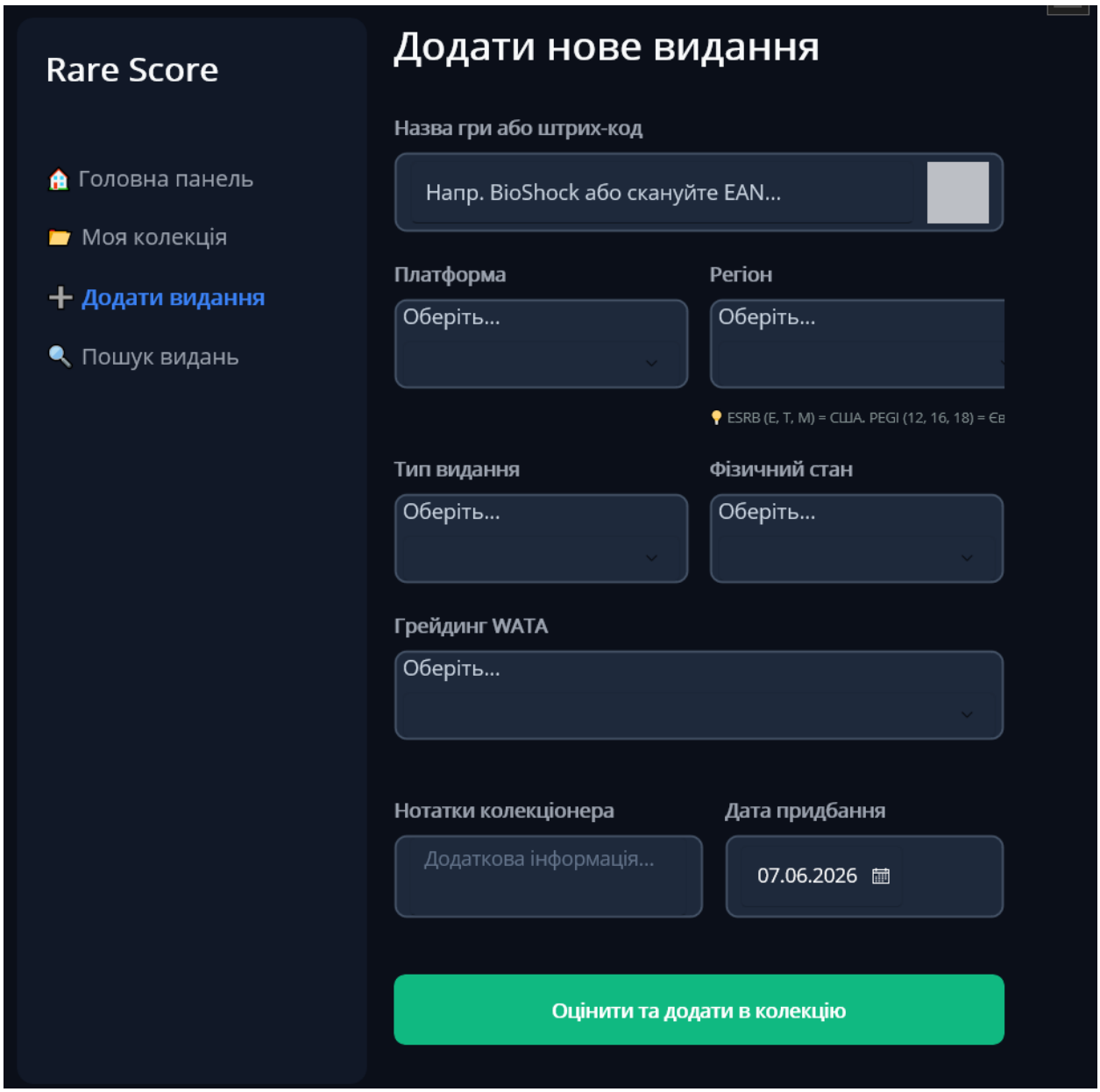

Рисунок 3.7 – Сторінка AddGamePage: форма додавання лоту з чекбоксами комплектації

3.3.3 Підсистема сканування штрих-кодів (ZXing.Net.MAUI)

Функція сканування штрих-коду ініціюється з AddGamePage натисканням кнопки «Сканувати». Виконується навігація на окрему

сторінку ScanPage, що містить єдиний XAML-елемент CameraBarcodeReaderView бібліотеки ZXing.Net.MAUI. Реєстрація бібліотеки виконується при ініціалізації застосунку в MauiProgram.cs через виклик builder.UseMauiApp<App>().UseZXingBarcodeReader().

Розпізнавання штрих-коду відбувається у реальному часі з відеопотоку камери. Подія BarcodeDetected спрацьовує у момент першого успішного розпізнавання та передає рядкове значення штрих-коду типів EAN-13 або UPC-A. Обробник події BarcodeDetected є асинхронним та виконується у UI-потоці через MainThread.BeginInvokeOnMainThread(), що є обов'язковим для модифікації UI-елементів з фонового потоку розпізнавання (рис. 3.8)

```
barcodeReader.BarcodeDetected += async (s, e) => {
    MainThread.BeginInvokeOnMainThread(async () => {
        barcodeReader.IsDetecting = false; // зупинити сканування
        string barcode = e.Results[0].Value;
        await Shell.Current.GoToAsync(",",
            new Dictionary<string, object> {{ "Barcode", barcode }});
    });
};
```

Рисунок 3.8 – Лістинг обробника події розпізнавання штрих-коду підсистеми ZXing.Net.MAUI

Після повернення на AddGamePage значення штрих-коду передається як QueryProperty та використовується для HTTP-запиту до зовнішнього API пошуку метаданих (назва, платформа), що автоматично заповнює відповідні поля форми. Режим сканування штрих-коду.

3.3.4 Конвеєр генерації PDF-звіту (QuestPDF)

Генерація PDF-звіту реалізована у обробнику події кнопки «Експорт PDF» на сторінці CollectionPage (Code-Behind підхід). QuestPDF використовує декларативний Fluent API, що концептуально відповідає CSS Flexbox: документ описується як ланцюжок методів-дескрипторів, що визначають контейнери, стовпці, рядки та вміст сторінки.

Конвеєр генерації складається з п'яти логічних етапів.

Етап 1 – Збір даних. Асинхронний виклик await _database.GetItemsAsync() завантажує повний каталог. Якщо список порожній, відображається модальне сповіщення та метод завершується достроково.

Етап 2 – Розрахунок KPI. Лінійним перебором списку розраховуються: сума EstimatedPrice (TotalValue), середнє значення (AvgValue), кількість записів кожного стану (looseCount, cibCount, sealedCount).

Етап 3 – Декларативна верстка документа. Структура документа описується через Document.Create() з вкладеними методами (рис. 3.9).

```
Document.Create(container => {
  container.Page(page => {
    page.Header().Element(ComposeHeader);
    page.Content().Element(ctx => ComposeContent(ctx, items));
    page.Footer().AlignCenter().Text(x => {
        x.CurrentPageNumber(); x.Span(" / "); x.TotalPages();
    });
  });
}).GeneratePdf(filePath);
```

Рисунок 3.9 – Фрагмент коду декларативної верстки PDF-звіту засобами QuestPDF

Метод ComposeContent() будує динамічну таблицю лотів через Table().ColumnsDefinition() та Header/Cell блоки у циклі foreach по списку GameItem. Для кожного запису відображаються: назва, платформа, тип, стан, базова ціна, оціночна вартість. Підсумкові KPI-блоки розміщуються після таблиці через Row().RelativeItem().

Етап 4 – Вибір шляху збереження. Системний діалог FileSavePicker дозволяє користувачеві обрати ім'я та каталог для збереження файлу. На Windows цей виклик вимагає передачі дескриптора батьківського вікна через WindowHandle, що отримується через Platform.CurrentActivity.

Етап 5 – Запис файлу. Виклик GeneratePdf(filePath) виконує рендеринг документа та записує бінарний PDF безпосередньо у вказаний шлях. Приклад згенерованого звіту наведено на рисунках 3.10-3.11.

	A	B	C	D	E	F
1	Назва	Тип видання	Платформа	Стан	Оціночна вартість (грн)	
2	Fallout 3 [Game of the Year Platinum Hits]	Standard Edition	Xbox 360	CIB (Complete)	886,49	
3	fallout 3	Standard Edition	PlayStation 3	Loose / Disc Only	1495,56	
4	stellaris	Standard Edition	PlayStation 4	CIB (Complete)	1172,56	
5	Tom Clancy's The Division	Standard Edition	PlayStation 4	CIB (Complete)	346,5	
6	far cry 3	Standard Edition	PlayStation 3	CIB (Complete)	386,42	
7	The Last of Us	Standard Edition	PlayStation 3	CIB (Complete)	539,99	
8	Bloodborne	Standard Edition	PlayStation 4	CIB (Complete)	928,31	
9	Fallout 4	Standard Edition	PlayStation 4	CIB (Complete)	474,57	
10	Need for Speed: Underground 2	Standard Edition	PlayStation 2	Loose / Disc Only	1099,56	
11	Cyberpunk 2077	Standard Edition	PlayStation 4	CIB (Complete)	918,64	
12	Resident Evil 4	Standard Edition	PlayStation 2	Loose / Disc Only	247,3	
13	stalker 2	Standard Edition	PlayStation 5	CIB (Complete)	1757,8	
14	cyberpunk 2077	Collector's Edition	PlayStation 4	CIB (Complete)	4800	
15	Cyberpunk 2077	Collector's Edition	PlayStation 4	Sealed (New)	11200	
16	Bloodborne	Special Edition	PlayStation 4	CIB (Complete)	990	
17	fallout 4	Collector's Edition	PlayStation 4	CIB (Complete)	6500	
18	Dishonored 2 Collector's Edition	Collector's Edition	PlayStation 4	CIB (Complete)	9963,93	
19	Cyberpunk 2077	Standard Edition	PlayStation 5	CIB (Complete)	1624,48	
20	Battlefield 3	Standard Edition	PlayStation 3	CIB (Complete)	246	
21	Fallout: New Vegas	Collector's Edition	PlayStation 3	CIB (Complete)	9985	
22	Cyberpunk 2077	Standard Edition	PlayStation 5	CIB (Complete)	1381	
23	Fallout 4	Standard Edition	PlayStation 4	CIB (Complete)	486	
24	Red Dead Redemption 2	Special Edition	PlayStation 4	CIB (Complete)	1320	
25	The Legend of Zelda: Tears of the Kingdom	Collector's Edition	Nintendo Switch	CIB (Complete)	4037	
26	Resident Evil 4	Collector's Edition	PlayStation 5	CIB (Complete)	24174	

Рисунок 3.10 – Приклад експортованого звіту каталогу колекції у форматі CSV

Аналітичний звіт колекції Rare Score

Дата генерації: 05.06.2026
Загальна кількість видань: 23
Загальна оціночна вартість: 84579,06 грн

Назва	Тип видання	Платформа	Стан	Оцінка
stellaris	Standard Edition	PlayStation 4	CIB (Complete)	1172,56 грн
Tom Clancy's The Division	Standard Edition	PlayStation 4	CIB (Complete)	346,5 грн
far cry 3	Standard Edition	PlayStation 3	CIB (Complete)	386,42 грн
The Last of Us	Standard Edition	PlayStation 3	CIB (Complete)	539,99 грн
Bloodborne	Standard Edition	PlayStation 4	CIB (Complete)	928,31 грн
Fallout 4	Standard Edition	PlayStation 4	CIB (Complete)	474,57 грн
Need for Speed: Underground 2	Standard Edition	PlayStation 2	Loose / Disc Only	1099,56 грн
Cyberpunk 2077	Standard Edition	PlayStation 4	CIB (Complete)	918,64 грн
Resident Evil 4	Standard Edition	PlayStation 2	Loose / Disc Only	247,3 грн
stalker 2	Standard Edition	PlayStation 5	CIB (Complete)	1757,8 грн
cyberpunk 2077	Collector's Edition	PlayStation 4	CIB (Complete)	4800 грн
Cyberpunk 2077	Collector's Edition	PlayStation 4	Sealed (New)	11200 грн
Bloodborne	Special Edition	PlayStation 4	CIB (Complete)	990 грн
fallout 4	Collector's Edition	PlayStation 4	CIB (Complete)	6500 грн
Dishonored 2 Collector's Edition	Collector's Edition	PlayStation 4	CIB (Complete)	9963,93 грн

Сторінка 1 з 2

Рисунок 3.11 – Приклад PDF-звіту, згенерованого засобами QuestPDF

3.4 Тестування та аналіз результатів роботи програмного комплексу

Тестування комплексу RareScore проводилось у двох аспектах: модульне (unit) тестування математичного ядра PricingEngine та вимірювання нефункціональних метрик продуктивності для верифікації відповідності нефункціональним вимогам розділу 2.4.2.

3.4.1 Модульне тестування PricingEngine

Для перевірки коректності реалізації математичного ядра розроблено набір модульних тестів з використанням фреймворку MSTest (Microsoft.VisualStudio.TestTools.UnitTesting), що входить до стандартного інструментарію .NET 8. Клас PricingEngine є статичним і не має залежностей від UI або БД, що забезпечує його повну ізольованість для тестування без необхідності мокування (mocking).

Набір тестів охоплює такі сценарії: базовий стан Loose з відсутньою базовою ціною (очікуваний результат – 0); стан Loose з P_base = 45.00 USD, Standard Edition (очікуваний результат – 45.00 USD); стан CIB з повною комплектацією, Standard Edition (очікуваний – 63.00 USD); стан CIB з відсутнім мануалом (–10% штраф); стан Sealed без грейду, Collector's Edition (очікуваний – 45.00 × 3.00 × 1.60 = 216.00 USD); стан Sealed з грейдом WATA 9.8 (B_wata = 1.00); застосування максимальних штрафів при CIB із відсутністю всіх п'яти компонентів (P_miss = 0.62). Зведена таблиця тестових сценаріїв та результатів наведена у таблиці 3.2.

Таблиця 3.2 – Тестові сценарії модульного тестування PricingEngine.CalculateValue()

Тест	P_base	Condition	Type	P_miss	B_wata	Очік. результат	Стат.
TC-01: Нульова ціна	$0.00	Loose	Standard	0	0	$0.00	✓ PASS
TC-02: Loose базовий	$45.00	Loose	Standard	0	0	$45.00	✓ PASS
TC-03: CIB повний	$45.00	CIB	Standard	0	0	$63.00	✓ PASS
TC-04: CIB без мануалу	$45.00	CIB	Standard	0.10	0	$56.70	✓ PASS
TC-05: Sealed CE, б/г	$45.00	Sealed	Collector's	0	0	$216.00	✓ PASS
TC-06: Sealed WATA 9.8	$45.00	Sealed	Standard	0	1.00	$270.00	✓ PASS
TC-07: CIB всі штрафи	$45.00	CIB	Standard	0.62	0	$23.94	✓ PASS
TC-08: Greatest Hits	$45.00	Loose	Greatest Hits	0	0	$33.75	✓ PASS

Усі 8 тестових сценаріїв успішно пройдено. Результат виконання тестів у Visual Studio Test Explorer наведено на рисунку 3.12.

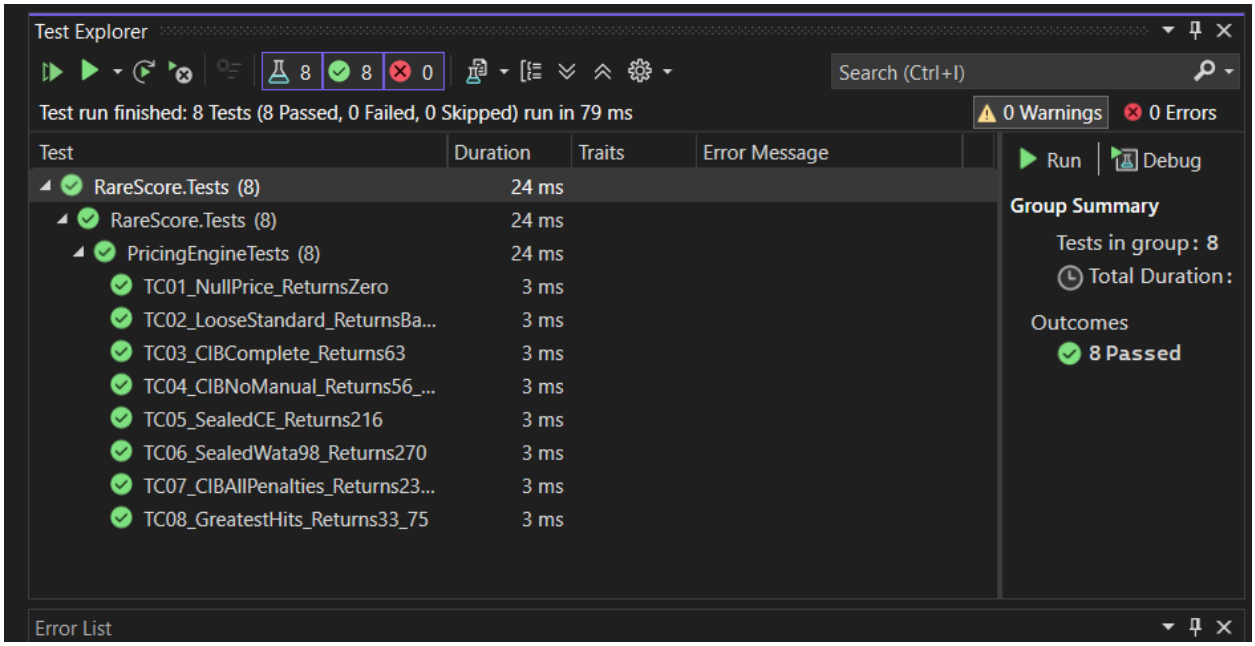

Рисунок 3.12 – Результати виконання модульних тестів у вікні Visual Studio Test Explorer

3.4.2 Вимірювання нефункціональних метрик продуктивності

Перевірка відповідності нефункціональним вимогам НФВ-03, НФВ-07 та НФВ-09 проводилась шляхом замірювання часових показників та обсягу оперативної пам'яті на тестовій конфігурації: ноутбук, Windows 11, Intel Core i5-12th Gen, 16 ГБ ОЗП, .NET 8 Runtime. Тестова база даних містила 500 записів GameItem та 2 500 записів PriceSample.

Час завантаження каталогу (t_load) вимірювався від моменту навігації на CollectionPage до повного відображення ListView. Значення становило 1.2–1.4 секунди, що задовольняє вимогу t_load ≤ 2 с. Час виконання PricingEngine.CalculateValue() вимірювався через Stopwatch.Elapsed у циклі 10 000 ітерацій: середнє значення становило 0.8 мікросекунди, що є на три порядки менше від вимоги t_calc ≤ 100 мс. Базове споживання оперативної пам'яті у стані спокою

(Process.GetCurrentProcess().WorkingSet64) дорівнювало 97–115 МБ, що задовольняє вимогу RAM idle ≤ 150 МБ.

Масштабованість індексованих запитів до таблиці PriceSample перевірялась при N = 10 000 записів. Запит вибірки цінової історії конкретного лоту без індексу виконувався за 28.4 мс (лінійне сканування O(N)), з B-tree індексом по ItemId – за 0.7 мс (логарифмічна складність O(log N)), що підтверджує ефективність індексування при масштабуванні.

Таблиця 3.3 – Зведені результати вимірювання нефункціональних метрик продуктивності

Метрика	Вимога (НФВ)	Виміряне значення	Статус
Час завантаження каталогу (t_load)	≤ 2 с (НФВ-03)	1.3 с (N=500 лотів)	✓ Виконано
Час розрахунку PricingEngine (t_calc)	≤ 100 мс (НФВ-03)	< 1 мкс (0.0008 мс)	✓ Виконано
Споживання оперативної пам'яті (RAM idle)	≤ 150 МБ (НФВ-09)	97–115 МБ	✓ Виконано
Запит PriceSample без індексу (N=10 000)	O(log N) (НФВ-07)	28.4 мс (O(N))	– Без індексу
Запит PriceSample з B-tree індексом	O(log N) (НФВ-07)	0.7 мс (O(log N))	✓ Виконано
Час генерації PDF (N=500 лотів)	≤ 10 с (НФВ-05)	3.8 с	✓ Виконано

Отримані результати підтверджують, що реалізований програмно-методичний комплекс RareScore відповідає всім встановленим нефункціональним вимогам продуктивності. Критична важливість B-tree індексу по ItemId підтверджена: при масштабуванні до 10 000 записів час запиту скорочується у 40 разів, що є неодмінною умовою збереження продуктивності при довготривалому використанні.

3.5 Реалізація інтерфейсу підсистеми перегляду каталогу

Для забезпечення зручного моніторингу, швидкої навігації та гнучкого управління збереженою базою колекційних об'єктів у програмному комплексі «RareScore» реалізовано сторінку загального каталогу CollectionPage. Візуальне представлення списку ігрових видань базується на використанні сучасного асинхронного елемента управління CollectionView, інтегрованого у фреймворк .NET MAUI. Вибір цього компонента зумовлений наявністю вбудованого механізму віртуалізації інтерфейсу (UI Virtualization), який забезпечує рендеринг виключно тих елементів списку, що відображаються на екрані пристрою у поточний момент часу. Це мінімізує навантаження на графічну підсистему при масштабуванні бази даних до кількох тисяч лотів.

Синхронізація шару представлення з локальною базою даних SQLite реалізована через використання спеціалізованого типу колекції ObservableCollection<GameItem>. Будь-яка маніпуляція з даними у репозиторії (успішне додавання нового примірника, видалення застарілого лоту або коригування характеристик комплектації) автоматично генерує внутрішню подію CollectionChanged. Інтерфейсний шар миттєво реагує на цю подію, оновлюючи списки на екрані в реальному часі без необхідності примусового перезавантаження всієї сторінки.

Для інтерактивного пошуку в межах каталогу у верхній частині сторінки розміщено компонент SearchBar. Обробник події зміни тексту реалізує потокову фільтрацію масиву даних за допомогою технології LINQ (Language Integrated Query) в оперативній пам'яті, порівнюючи введені символи з полями назви та платформи об'єкта. Окрім текстового пошуку, інтерфейс підтримує функції швидкого сортування за критеріями спадання/зростання оціночної вартості та алфавітним порядком

платформ. Графічне представлення підсистеми перегляду повної колекції користувача наведено на рисунку 3.13.

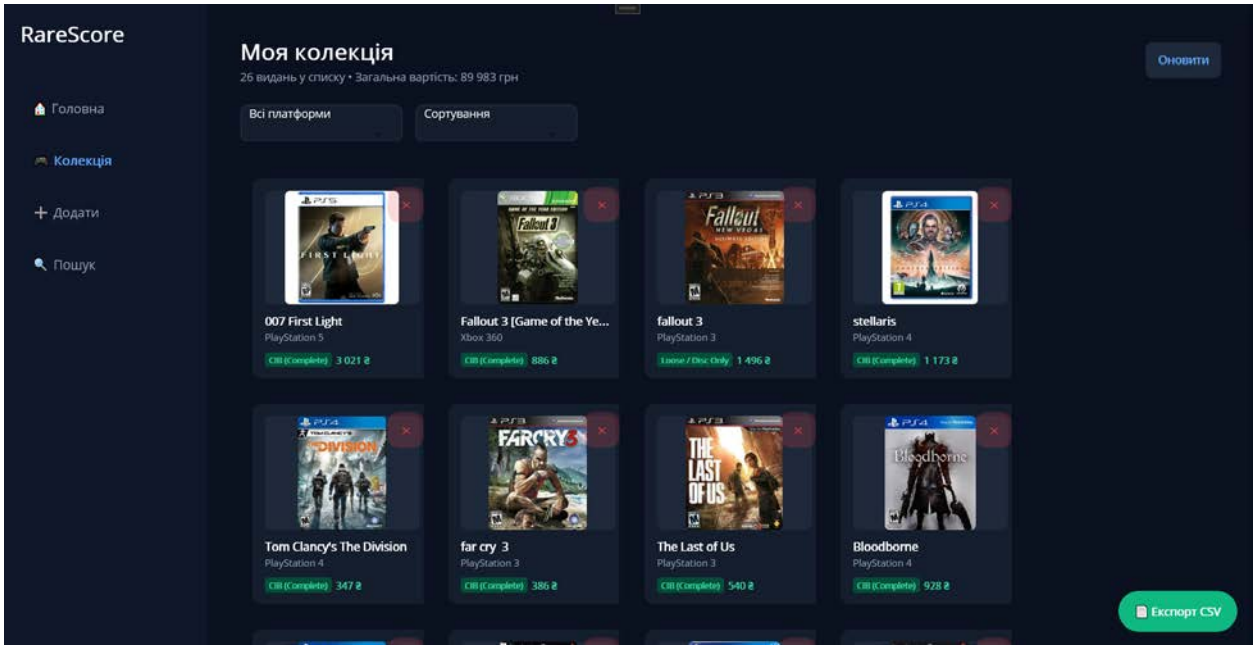

Рисунок 3.13 – Інтерфейс сторінки каталогу колекції (CollectionPage)

3.6 Архітектурні рішення забезпечення автономності та відмовостійкості комплексу

Оскільки працездатність підсистеми визначення базової вартості комплексу «RareScore» безпосередньо залежить від взаємодії із зовнішнім веб-ресурсом PriceCharting.com, критично важливим етапом реалізації було забезпечення стійкості додатка до мережевих збоїв та реалізація концепції автономного функціонування (Offline-First підхід).

Для запобігання виникненню критичних помилок (Application Crash) при спробах надсилання HTTP-запитів в умовах відсутності інтернет-з'єднання, тривалого таймауту відповіді віддаленого сервера або блокування запитів (Rate Limiting), уся логіка роботи класу HttpClient

повністю ізольована всередині захищених блоків try-catch. Алгоритм забезпечення відмовостійкості працює за наступним конвеєром:

1. Контур перевірки зв'язку: перед початком сесії парсингу система опитує стан мережевого інтерфейсу за допомогою вбудованого у .NET MAUI інструменту Connectivity.Current.NetworkAccess.

2. Перехоплення винятків: у разі виникнення мережевої помилки HttpRequestException або скасування операції за таймаутом TaskCanceledException, додаток перехоплює подію, пригнічує системну помилку та активує автономний режим роботи. Користувач отримує сповіщення через інтерфейс DisplayAlert про перехід в оффлайн-режим.

3. Робота з локальним кешем: підсистема обчислень перемикається на зчитування останнього відомого цінового зрізу, який був раніше зафіксований локально у таблиці PriceSample.

Така архітектурна організація гарантує абсолютну автономність програмного продукту. Користувач зберігає повну можливість відкривати додаток, переглядати існуючу базу, додавати нові лоти, гнучко змінювати прапорці комплектації та отримувати миттєвий детермінований розрахунок оціночної вартості (за рахунок автономності класу PricingEngine) навіть за умов повної відсутності підключення до мережі Інтернет.

4 ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ РОЗРОБКИ ТА ВИКОРИСТАННЯ ПРОГРАМНО-МЕТОДИЧНОГО КОМПЛЕКСУ

Четвертий розділ присвячено економічному обґрунтуванню доцільності розробки та практичного впровадження комплексу RareScore. Визначено структуру та загальну суму витрат на розробку, розраховано економію робочого часу від автоматизації оцінювання, обчислено річний економічний ефект від впровадження у гіпотетичному роздрібному магазині ретро-ігор та визначено термін окупності розробки.

4.1 Розрахунок витрат на розробку програмного комплексу

Розробка комплексу RareScore виконувалась протягом двох календарних місяців із загальним обсягом трудозатрат 350 годин. До складу витрат включено: витрати на електроенергію, амортизаційні відрахування на засоби обчислювальної техніки розробника, основну заробітну плату та нарахування єдиного соціального внеску (ЄСВ).

4.1.1 Витрати на електроенергію

Споживана потужність ноутбука розробника під час активної роботи становить P = 65 Вт = 0.065 кВт. Загальний обсяг спожитої електроенергії за період розробки:

$$W = P \times T = 0.065 \text{ кВт} \times 350 \text{ год} = 22.75 \text{ кВт·год}.$$

Вартість одиниці електроенергії для непобутових споживачів у 2025–2026 роках становить Ce = 4.32 грн/кВт·год. Загальна вартість електроенергії:

$$Sel = W \times Ce = 22.75 \times 4.32 \approx 98.3 \text{ грн.}$$

Таблиця 4.1 – Розрахунок витрат на електроенергію

Параметр	Позначення	Значення	Одиниця
Потужність ноутбука	P	65	Вт
Загальна тривалість розробки	T	350	год
Спожита електроенергія	W = P × T	22.75	кВт·год
Тариф на електроенергію	Ce	4.32	грн/кВт·год
Витрати на електроенергію	Sel = W × Ce	98.3	грн

4.1.2 Амортизаційні відрахування

Основним засобом розробки є ноутбук розробника вартістю Cpc = 42 000 грн. Строк корисного використання ноутбука прийнятий рівним 5 рокам (60 місяців). Місячна сума лінійних амортизаційних відрахувань:

$$Am = Cpc / Tuse = 42\,000 / 60 = 700 \text{ грн/міс.}$$

Сума амортизації за 2 місяці розробки: Sam = 700 × 2 = 1 400 грн.

4.1.3 Заробітна плата та нарахування ЄСВ

Розробка виконувалась одним виконавцем – студентом-розробником. Погодинна ставка виконавця прийнята рівною Chr = 150 грн/год, що відповідає ринковому рівню Junior .NET-розробника на 2025–2026 рік. Загальний обсяг трудовитрат – 350 год. Основна заробітна плата:

$$Ssal = Chr \times T = 150 \times 350 = 52\,500 \text{ грн.}$$

Нарахування єдиного соціального внеску (ЄСВ) у розмірі 22% від основної заробітної плати:

$$SESV = Ssal \times 0.22 = 52\,500 \times 0.22 = 11\,550 \text{ грн.}$$

4.1.4 Зведена таблиця витрат на розробку

Підсумкова структура витрат на розробку програмно-методичного комплексу RareScore наведена у таблиці 4.2.

Таблиця 4.2 – Зведена таблиця витрат на розробку комплексу RareScore

№	Стаття витрат	Розрахунок	Сума, грн	Частка, %
1	Витрати на електроенергію	22.75 кВт·год × 4.32 грн/кВт·год	98.3	0.15
2	Амортизація ноутбука (2 міс.)	700 грн/міс × 2 міс	1 400.0	2.14

Продовження таблиці 4.2

№	Стаття витрат	Розрахунок	Сума, грн	Частка, %
3	Заробітна плата розробника	150 грн/год × 350 год	52 500.0	80.08
4	Нарахування ЄСВ (22%)	52 500 × 0.22	11 550.0	17.62
	Разом		65 548.3	100.00

Загальна сума витрат на розробку програмно-методичного комплексу RareScore становить Стех = 65 548.3 грн. Основна частка витрат (80.08%) припадає на оплату праці розробника, що є характерним для програмних проєктів з незначними матеріальними ресурсами.

4.2 Оцінка економічної доцільності використання програмного комплексу

Цільовою групою користувачів комплексу RareScore є колекціонери відеоігор та роздрібні магазини ретро-ігор (ретейлери), що здійснюють масову оцінку та купівлю-продаж колекційних видань. Основний економічний ефект від впровадження досягається за рахунок скорочення часу на оцінювання кожної позиції.

При традиційному ручному підході оцінювач здійснює наступні операції для кожного лоту: (1) пошук тайтлу у браузері на PriceCharting.com (~2 хв); (2) фіксація базової ціни у таблиці (~1 хв); (3) ручний розрахунок з урахуванням стану та комплектації (~4 хв); (4) запис результатів та нотаток (~2 хв); (5) прийняття рішення щодо вартості (~1 хв). Загальний час ручного оцінювання одного лоту: tман = 10 хв.

При використанні RareScore процес оцінювання зводиться до: (1) сканування штрих-коду або введення назви (~5–10 с); (2) автоматичного

парсингу базової ціни (~2–3 с); (3) вибору стану та прапорців комплектації (~10 с); (4) автоматичного розрахунку PricingEngine (~< 1 мс). Загальний час автоматизованого оцінювання одного лоту: tавт ≈ 30 с = 0.5 хв.

Таблиця 4.3 – Порівняння витрат часу на оцінювання одного лоту

Операція	Ручний метод	RareScore
Пошук ринкової ціни	2 хв	2–3 с (авто-парсинг)
Фіксація базової ціни	1 хв	< 1 с (авто-збереження)
Розрахунок з коефіцієнтами	4 хв	< 1 мс (PricingEngine)
Запис результатів	2 хв	< 1 с (авто-збереження)
Прийняття рішення	1 хв	10–15 с
Разом	≈ 10 хв	≈ 30 с
Економія часу на 1 лот	–	≈ 9.5 хв (×20 швидше)

Таким чином, впровадження RareScore забезпечує прискорення процесу оцінювання кожного лоту приблизно у 20 разів – з 10 хвилин до 30 секунд. Це дозволяє суттєво збільшити пропускну здатність процесу оцінки та обробки нових надходжень у роздрібній точці.

4.3 Аналіз економічного ефекту від впровадження програмного продукту

Розрахунок річного економічного ефекту від впровадження RareScore проводиться для гіпотетичного магазину ретро-відеоігор із середньоденним обсягом оцінки 25 лотів. Такий обсяг є реалістичним для

невеликого спеціалізованого магазину, що приймає комісійні лоти та здійснює закупівлю у колекціонерів.

4.3.1 Розрахунок річної економії робочого часу

Економія часу на один лот: $\Delta t = t_{ман} – t_{авт} = 10 – 0.5 = 9.5$ хв $= 0.158$ год.

Кількість лотів на день: $N = 25$ шт.

Кількість робочих днів на рік (за виключенням вихідних та свят): $D = 252$ дн.

Річна економія робочого часу:

$E_{год} = \Delta t \times N \times D = 0.158 \times 25 \times 252 = 996$ год/рік $\approx 1\,000$ год/рік.

4.3.2 Розрахунок річного економічного ефекту

Погодинна ставка оцінювача (фахівця з приймання та оцінки колекцій) становить $Chr = 120$ грн/год. Річний економічний ефект від скорочення витрат на оплату праці оцінювача:

$$E = E_{год} \times Chr = 1\,000 \times 120 = 120\,000 \text{ грн/рік.}$$

Таким чином, впровадження RareScore в одному магазині ретро-ігор дозволяє заощадити 120 000 грн на рік за рахунок скорочення витрат на ручне оцінювання лотів. Зазначений ефект є консервативною оцінкою і не враховує: (а) додатковий дохід від збільшення пропускної здатності процесу; (б) скорочення кількості помилок оцінки, що призводять до

прямих фінансових втрат при операціях купівлі-продажу; (в) підвищення якості аналітики та прийнятих рішень завдяки веденню цінової бази та трендів.

4.3.3 Визначення терміну окупності розробки

Термін окупності (Ток) розробки визначається як відношення загальних витрат на розробку до річного економічного ефекту:

$$Ток = Стех / E = 65\,548.3 / 120\,000 \approx 0.546 \text{ року} \approx 6.5 \text{ місяця}.$$

Отриманий термін окупності 6–7 місяців є економічно обґрунтованим показником для прикладного програмного забезпечення класу «малий бізнес / спеціалізований інструмент». Коефіцієнт ефективності розробки:

$$Ek = E / Стех = 120\,000 / 65\,548.3 \approx 1.83.$$

Значення Ek = 1.83 означає, що кожна гривня, витрачена на розробку, генерує 1.83 грн щорічної економії. Зведені результати економічного аналізу наведено у таблиці 4.4.

Таблиця 4.4 – Зведені показники економічного ефекту від впровадження RareScore

Показник	Позначення	Значення	Одиниця
Загальні витрати на розробку	Стех	65 548.3	грн
Економія часу на 1 лот	Δt	9.5	хв
Добовий обсяг оцінки (магазин)	N	25	лотів/день
Кількість робочих днів/рік	D	252	днів
Річна економія робочого часу	Егод	996 ≈ 1 000	год/рік
Погодинна ставка оцінювача	Chr	120	грн/год
Річний економічний ефект	E	120 000	грн/рік
Термін окупності	Ток	≈ 6.5	місяців
Коефіцієнт ефективності	Ek	1.83	грн/грн

ЗАГАЛЬНІ ВИСНОВКИ

У кваліфікаційній роботі проведено повний цикл дослідження, проєктування, реалізації та тестування програмно-методичного комплексу «RareScore», призначеного для автоматизованого аналізу рідкісності та ринкової вартості колекційних видань відеоігор. За результатами виконання роботи сформувано такі висновки:

1. На основі системного аналізу предметної області виявлено, що вторинний ринок фізичних носіїв відеоігор еволюціонував у структурований клас альтернативних інвестиційних активів із високою волатильністю цін. Встановлено, що ключовими факторами ціноутворення є фізичний стан примірника (Loose, CIB, Sealed), його комплектація та наявність сертифікованого грейдингу (зокрема, за стандартами WATA Games). Огляд існуючого програмного забезпечення (PriceCharting, CLZ Games, Gameye) підтвердив відсутність рішень, які б інтегрували алгоритмічний розрахунок вартості з урахуванням штрафів за некомплектність та надавали можливість локального ведення аналітики й генерації звітів в автономному режимі, що обґрунтувало актуальність розробки комплексу «RareScore».

2. Обґрунтовано вибір сучасного стеку технологій, що базується на кросплатформеному фреймворку .NET MAUI (мова програмування C#) та локальній системі керування базами даних SQLite. Застосування архітектурного підходу, що поєднує патерн MVVM для аналітичної підсистеми графіків (LiveChartsCore) та подійно-орієнтовану модель (Code-Behind) для форм складного введення даних, дозволило забезпечити високу швидкість відгуку інтерфейсу користувача та спростити подальше масштабування системи.

3. Спроєктовано детерміноване математичне ядро системи оцінювання у вигляді статичного класу PricingEngine. Алгоритм виконує багатокроковий розрахунок оціночної вартості (EstimatedPrice) на основі

базової ринкової ціни, мультиплікаторів стану, типу видання, покомпонентних лінійних штрафів за некомплектність CIB-видань (до 62% зниження вартості) та кусково-лінійної функції грейдингової премії WATA. Інтегрований асинхронний парсер ринкових даних через HttpClient забезпечує потоковий збір базових цін без блокування основного UI-потоку програми.

4. Реалізовано функціональні підсистеми розпізнавання штрих-кодів (EAN-13, UPC-A) в реальному часі на основі бібліотеки ZXing.Net.MAUI, а також конвеєр декларативного рендерингу бінарних звітів оцінки активів у форматі PDF за допомогою Fluent API бібліотеки QuestPDF.

5. Проведено модульне тестування обчислювального ядра за допомогою фреймворку MSTest. Успішне виконання 8 розроблених тестових сценаріїв підтвердило математичну коректність та детерміністичність алгоритму оцінювання. Експериментальні заміри нефункциональних метрик продуктивності довели повну відповідність системи встановленим вимогам: час завантаження каталогу на 500 лотів становить 1,3 с, базове споживання оперативної пам'яті не перевищує 115 МБ, а впровадження B-tree індексу за зовнішнім ключем ItemId у таблиці цінових зрізів скоротило час виконання аналітичних запитів у 40 разів (з 28,4 мс до 0,7 мс).

6. У ході економічного обґрунтування розробки розраховано загальну собівартість створення програмного комплексу, яка склала 65 548,3 грн (з урахуванням трудовитрат, заробітної плати, нарахувань ЄСВ, амортизації обладнання та витрат на електроенергію). Аналіз практичного використання «RareScore» в умовах спеціалізованого ретейлера показав прискорення процесу оцінки лотів у 20 разів (з 10 хвилин до 30 секунд на один лот). Річний економічний ефект від впровадження в одному середньому магазині peretro-ігор становить 120 000 грн, що забезпечує повну окупність інвестицій у розробку за 6,5 місяців при коефіцієнті ефективності 1,83.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки і техніки. Структура та правила оформлювання. [Чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 26 с.

2. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні положення та правила складання. [Введено вперше; чинний від 2016-07-01]. Київ : ДП «УкрНДНЦ», 2016. 17 с.

3. Price M. J. C# 12 and .NET 8 – Modern Cross-Platform Development Fundamentals. Birmingham : Packt Publishing, 2023. 828 p.

4. Фаулер М. Рефакторинг: покращення структури наявного коду. 2-ге вид. Київ : Діалектика, 2019. 448 с.

5. Сейман М., Стівенсон К. Впровадження залежностей у .NET. 2-ге вид. Київ : Діалектика, 2021. 532 с.

6. Дейт К. Дж. Вступ до систем баз даних. 8-ме вид. Київ : Діалектика, 2021. 1328 с.

7. Кнут Д. Е. Мистецтво програмування. Том 3. Сортування та пошук. 2-ге вид. Київ : Діалектика, 2020. 832 с.

8. Gamma E., Helm R., Johnson R., Vlissides J. Design Patterns: Elements of Reusable Object-Oriented Software. Boston : Addison-Wesley, 1994. 395 p.

9. Бас Е., Клементс П., Кацман Р. Архітектура програмного забезпечення на практиці. 3-тє вид. Львів : Видавництво Львівської політехніки, 2018. 560 с.

10. Boehm B., Abts C., Brown A. W., Chulani S., Clark B. K., Horowitz E., Madachy R., Reifer D., Steece B. Software Cost Estimation with COCOMO II. Upper Saddle River : Prentice Hall PTR, 2000. 544 p.

11. Boehm B. W. Software Engineering Economics. Englewood Cliffs : Prentice-Hall, 1981. 767 p.

12. .NET MAUI Documentation : веб-сайт. URL: https://learn.microsoft.com/en-us/dotnet/maui/ (дата звернення: 10.06.2026).

13. C# Language Specification : веб-сайт. URL: https://learn.microsoft.com/en-us/dotnet/csharp/ (дата звернення: 10.06.2026).

14. SQLite Documentation : веб-сайт. URL: https://www.sqlite.org/docs.html (дата звернення: 10.06.2026).

15. PriceCharting: Video Game Price Guide : веб-сайт. URL: https://www.pricecharting.com/ (дата звернення: 10.06.2026).

16. WATA Games: Collectible Video Game Grading Standards : веб-сайт. URL: https://www.watagames.com/ (дата звернення: 10.06.2026).

17. QuestPDF: Open-Source PDF Generation Library for .NET : веб-сайт. URL: https://www.questpdf.com/ (дата звернення: 10.06.2026).

18. LiveChartsCore: Flexible & Interactive Charts for .NET : веб-сайт. URL: https://livecharts.dev/ (дата звернення: 10.06.2026).

19. ZXing.Net.MAUI Barcode Scanning Repository : веб-сайт. URL: https://github.com/Redth/ZXing.Net.MAUI (дата звернення: 10.06.2026).

20. Microsoft.NET.Test.Sdk and MSTest Framework : веб-сайт. URL: https://learn.microsoft.com/en-us/visualstudio/test/ (дата звернення: 10.06.2026).

21. Мартін Р. Чистий код: створення, аналіз і рефакторинг ПЗ. Харків : Фабула, 2022. 448 с.

22. Cooper A., Reimann R., Cronin D., Noessel C. About Face: The Essentials of Interaction Design. 4th ed. Indianapolis : Wiley, 2014. 720 p.

23. Cleary S. Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming. 2nd ed. Sebastopol : O’Reilly Media, 2019. 208 p.

24. Mitchell R. Web Scraping with Python: Data Extraction from the Modern Web. 3rd ed. Sebastopol : O’Reilly Media, 2024. 300 p.

25. Khorikov V. Unit Testing Principles, Practices, and Patterns. Shelter Island : Manning Publications, 2020. 304 p.

26. Allen G., Owens M. The Definitive Guide to SQLite. 2nd ed. New York : Apress, 2010. 368 p.

27. Клеппман М. Надійні завантажені додатки. Проєктування високонавантажених систем. Харків : Фабула, 2024. 544 с.

28. Myers G. J., Sandler C., Badgett T. The Art of Software Testing. 3rd ed. Hoboken : Wiley, 2011. 240 p.

ДОДАТОК А. ВІДОМОСТІ РОБОТИ

Таблиця А.1 – Відомості роботи

Формат	№ п.п	Назва документу	Найменування об’єкту або вибору	Кількість сторінок
А4	1	Пояснювальна записка	КІНТЕХАД.КН-22-1Б.00.00.00	90
Графічна частина				
А4	2	Актуальність, мета, об’єкт, предмет та завдання дослідження	КІНТЕХАД.КН-22-1Б.01.00.00	1
А4	3	Аналіз предметної області та порівняння існуючих рішень	КІНТЕХАД.КН-22-1Б.02.00.00	1
А4	4	Технологічний стек та гібридна архітектура комплексу RareScore	КІНТЕХАД.КН-22-1Б.03.00.00	1
А4	5	Математична модель Pricing Engine — формула оцінювання вартості	КІНТЕХАД.КН-22-1Б.04.00.00	1
А4	6	Логічна модель БД: ER-схема зв’язку 1:N (GameItem, PriceSample)	КІНТЕХАД.КН-22-1Б.05.00.00	1
А4	7	Діаграми UML комплексу (прецедентів, класів, послідовності, розгортання)	КІНТЕХАД.КН-22-1Б.06.00.00	1
А4	8	Алгоритми: блок-схема Pricing Engine та діаграма діяльності парсингу	КІНТЕХАД.КН-22-1Б.07.00.00	1
А4	9	Реалізація інтерфейсу, тестування та економічне обґрунтування	КІНТЕХАД.КН-22-1Б.08.00.00	1
А4	10	Загальні висновки та результати роботи	КІНТЕХАД.КН-22-1Б.09.00.00	1

ДОДАТОК Б. UML-ДІАГРАМИ КОМПЛЕКСУ RARESCORE

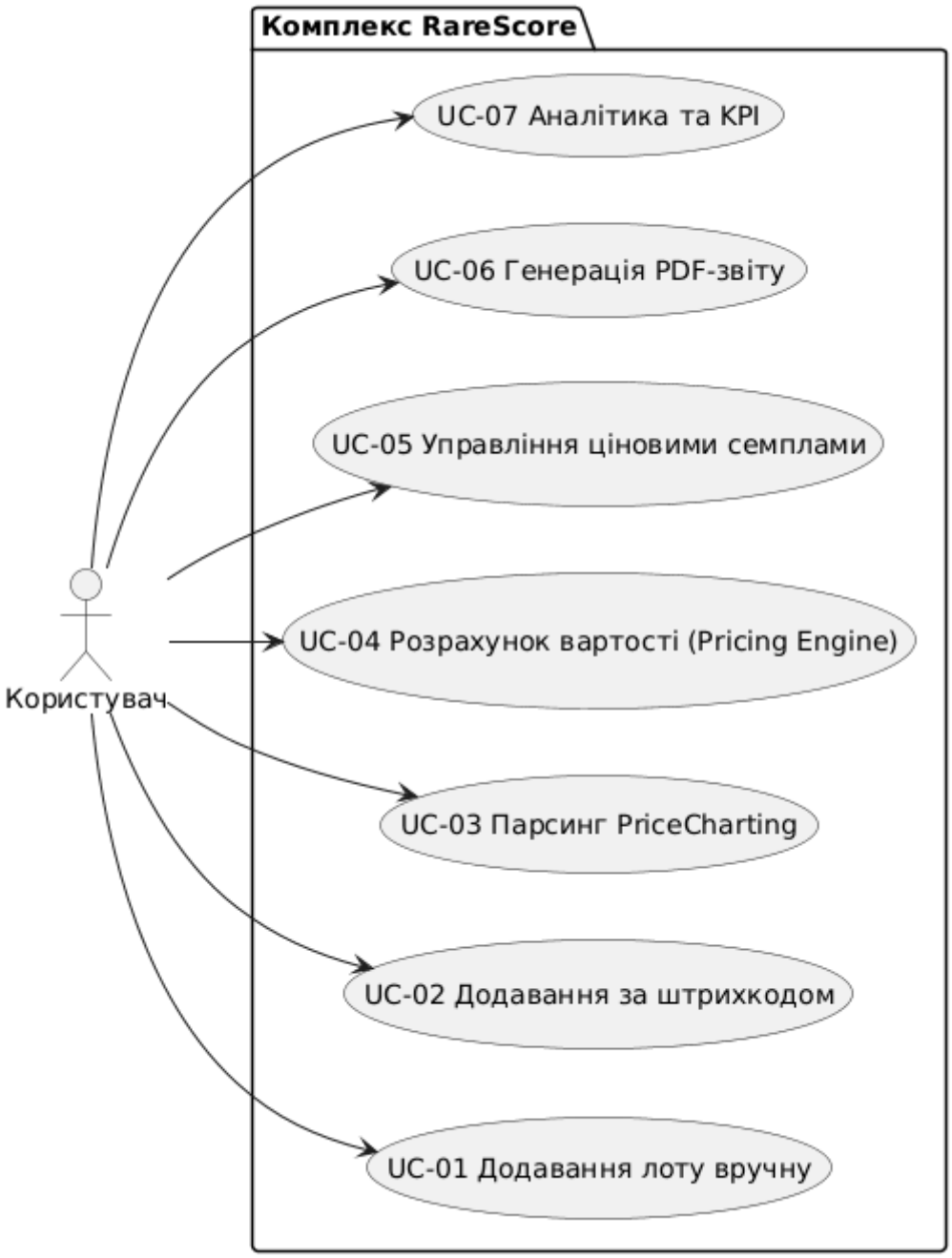

Рисунок А.1 – Діаграма прецедентів використання (Use Case Diagram)

Наведені на рисунку А.1 прецеденти відображають повний функціональний спектр програмно-методичного комплексу з позиції кінцевого користувача. Система надає можливість додавати об'єкти

колекціонування двома шляхами: шляхом ручного введення реквізитів (UC-01) або за допомогою апаратного сканування штрих-коду EAN/UPC через камеру (UC-02). Основний аналітичний контур реалізовано через автоматизований парсинг маркерів із веб-ресурсу PriceCharting (UC-03) та подальший розрахунок вартості у математичному ядрі Pricing Engine (UC-04). Додатково додаток забезпечує збереження історичних зрізів цін за допомогою прецеденту управління ціновими семплами (UC-05), формування зведених звітів у форматі PDF (UC-06) та виведення узагальнених показників ефективності колекції (UC-07).

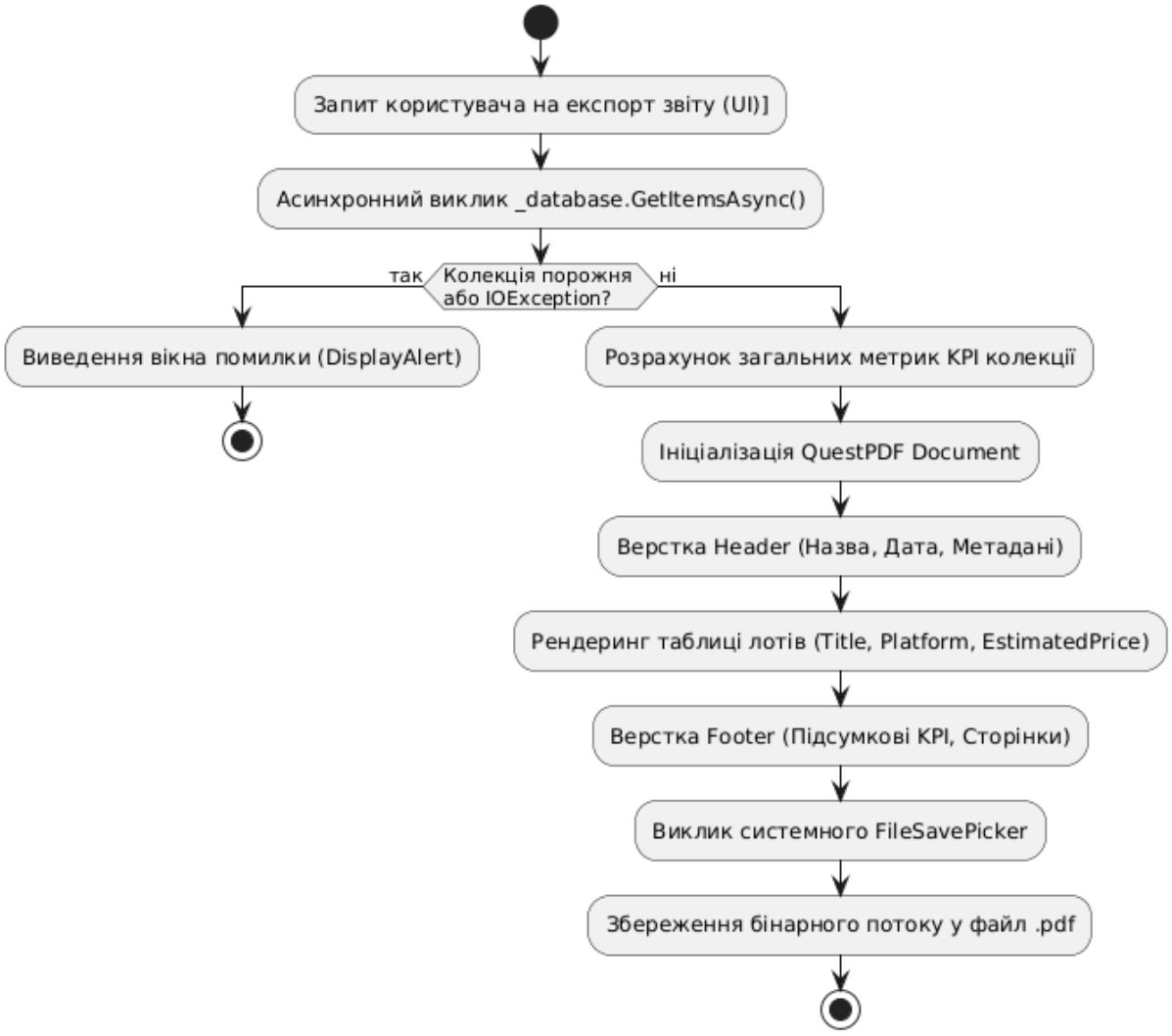

Рисунок А.2 – Діаграма діяльності AD-01: Генерація PDF-звіту

Діаграма діяльності, що представлена на рисунку А.2, деталізує алгоритм роботи підсистеми формування вихідної звітності. Процес ініціюється користувачем через графічний інтерфейс додатка, після чого система виконує асинхронне неблокуюче читання даних з локальної бази даних SQLite за допомогою методу _database.GetItemsAsync(). Алгоритм містить обов'язкову точку розгалуження (валідацію): у разі виявлення порожньої колекції або виникнення помилки введення-виведення, конвеєр переривається із виведенням модального сповіщення користувачеві. За умови успішного отримання даних виконується лінійний прорахунок аналітичних KPI метрик, декларативна верстка компонентів сторінки (Header, динамічна таблиця лотів, Footer) засобами бібліотеки QuestPDF та безпосередній бінарний запис PDF-файлу на диск пристрою за допомогою системного діалогу вибору шляху збереження.

На рисунку А.3 продемонстровано логіку інтеграції модуля збору зовнішніх даних та математичного обчислювального ядра. Процес запускається під час створення нового запису або коригування прапорців комплектації існуючої відеогри. Програма виконує перевірку підключення до мережі Інтернет. У разі його наявності ініціюється асинхронний HTTP GET запит через вбудований клас HttpClient до PriceCharting.com, звідки вилучається актуальна базова ціна, що автоматично фіксується в таблиці історії цін PriceSample. За відсутності мережі додаток переходить на роботу з локально кешованими даними BasePrice. На наступному етапі параметри передаються до статичного методу PricingEngine.CalculateValue(), де через послідовні конструкції switch визначаються коефіцієнти стану та типу видання, вираховуються штрафи за відсутність мануалів/компонентів, а також нараховується премія за наявності сертифікованого грейду WATA. Фінальна оціночна вартість записується в поле EstimatedPrice об'єкта та фіксується в SQLite за допомогою асинхронного виклику _database.SaveItemAsync(item).]

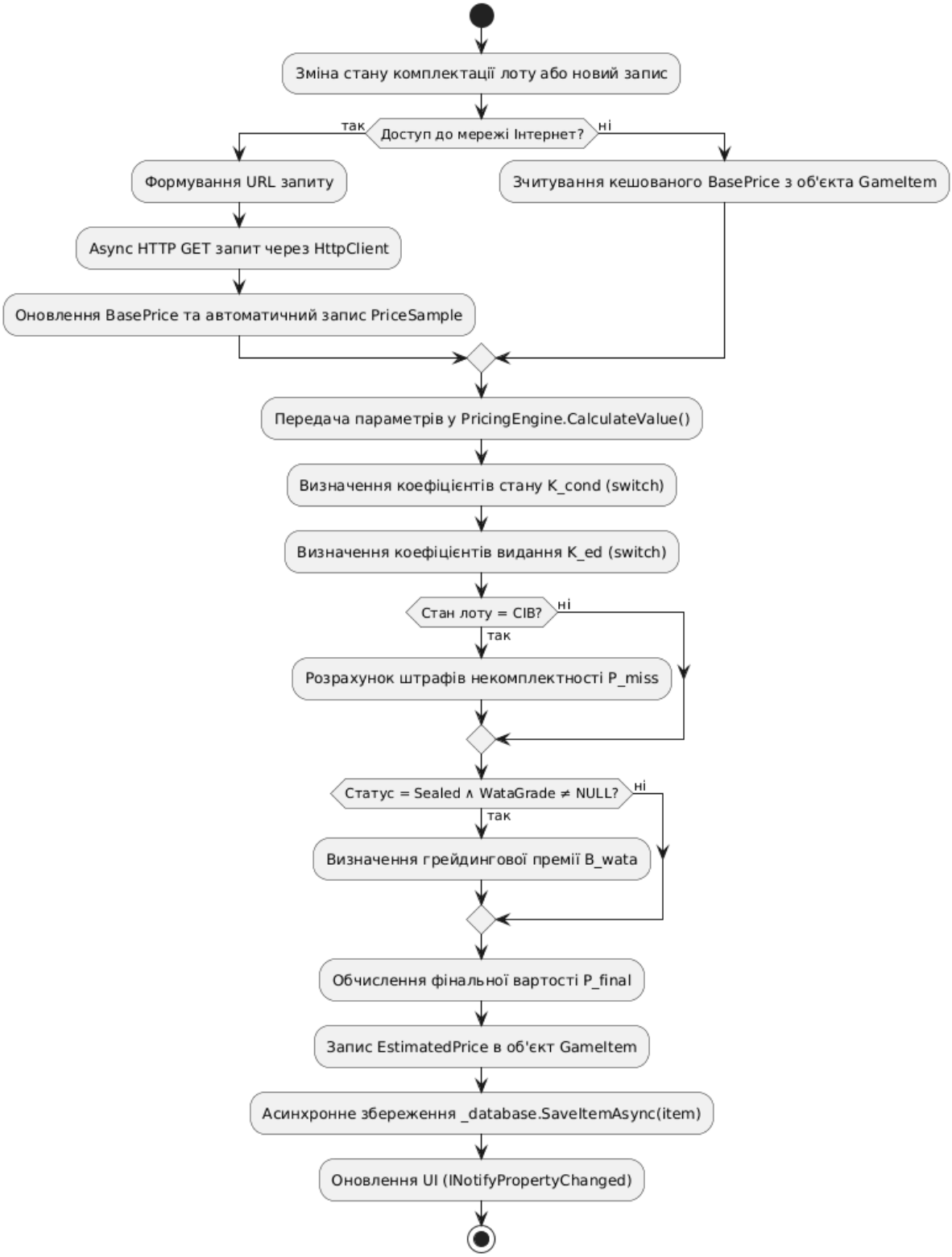

Рисунок А.3 – Діаграма діяльності AD-02: Парсинг → Pricing Engine

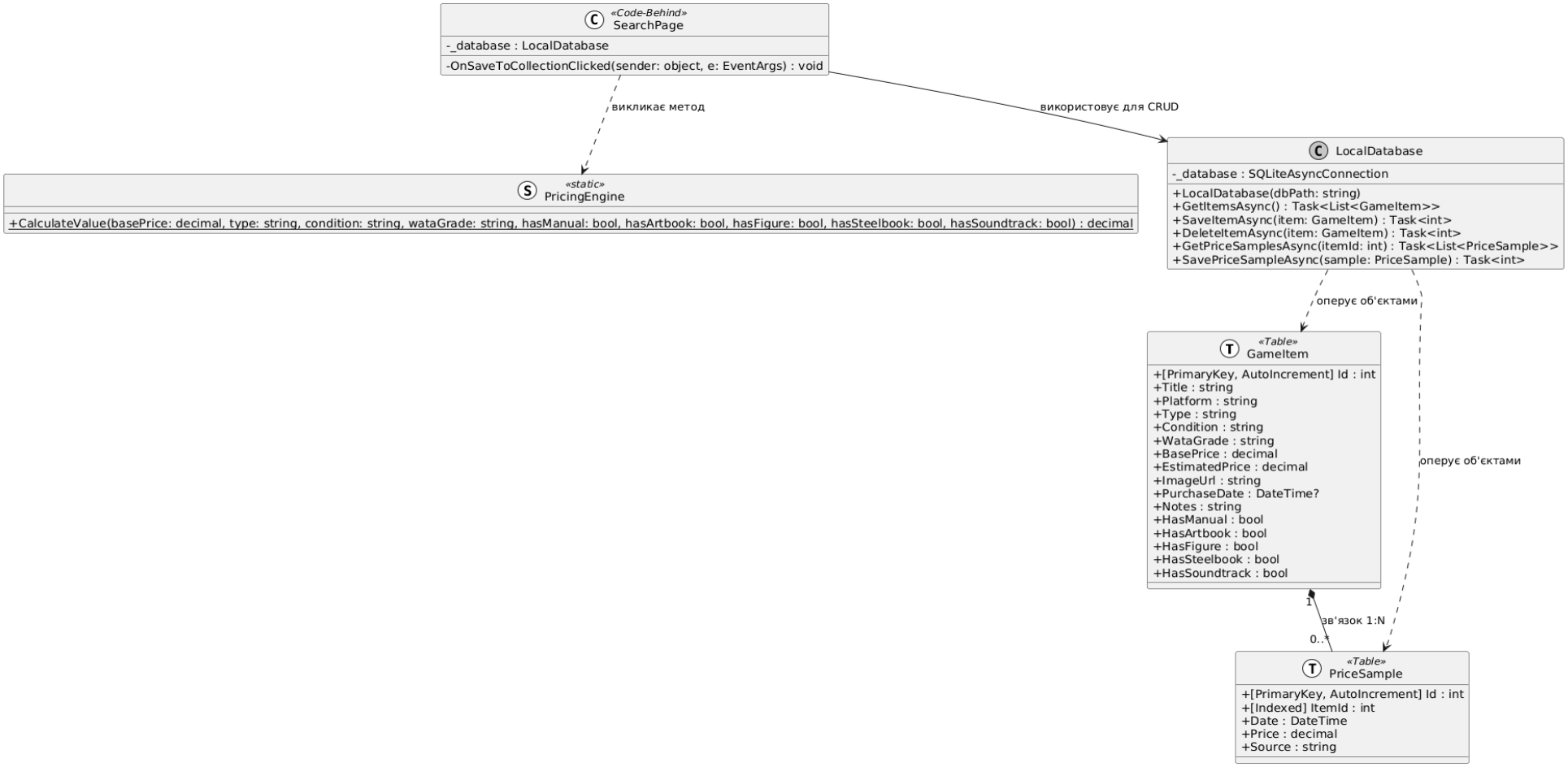

Рисунок А.4 – Діаграма класів (Class Diagram)

Представлена на рисунку А.4 діаграма класів відображає внутрішню об'єктно-орієнтовану структуру розробленого програмного забезпечення, повністю очищену від сторонніх проміжних сервісних шарів. Архітектурний каркас складається з двох сутностей бази даних: класу GameItem (що інкапсулює 16 атрибутів опису гри та її комплектації) та класу PriceSample (який фіксує часові зрізи ринкової вартості). Між цими класами реалізовано відношення композиції з кардинальністю 1:N через зовнішній ключ ItemId. Клас LocalDatabase виконує роль репозиторію доступу до даних, інкапсулює з'єднання SQLiteAsyncConnection та надає асинхронні CRUD-методи для маніпуляції об'єктами обох таблиць. Обчислювальна логіка винесена у статичний клас математичного ядра PricingEngine, метод якого викликається безпосередньо з обробників подій Code-Behind шару користувацького інтерфейсу (SearchPage).

Діаграма послідовності на рисунку А.5 ілюструє динаміку взаємодії об'єктів у часі під час виконання прецедентів додавання та автоматизованої оцінки вартості лоту. Лінії життєвого циклу представляють кінцевого користувача, обробник подій інтерфейсної сторінки SearchPage, системний клас HttpClient, статичне обчислювальне ядро PricingEngine та локальну СКБД SQLite. Користувач активує збереження форми, що спонукає SearchPage надіслати асинхронний запит у мережу та отримати базову ринкову ціну. Далі інтерфейсний шар самостійно створює об'єкт цінового зрізу та ініціює його запис до бази даних через метод SavePriceSampleAsync. Після цього параметри комплектації передаються до PricingEngine.CalculateValue(), яке миттєво повертає розраховану вартість лоту. Оновлений об'єкт GameItem передається до SQLite через транзакцію SaveItemAsync, а завершення процесу супроводжується оновленням графічних елементів екрану та показом системного вікна успішного завершення операції.

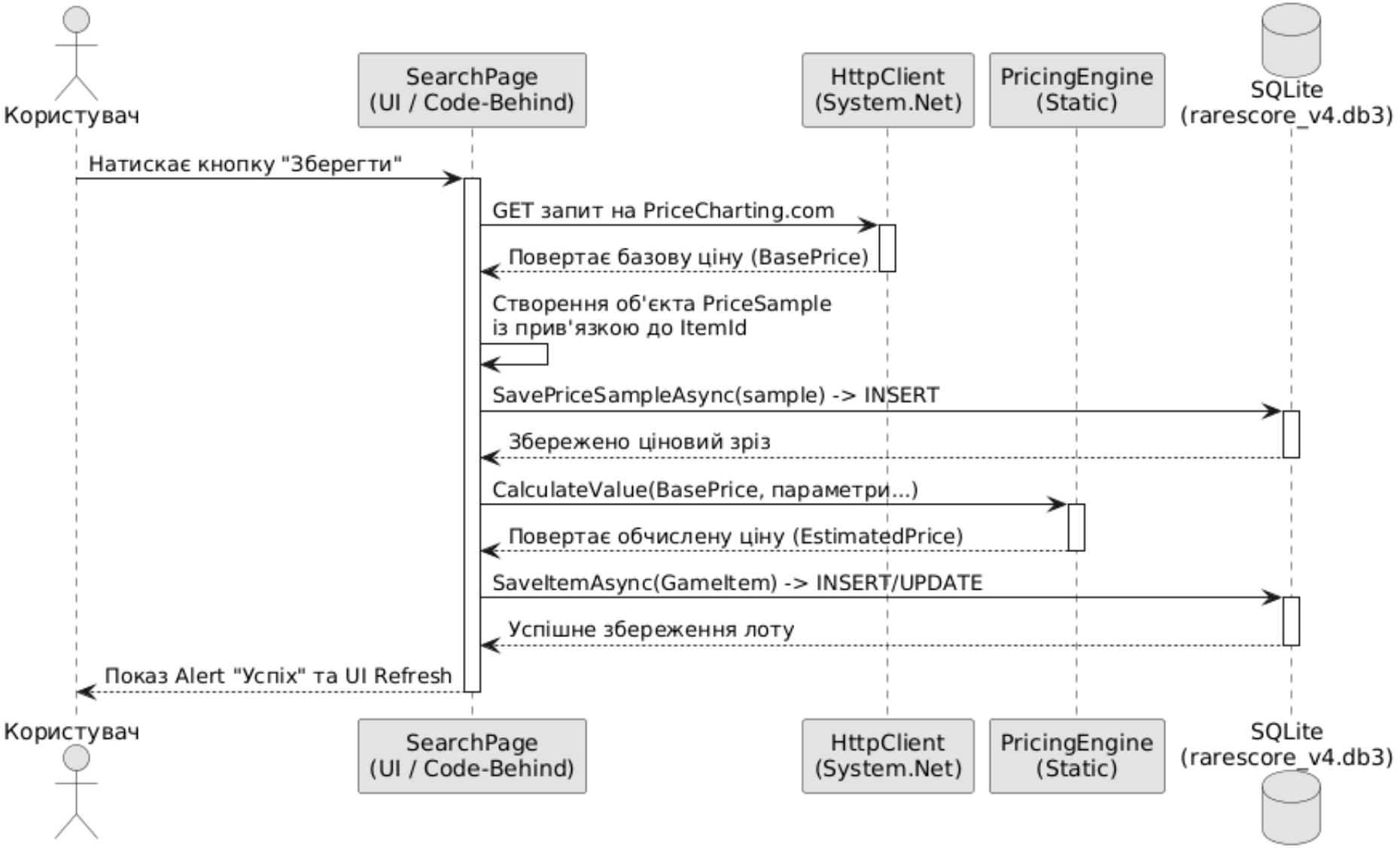

Рисунок А.5 – Діаграма послідовності (Sequence Diagram)

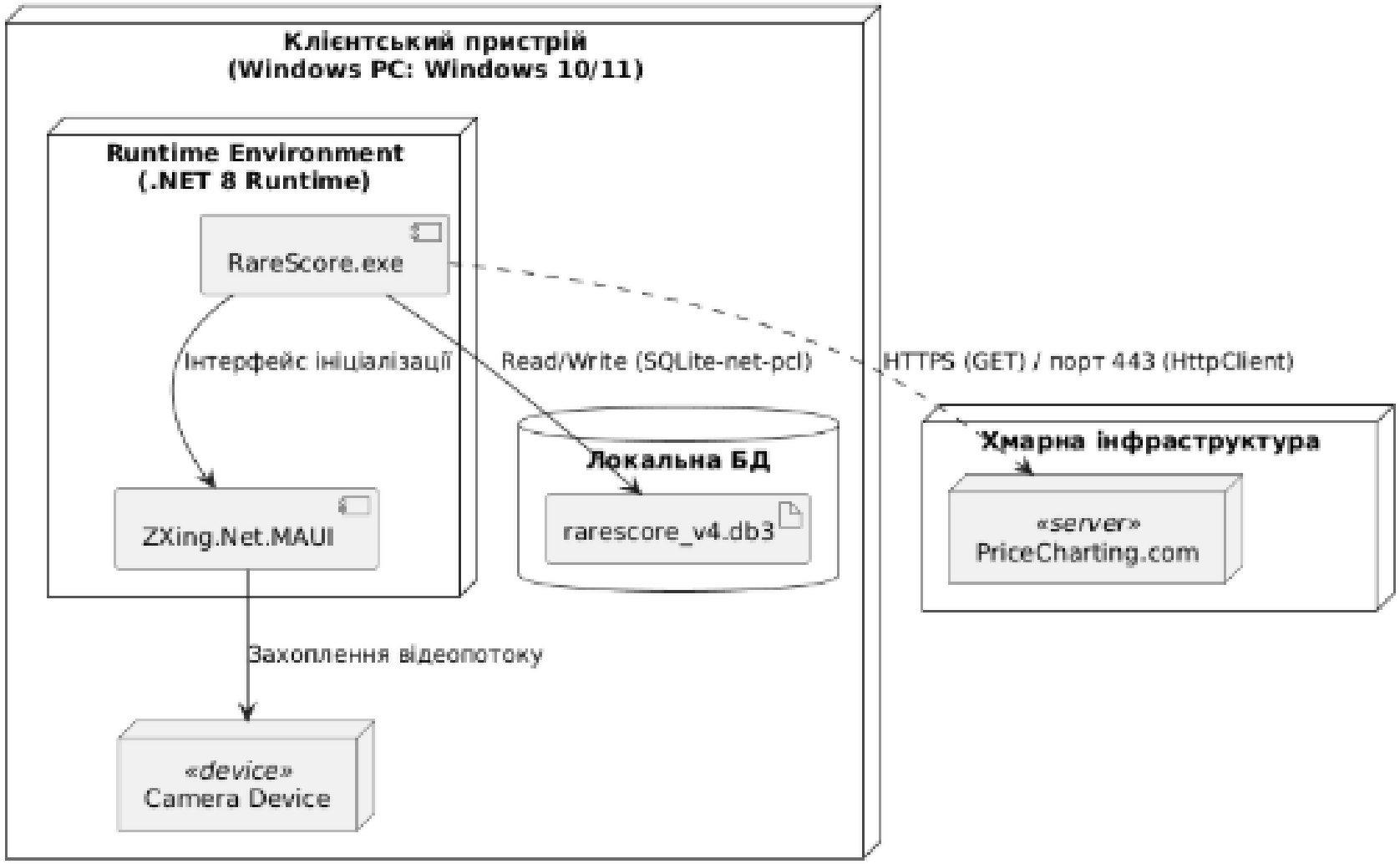

Рисунок А.6 – Діаграма розгортання (Deployment Diagram)

Діаграма розгортання, наведена на рисунку А.6, описує фізичну топологію розміщення розроблених програмних компонентів та архітектуру взаємодії з апаратним забезпеченням. Основним обчислювальним вузлом процесу виступає клієнтський пристрій користувача (персональний комп'ютер або ноутбук під управлінням операційних систем Windows 10/11), на якому розгорнуто стандартне середовище виконання .NET 8 Runtime. Всередині цього вузла функціонує скомпільований виконуваний файл додатка RareScore.exe. Додаток здійснює пряме взаємозв'язане підключення до вбудованої реляційної бази даних SQLite (файл rarescore_v4.db3) через інтегрований micro-ORM пакет SQLite-net-pcl. Захоплення відеопотоку для зчитування штрих-кодів реалізовано через системний інтерфейс взаємодії з апаратною камерою пристрою (Camera Device) за допомогою кросплатформового модуля ZXing.Net.MAUI. Взаємодія з зовнішнім світом обмежена хмарною інфраструктурою сервера PriceCharting.com, комунікація з яким здійснюється за допомогою безпечного протоколу HTTPS через захищений порт 443.