Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук
(повна назва)
Кафедра Штучного інтелекту
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

ШІ-експерт з нумізматичної продукції
(тема)

Виконав:
здобувач четвертого року навчання,
групи ІТШІ-21-5
Родіон Гармаш
(власне ім'я, прізвище)
Спеціальність 122 Комп'ютерні науки
(код і повна назва спеціальності)
Тип програми освітньо-професійна
Освітня програма Штучний інтелект
(повна назва освітньої програми)
Керівник ас. Ірина Малєєва
(посада, власне ім'я, прізвище)

Допускається до захисту

Завідувач кафедри ШІ _____ Олег ЗОЛОТУХІН
(підпис) (власне ім'я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет Комп'ютерних наук

Кафедра Штучного інтелекту

Рівень вищої освіти перший (бакалаврський)

Спеціальність 122 Комп'ютерні науки
(код і повна назва)

Тип програми освітньо-професійна

Освітня програма Штучний інтелект
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«_____» _____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Гармашу Родіону Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи ШІ-експерт з нумізматичної продукції

затверджена наказом університету від 19 травня 20 25 р. № 378Ст

2. Термін подання студентом роботи до екзаменаційної комісії 18 червня 20 25 р.

3. Вихідні дані до роботи ASP.NET Core MVC, Entity Framework Core, PostgreSQL, HTML, CSS, JavaScript, Visual Studio 2022, Python, інформація з відкритих джерел про монети та їх зображення

4. Перелік питань, що потрібно опрацювати в роботі
1) Аналіз предметної галузі
2) Теоретичні відомості обраних технологій
3) Проектування та розробка вебдодатку
4) Дослідження розробленого вебдодатку

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Отримання завдання на кваліфікаційну роботу	19.05.2025	виконано
2	Аналіз предметної галузі	20.05 – 22.05	виконано
3	Постановка задачі	23.05 – 25.05	виконано
4	Обґрунтування вибору використаних технологій	26.05 – 29.05	виконано
5	Програмна реалізація	30.05 – 04.06	виконано
6	Оформлення пояснювальної записки	05.06 – 13.06	виконано
7	Підготовка до захисту	14.06 – 17.06	виконано
8	Захист перед ЕК	18.06.2025	виконано

Дата видачі завдання 19 травня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ ас. Ірина Малєєва
(підпис) (посада, власне ім’я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 88 с., 39 рис., 3 табл., 3 дод., 20 джерел.

ВЕБДОДАТОК, НУМІЗМАТИКА, ШТУЧНИЙ ІНТЕЛЕКТ, ASP.NET CORE MVC, CSS, ENTITY FRAMEWORK CORE, HTML, JAVASCRIPT, POSTGRESQL, RAZOR PAGES.

Об’єкт дослідження – вебдодаток для аналітичного супроводу нумізматики, його ринку та учасників.

Предмет дослідження – розробка та реалізація інтелектуального вебдодатку на основі технологій ASP.NET Core MVC та Entity Framework Core для аналізу, оцінки та прогнозу вартості монет.

Мета роботи – створити інтелектуальну інформаційну систему, яка забезпечить зручний інтерфейс і сучасні аналітичні інструменти для роботи з нумізматичною продукцією, з урахуванням поточних ринкових умов і потреб користувача.

Методи дослідження – у процесі дослідження застосовувалися методи аналізу предметної галузі, проєктування архітектури вебдодатку, реалізація серверної та клієнтської частини з використанням ASP.NET Core MVC, EF Core, а також МН для обробки зображень та прогнозування цін.

У результаті роботи був створений багатофункціональний вебдодаток, що дозволяє проводити оцінку стану монет, аналізувати динаміку цін та формувати рекомендації для користувачів.

Дана робота демонструє приклад ефективного поєднання засобів веброзробки з технологіями штучного інтелекту та відкриває перспективи для подальшого розширення функціональності – зокрема, інтеграції більшої кількості інформації чи створення мобільної версії.

ABSTRACT

Bachelor’s thesis contains: 88 pp., 39 fig., 3 tabl., 3 ann., 20 references.

ARTIFICIAL INTELLIGENCE, ASP.NET CORE MVC, CSS, ENTITY FRAMEWORK CORE, HTML, JAVASCRIPT, NUMISMATICS, POSTGRESQL, RAZOR PAGES, WEB APPLICATION.

The object of the study is a web application for analytical support of numismatics, its market and participants.

The subject of research is the development and implementation of an intelligent web application based on ASP.NET Core MVC and Entity Framework Core technologies for the analysis, evaluation and forecast of the value of coins.

The purpose of the work is to develop an intelligent information system that will provide a convenient interface and modern analytical tools for working with numismatic products, taking into account current market conditions and user needs.

Research methods – the research process used methods of domain analysis, web application architecture design, implementation of the server and client parts using ASP.NET Core MVC, EF Core, as well as ML for image processing and price forecasting.

As a result of the work, a multifunctional web application was created that allows you to assess the condition of coins, analyze price dynamics and generate recommendations for users.

This work demonstrates an example of an effective combination of web development tools with artificial intelligence technologies and opens up prospects for further expansion of functionality – in particular, integrating more information or creating a mobile version.

ЗМІСТ

Перелік умовних позначень, символів, одиниць, скорочень і термінів 9
Вступ 10
1 Аналіз предметної галузі 12
1.1 Опис предметної галузі та фактори її формування 12
1.1.1 Опис предметної галузі 12
1.1.2 Сучасний стан та особливості ринку нумізматики 13
1.1.3 Фактори формування цінності монети 15
1.2 Аналіз існуючих рішень 18
1.2.1 Інтернет-магазин НБУ 18
1.2.2 Violity 19
1.2.3 Hotline.ua 19
1.2.4 eBay 20
1.2.5 NGCcoin 21
1.3 Зіставлення рішень 21
1.4 Постановка задачі 23
2 Теоретичні відомості обраних технологій 24
2.1 Серверна логіка на основі ASP.NET Core 24
2.1.1 Фреймворк ASP.NET 24
2.1.2 Архітектурна модель MVC та переваги 25
2.2 Робота з базами даних через EF Core та PostgreSQL 27
2.3 Динамічний вебінтерфейс з Razor та Bootstrap 29
2.4 Збір даних із зовнішніх джерел 30
2.5 Інтелектуальні модулі штучного інтелекту 32
2.5.1 Класифікація стану монети за зображенням 32
2.5.2 Прогнозування ціни монети 33
2.5.3 Рекомендаційна система на основі взаємодій 34
3 Проєктування та розробка вебдодатку 36
3.1 Опис технічного завдання на розробку 36

3.2 Структура проєкту 37
3.3 Моделі та база даних 39
3.3.1 Таблиця Coins 41
3.3.2 Таблиці Countries, Currencies, Metals 42
3.3.3 Таблиці CoinPrices, PriceHistories 44
3.3.4 Таблиці UserProfiles, FavoriteCoins, ViewHistories 46
3.3.5 Таблиці Recommendations, ImageEvaluationResults, AuctionLots 48
3.4 Розробка контролерів 51
3.4.1 AccountController 52
3.4.2 AdminController 53
3.4.3 AnalyticsController 54
3.4.4 CoinsController 55
3.4.5 EvaluationController 57
3.4.6 HomeController 58
3.5 Застосування інструментів МН та ШІ 58
3.5.1 Прогнозування динаміки цін монети 59
3.5.2 Оцінювання стану монети 61
3.5.3 Рекомендаційна система 62
3.6 Допоміжні елементи 64
3.6.1 Сервіси 64
3.6.2 Утиліти 65
3.6.3 Парсери 66
3.7 Представлення 67
4 Дослідження розробленого вебдодатку 70
4.1 Огляд інтерфейсу додатка 70
4.1.1 Головна сторінка 70
4.1.2 Каталог 70
4.1.3 Сторінка монети 71
4.1.4 Прогнозування ціни 71

4.1.5 Оцінка стану 72
4.1.6 Адміністраційна панель 72
4.1.7 Автентифікація та авторизація 73
4.2 Загальний аналіз роботи вебдодатку 73
Висновки 75
Перелік джерел посилання 76
Додаток А Статистичні дані про навчання моделей машинного навчання 79
Додаток Б Представлення користувацького інтерфейсу 80
Додаток В Відомість кваліфікаційної роботи 88

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

МН – машинне навчання;

ШІ – штучний інтелект;

AI – Artificial Intelligence – штучний інтелект;

API – Application Programming Interface – інтерфейс програмування додатків;

ASP – Active Server Pages – служба обробки активних сторінок;

CSV – Comma-Separated Values – формат табличних даних, розділених комами;

CSS – Cascading Style Sheets – мова стилів;

DTO – Data Transfer Object – об’єкт для передачі даних;

HTML – Hypertext Markup Language – мова розмітки;

HTTP – Hypertext Transfer Protocol – протокол передачі даних;

JS – JavaScript – мова для створення інтерактивних та динамічних веб-сторінок;

JSON – JavaScript Object Notation – текстовий формат обміну даними;

LGBM – Light Gradient Boosting Machine – алгоритм градієнтного бустингу;

ML – Machine Learning – машинне навчання;

MLP – Multi-Layer Perceptron – багатошаровий перцептрон;

MVC – Model View Controller – модель вид контролер;

ONNX – Open Neural Network Exchange – формат представлення нейронних мереж;

ORM – Object-Relational Mapping – об’єктно-реляційне відображення;

.NET – платформа розробки програмного забезпечення.

ВСТУП

З розвитком технологій штучного інтелекту зростає потреба в автоматизації процесів, що раніше повністю базувалися на досвіді й суб'єктивній експертизі людини. Однією з таких сфер є нумізматика – галузь, що поєднує в собі історичну науку, колекціонування, інвестування та комерцію. З огляду на розгалуженість джерел, нестабільність ринку та складність уніфікованої оцінки стану чи вартості монет, перед розробниками постає необхідність створення інструментів, що б поєднували експертність, динамічну аналітику та технологічну масштабованість.

Актуальність теми зумовлена тим, що більшість існуючих рішень, що дотична до нумізматики – від офіційних сайтів банків до аукціонних платформ – зосереджені лише на окремих аспектах: продажі, перегляді чи сертифікації. Жодна з них не пропонує комплексного, аналітично-орієнтованого підходу до роботи з монетами. Ситуацію ускладнює ще й те, що оцінка стану та вартості часто здійснюється вручну або ґрунтується на недостатньо формалізованих критеріях.

Метою кваліфікаційної роботи стало проєктування та реалізація інтелектуального вебдодатку, що виконує роль системи експертного аналізу монети як основного об'єкту нумізматики. Основною концепцією є створення аналітичного каталогу, який агрегує дані з різних джерел, аналізує цінові тенденції, робить пропозиції щодо майданчиків, де можна купити, оцінює стан монет на основі зображень, та надає персоналізовані рекомендації, що дозволяє створити єдине середовище, що об'єднує історичну глибину предметної галузі з інструментами штучного інтелекту.

Технологічна основа системи базується на фреймворку ASP.NET Core, використанні шаблону MVC, ORM-бібліотеці EF Core у поєднанні з PostgreSQL, а також інтеграції моделей машинного навчання, реалізованих засобами Python (для зображень та прогнозу цін) та ML.NET (для

рекомендацій). Ключову роль в системі відіграють три модулі: класифікатор стану монети на основі зображення, регресійна модель прогнозу ціни з урахуванням динаміки та фізичних характеристик монети, а також рекомендаційна система, що аналізує історію переглядів користувача.

Окрему увагу в рамках практики приділено теоретичному обґрунтуванню обраних технологій, аналізу предметної галузі, виявленню обмежень сучасних платформ та формуванню логіки функціонування кожного модуля системи. Запропоноване рішення дозволяє поєднати інтелектуальні обчислення з високою продуктивністю, будучи цінним як кінцевому користувачу, так і слугувати за основу для подальшої науково-прикладної розробки.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Опис предметної галузі та фактори її формування

1.1.1 Опис предметної галузі

Нумізматика – це історична дисципліна, що за мету ставить аналіз та вивчення таких об’єктів, як монети, медалі, банкноти, жетони, знаки та інші предмети грошової діяльності, що в свою чергу є джерелами історичних, політичних, економічних та культурних знань. В цю сферу також входить історія грошового обігу, процес виготовлення грошових знаків, їх фальсифікація та боротьба з цим явищем у різні історичні проміжки. На основі цієї інформації, історики мають змогу досліджувати розвиток суспільств та цивілізацій в економічних, культурних, політичних та інших аспектах враховуючи вплив тої чи іншої фінансової системи, що і робить нумізматику важливою складовою історії як науки.

Нумізматика бере свій початок з появою перших приблизно у 7 столітті до н.е. у Китаї, Індії та Лідії, землі якої пролягали на території сучасної Туреччини. Монети були виготовленні із бронзи, срібла чи золота. Саме монети з Лідії, які поширились згодом на інші грецькі поліси, мали дископодібну форму та були проштамповані з обох сторін, що відповідає сучасному вигляду монети. Вже з розповсюдженням по всьому світу поступово з’являлась необхідність вивчення та класифікації їх різних видів та типів, що послугувало народженню нумізматики для вивчення монет та елементів грошового обігу [1].

Середньовіччя стало знаковим періодом з точки зору зросту популярності нумізматики, коли монети стали не лише засобом обігу, але й способом зберігання та транспортування багатства. В цей же час з’явилися перші нумізмати, які почали колекціонувати монети різних часів та країн. У XVIII–XIX століттях нумізматика стала повноцінною наукою, яка вивчає не

тільки самі монети, але й їх історію, значення та роль у різних економічних та соціальних системах.

Нумізматика нині на ряду з іншими є однією з головних галузей археології та історії. Вона дозволяє вивчати історію різних країн та культур, а також зберігати та досліджувати цінні грошові пам’ятки. Нумізмати продовжують збирати монети, створюючи великі та цінні колекції, які допомагають зберегти та вивчати культурну спадщину різних епох.

В Україні нумізматика має свої особливості розвитку. Перші обігові монети були викарбувані в 1992–1993 роках, коли на Луганському верстатобудівному заводі та Римському монетному дворі було виготовлено пробні екземпляри копійок із різних металів. Ці монети стали основою для розвитку сучасної української нумізматики. З 1998 року виробництво монет перемістилося до Києва, на Банкнотно-монетний двір Національного банку України, який і карбує національну валюту до сьогодні. На сьогодні він карбує як обігові монети, так і колекційні із недорогоцінних сплавів, а також срібла та золота.

1.1.2 Сучасний стан та особливості ринку нумізматики

Ринок нумізматики в Україні перебуває на етапі активного розширення, поєднуючи традиційні підходи до колекціонування з цифровою торгівлею та інвестиційною привабливістю. Його учасниками виступають як приватні особи, що мають особисту зацікавленість у колекціонуванні монет, так і професійні нумізмати, дилери, аукціоністи та інвестори, які розглядають колекціонування монет як фінансовий інструмент.

Колекціонери, будучи найвагомішою частиною ринку, зазвичай керуються історичними, культурними або естетичними міркуваннями при виборі екземплярів. Вони збирають як обігові монети з дефектами чи браком у штампуванні, так і пам’ятні серії, які присвячені різній

тематиці (культурні пам’ятки, визначні події та постаті в історії тої чи іншої країни, тощо). Дилери, своєю чергою, оперують великими обсягами монет, займаючись їх купівлею, перепродажем, оцінкою та сертифікацією. Інвестори зазвичай зацікавлені в монетах з дорогоцінних металів, зокрема інвестиційними виробами із золота та срібла, які мають гарантовану пробу та вагу.

У структурі ринку за типами нумізматичної продукції виокремлюються три основні категорії монет: обігові, пам’ятні та інвестиційні. Обігові монети, хоч і не є основним предметом інтересу, все ж мають свою аудиторію серед колекціонерів-початківців, а також нумізматів, що збирають екземпляри монет із фабричними дефектами. Пам’ятні монети, які випускаються Банкнотно-монетним двором НБУ чи іншими подібними установами в різних країнах, відзначаються художнім оформленням і тематикою, яка відображає знакові події національного значення. Їх колекційна вартість зростає завдяки обмеженим тиражам та високій якості карбування. Інвестиційні монети, зазвичай виготовлені з дорогоцінних металів, мають фіксовану вартість, прив’язану до ринку золота чи срібла, і використовуються як спосіб зберігання капіталу [2].

Ринок в Україні істотно змінився після запровадження цифрових каналів збуту. Віднедавна НБУ запустив повноцінний офіційний інтернет-магазин, що дозволяє фізичним особам напряму купувати нумізматичну продукцію, яка щойно вводиться в обіг та продаж, що суттєво спростило доступ до нумізматичної продукції, зробивши процес прозорим, доступним і контрольованим. Окрім цього, популярності набувають незалежні платформи, як-от Violity, які виступають у ролі онлайн-аукціонів для нумізматичних товарів. Така комерціалізація через електронні канали є важливим чинником у формуванні відкритого й конкурентного середовища.

Глобалізація істотно вплинула на розвиток нумізматичного ринку в Україні, сприяючи активному залученню українських колекціонерів до міжнародного середовища. Сьогодні учасники внутрішнього ринку мають

змогу безперешкодно брати участь в аукціонах за кордоном, придбавати рідкісні монети з усього світу, а також реалізовувати власні колекції на глобальних онлайн-платформах. Таке середовище створює передумови для більшої конкуренції, стандартизації вимог до оцінки автентичності та стану монет, а також формує нові підходи до ціноутворення.

Водночас глобалізація дозволяє українським продавцям пропонувати свою продукцію міжнародним покупцям, підвищуючи інвестиційну привабливість українських монет на світовому рівні. Поширення загальноприйнятих систем сертифікації, таких як грейдинг від NGC чи PCGS, пришвидшує інтеграцію в нумізматичну спільноту на міжнародному рівні та знижує бар'єри для взаємодії між ринками.

Серед нових трендів варто також відзначити зростання інтересу до цифрових нумізматичних активів – токенізованих монет або колекцій у форматі NFT. Попри те, що сегмент ще перебуває на ранніх стадіях розвитку, він демонструє потенціал для об'єднання технологій блокчейну та класичного колекціонування. Такі інновації дозволяють фіксувати права власності та походження об'єктів, що має неабияке значення в умовах поширення підробок і втрати автентичності.

1.1.3 Фактори формування цінності монети

Формування вартості монет є складним і багатофакторним процесом, що залежить від кількох взаємопов'язаних критеріїв. Одним із ключових чинників є рідкість монети, яка визначається тиражем її випуску, металу, з якого вона вироблена та її станом. Для визначення її ступеня збереженості використовують спеціальні шкали, наприклад шкалу Шелдона, яка має 70 градацій – від найгіршого стану (Poor, P-1) до ідеального (Mint State, MS-70) [2]. Детальний опис мірила відображений у таблиці 1.1.

Таблиця 1.1 – Оцінка стану монети

Міжнародна система	Шкала Шелдона	Пояснення
PF (Proof)	PF 1-70	Полірований
PL (Proof-like)	MS 60-70 PL	Чудовий із дзеркальним блиском
BU (Brilliant Uncirculated)	MS 65-70	Чудовий
UNC (Uncirculated)	MS 60-64	
AU+ (Choice Almost Uncirculated)	AU 55, 58	Майже чудовий
AU (Almost Uncirculated)	AU 50, 53	
XF+ (Choice Extremely Fine)	XF 45	Відмінний
XF (Extremely Fine)	XF 40	
VF+ (Choice Very Fine)	VF 30, 35	Майже відмінний
VF (Very Fine)	VF 20, 25	Дуже добрий
F (Fine)	F 12, 15	Добрий
VG (Very Good)	VG 8, 10	Задовільний
G (Good)	G 4, 6	Незадовільний
AG (Almost Good)	AG 3	
FA (Fair)	FA 2	
PR (Poor)	PR 1	

Наступним чинником, що значно грає у формування вартості є попит, який суттєво варіюється залежно від трендів, популярності певних серій або суспільно-політичних подій. Його високий рівень може значно підвищити ринкову ціну монет, особливо якщо йдеться про серії, пов’язані з важливими історичними подіями або видатними особистостями.

Проте, оцінка вартості монет нерідко супроводжується суб’єктивними чинниками, які створюють певні складнощі на ринку. Відсутність чітких

стандартів чи недостатня кваліфікація оцінювачів можуть призводити до суттєвих розбіжностей у цінах. Різні спеціалісти можуть по-різному оцінити одну й ту саму монету через особисті уподобання або індивідуальний досвід, що зумовлює важливість застосування професійних методів сертифікації, де монета упаковується у спеціальний герметичний контейнер – слаб (його приклад наведений на рисунку 1.1), із зазначенням стану та автентичності, що зменшує ризик суб'єктивних оцінок.

Рисунок 1.1 – Приклад слабу NGC з оцінкою MS 62 PL

Для більш об'єктивного підходу на допомогу колекціонерам приходять спеціалізовані каталоги монет. Вони містять детальну інформацію щодо тиражу, року випуску, художніх та технічних характеристик монет, а також вказують орієнтовні ринкові ціни. Використання каталогів допомагає уніфікувати підхід до визначення вартості, проте вони не завжди можуть швидко реагувати на зміни ринкових умов.

Також останнім часом на допомогу експертам приходять сучасні технологічні інструменти, зокрема макрозйомка та цифровий аналіз поверхні монет, які дають змогу виявляти навіть незначні дефекти чи

втручання, що значно підвищують точність і прозорість процесу оцінки. Важливе значення мають також професійні форуми та онлайн-спільноти, де експерти можуть обговорювати та уточнювати оцінку монет, враховуючи колективний досвід і поточні ринкові тренди. Втім, остаточну вартість будь-якої монети завжди визначає ринок – ціна, яку реальний покупець готовий за неї заплатити.

1.2 Аналіз існуючих рішень

На сьогодні нумізматичний ринок активно представлений різними інструментами, онлайн-сервісами та платформами, які забезпечують можливість купівлі, продажу та оцінки монет. Попри це, більшість із них мають суттєві обмеження, пов’язані з недостатньою автоматизацією процесів ціноутворення, суб’єктивністю оцінки вартості чи стану, а також відсутністю ефективних інструментів прогнозування ринкових тенденцій.

Існує кілька основних онлайн-платформ для продажу чи аналізу стану нумізматичної продукції, серед яких виокремимо найосновніші.

1.2.1 Інтернет-магазин НБУ

Офіційний інтернет-магазин Національного банку України пропонує широкий асортимент пам’ятних та інвестиційних монет, випущених банком, що гарантує їхню автентичність та високу якість. Це зручний і надійний канал для покупки монет з нейзильберу, срібла чи золота, що приваблює колекціонерів та інвесторів через те, що в даному випадку магазин виступає першоджерелом реалізації нової продукції [3]. Однак в цьому і криється обмеження платформи, так як продукції старшого року випуску знайти буде майже неможливо.

Магазин підтримує динамічне ціноутворення в питанні щодо монет з дорогоцінних металів, так як вони фізично прив’язані до ринкової вартості

срібла та золота. Щодо недорогоцінних металів та формування ціни за рівнем попиту, то це відсутнє в інтернет-магазині. Також відсутність інтелектуальних інструментів для прогнозування ціни та рекомендацій обмежує можливості для інвесторів, які потребують більш гнучкого підходу до покупок.

1.2.2 Violity

Violity – найбільший онлайн-аукціон для колекціонерів в Україні, де можна купувати та продавати монети, медалі, банкноти та інші предмети колекціонування [4]. Платформа функціонує за принципом аукціону, де покупці роблять ставки на товари, і в кінцевому підсумку ціна визначається через процес торгів. Також існує опціонально механізм прямої покупки. Організація платформи дозволяє продавцям та покупцям знаходити найбільш вигідні пропозиції в рамках конкуренції, забезпечуючи певний рівень динаміки на ринку.

Проте, хоча Violity і є популярною платформою, вона має кілька недоліків. Так як ціни на монети формуються безпосередньо через ставки користувачів, це може призводити до значних коливань вартості, яка залежить від активності учасників аукціону, а не від реального ринкового попиту чи рідкості монети, що певною мірою може створити цінову бульбашку. Крім того, відсутність інтелектуального аналізу чи алгоритмів прогнозування цін обмежує можливості для покупців, які шукають об’єктивну оцінку монет.

1.2.3 Hotline.ua

Hotline.ua – це український агрегатор товарів, який виконує функцію каталогу з можливістю порівняння цін із різних інтернет-магазинів. Користувачі мають змогу переглядати пропозиції за різними

характеристиками, фільтрувати результати за ціною, рейтингом, брендом, технічними параметрами тощо [5]. Платформа не здійснює прямий продаж, а перенаправляє на сайти магазинів, виступаючи посередником у процесі вибору. Hotline охоплює широкий спектр категорій товарів – від побутової техніки до косметики й одягу, з акцентом на прозорість і зручність пошуку.

Попри продуману логіку побудови каталогу та зручну навігацію, сервіс не передбачає окремого розділу, присвяченого нумізматиці. Відсутні категорії, фільтри чи параметри, релевантні для монет, що робить платформу непридатною для вузької цільової аудиторії нумізматів, які потребують порівняння або аналітики цін на монети.

1.2.4 eBay

Платформа eBay є одним із наймасштабніших світових онлайн-платформ з комерції, що активно використовується також в нумізматичних колах [6]. Завдяки глобальному охопленню, користувачі мають змогу знайти монети практично будь-якої країни, періоду чи типу – від обігових до унікальних старовинних екземплярів, що відкриває великі можливості для колекціонерів та продавців з усього світу, а також дозволяє вивчати ринок у міжнародному контексті. Популярність платформи забезпечує високу конкуренцію серед продавців, що в свою чергу дає змогу знайти товари за вигідною ціною.

Разом з тим, eBay не є спеціалізованим ресурсом для нумізматів. Вартість товарів визначається або ставками учасників, або встановлюється продавцем вручну. Відсутність вбудованих інструментів оцінки, каталогів чи фільтрів за критеріями, важливими для колекціонерів (наприклад, грейдинг, рідкість, сертифікація), ускладнює орієнтацію на ринку. Крім того, ризики, пов’язані з автентичністю монет і можливістю придбати підробку, залишаються актуальними, попри репутаційну систему і підтримку покупців.

1.2.5 NGCcoin

NGCcoin.com – це спеціалізована міжнародна платформа, що належить компанії Numismatic Guaranty Company (NGC), яка займається професійною сертифікацією монет [7]. Основна функція цього сервісу полягає в оцінці автентичності та фізичного стану монет відповідно до загальноприйнятих міжнародних стандартів. Після оцінювання монета упаковується у захисний контейнер (слаб), на якому зазначено її грейд за шкалою Шелдона. Такі сертифіковані монети мають вищу довіру серед покупців і активно котируються на вторинному ринку.

Сервіс NGC має важливе значення для розвитку стандартизованого підходу в нумізматиці, однак він не є торговим майданчиком у класичному сенсі. Платформа не надає функціоналу для купівлі-продажу монет, не аналізує ринкову динаміку та не формує ціни. Вона зосереджена виключно на оцінці, архівації та реєстрації нумізматичних об’єктів. В свою чергу це робить її надзвичайно важливою в частині підтвердження якості монет, але обмежує її практичну цінність для колекціонерів, які шукають інтегровані рішення – зокрема ті, що поєднують оцінку, аналітику, динамічне ціноутворення та електронну комерцію.

1.3 Зіставлення рішень

Аналіз наявних онлайн-сервісів, проведений у попередньому підрозділі, дозволив окреслити сильні сторони кожної платформи та водночас виявити низку функціональних обмежень. Усі рішення, що розглядаються, зосереджені лише на окремих аспектах роботи з нумізматичною продукцією – продаж, аукціон, сертифікація або порівняння цін – проте жодне з них не поєднує ключові можливості в єдиній системі аналітики.

Переважна більшість платформ працює у форматі торгових майданчиків або аукціонів. Інтернет-магазин НБУ реалізує лише нову продукцію і не охоплює вторинний ринок. Violity та eBay дозволяють навпаки, торгувати монетами на вторинному ринку, проте цінова динаміка формується виключно ставками або продавці самі формують ціну товару, без урахування ринкових індикаторів чи трендів, що може призвести до невдалої спроби продажу та витраченого часу. NGCcoin взагалі не передбачає взаємодії з ринком, зосереджуючись на оцінці та сертифікації.

Жодна з платформ не пропонує повноцінного механізму динамічного ціноутворення, який враховував би рідкість монет, стан, поведінку покупців та вартість металу в реальному часі. Ціни, як правило, задаються вручну або залежать від активності аукціонних ставок, що ускладнює об’єктивний аналіз і прогнозування. Жодна з розглянутих платформ не використовує автоматизовані інструменти оцінки вартості монет. NGC здійснює сертифікацію вручну, через фахівців, без застосування алгоритмів або цифрової обробки, що виключає можливість масштабування та інтеграції в аналітичні системи.

Із розглянутих платформ базову систему рекомендацій реалізовано лише на eBay, де користувачу пропонуються схожі товари залежно від історії переглядів і пошукових запитів. Водночас ці алгоритми не враховують специфіку нумізматичного ринку та не орієнтуються на параметри, релевантні для монети як сутності.

Наявні платформи виконують лише часткові функції: одні займаються продажем нових монет, інші – торгами між користувачами, а деякі – професійною сертифікацією. Проте жодна з них не об’єднує в єдиному середовищі можливості оцінки монети, визначення її ринкової вартості, надання персоналізованих рекомендацій та зручного механізму агрегації. Відсутність інструментів штучного інтелекту чи базової автоматизації створює потребу в новому типі системи – інтелектуальному вебдодатку-

каталогу, який забезпечить комплексний підхід, закриваючи наявні функціональні потреби.

1.4 Постановка задачі

Метою кваліфікаційної роботи є створення інтелектуального вебкаталогу нумізматичної продукції, що надає користувачам структурований доступ до актуальної інформації про монети, аналітику цінових тенденцій, персоналізовані рекомендації та інструменти для оцінки вартості.

Каталог може використовуватись як колекціонерами й покупцями для моніторингу ринку, так і продавцями – для попереднього аналізу ринку, оцінки доцільності продажу тієї чи іншої монети, а також орієнтації щодо ціноутворення. Функціонально платформа повинна включати:

– каталог монет з агрегованими пропозиціями (з різних джерел), нормалізованими назвами, описами, фото та зазначеними цінами;

– модель аналізу динаміки вартості монет на основі історичних даних для виявлення трендів, сезонності та стабільності на ринку;

– модель оцінки монети за зображенням, що поєднує візуальну частину та текстову інформацію про екземпляр з нумізматичної літератури та каталогів для прогнозування ринкової вартості;

– рекомендаційну систему, що базується на історії переглядів користувача з урахуванням популярності, схожості, тематичного та цінового сегмента;

– можливість фільтрації, сортування та підбору монет за ключовими параметрами: рік, тип, метал, стан, країна, орієнтовна вартість;

– адаптивний інтерфейс з вебреалізацією, серверною частиною на C# і базою даних для збереження інформації про монети, ціни, динаміку та користувацькі дії.

2 ТЕОРЕТИЧНІ ВІДОМОСТІ ОБРАНИХ ТЕХНОЛОГІЙ

2.1 Серверна логіка на основі ASP.NET Core

2.1.1 Фреймворк ASP.NET

ASP.NET Core – це відкритий фреймворк, розроблений Microsoft, слугує для творення динамічних вебдодатків та вебсервісів. Інструмент є частиною платформи .NET, відзначається кросплатформністю, високою продуктивністю, і дозволяє запускати програми на Windows, Linux і macOS [8].

Головні плюси ASP.NET Core: висока продуктивність, модульність, впровадження технології Dependency Injection (інжекції залежностей), вбудована підтримка асинхронних операцій, а також гнучка конфігурація обробки HTTP-запитів за допомогою Middleware-компонентів [2]. В рейтингах продуктивності вебфреймворків ASP.NET Core незмінно в лідерах, що пояснюється оптимізаціями, внесеними в новіші версії фреймворку.

У вебфреймворку ASP.NET Core широко застосовується шаблон проєктування MVC (Model–View–Controller), забезпечуючи чіткий розподіл обов’язків між складовими системи: моделями, контролерами та представленнями. Підхід в свою чергу допомагає структурувати код, полегшує супровід, масштабування та тестування програми.

ASP.NET Core також гарантує зручну інтеграцію зі сторонніми бібліотеками та сервісами завдяки вбудованому механізму інжекції залежностей, що дозволяє зменшити взаємозалежність коду, спростити його підтримку та розширити функціонал у майбутньому.

Беручи до уваги вищезгадані характеристики, ASP.NET Core було обрано як основу для серверної логіки в проєкті для забезпечення стабільної

та швидкої роботи системи, а також ефективної взаємодії з базами даних, машинним навчанням та іншими функціональними модулями.

2.1.2 Архітектурна модель MVC та переваги

Архітектурна модель MVC (Model-View-Controller) є одним з найпопулярніших шаблонів проєктування, які використовуються для створення структурованих вебдодатків. Її основна ідея полягає у чіткому розділенні відповідальності між трьома компонентами: моделлю, представленням та контролером (рисунок 2.1) [9].

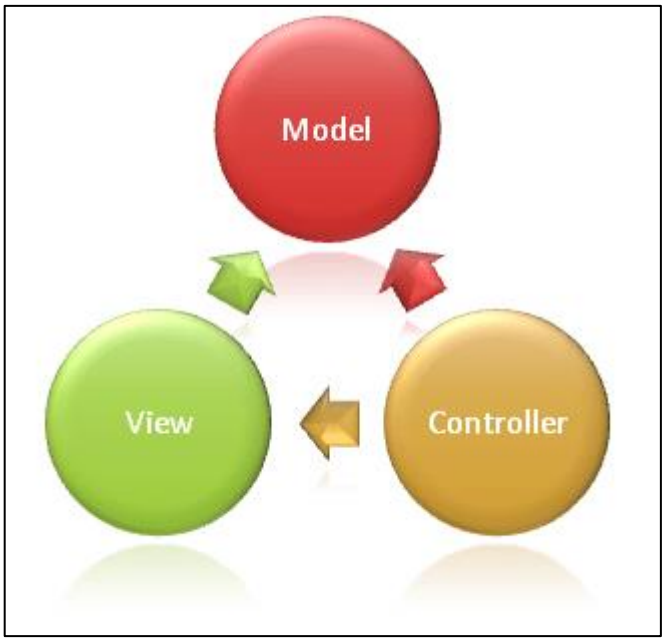

Рисунок 2.1 – Схема моделі MVC

Компонент моделі (Model) відповідає за дані, бізнес-логіку та правила роботи з ними. Саме в цьому компоненті відбувається взаємодія з базою даних, обчислення та логіка предметної області. Особливістю такого підходу є незалежність моделі від візуального оформлення чи взаємодії з користувачем, надаючи змогу зберігати логіку додатку без змін навіть при значних перетвореннях способу відображення даних чи структури

інтерфейсу [9]. Завдяки цьому модель можна тестувати окремо від інших компонентів, що підвищує якість програмного продукту та знижує ризик помилок під час розробки.

Представлення (View) – це інтерфейс користувача, що візуалізує інформацію, отриману від моделі через контролер. Представлення не повинне містити складної обчислювальної логіки, його завдання – лише форматувати та показувати дані користувачеві. Відсутність складної логіки в представленні дає змогу розробникам зосередитися на дизайні та зручності користування. Такий підхід суттєво полегшує створення та супровід вебінтерфейсів, дозволяючи змінювати зовнішній вигляд програми без впливу на інші її складові.

Контролер (Controller) виконує роль посередника між моделлю і представленням. Його основні функції – приймати HTTP-запити від користувачів, їх обробляти, передавати інформацію до моделі для обчислень та отримувати результати від моделі. Контролер також визначає відповідне представлення для відображення інформації користувачеві. Завдяки контролеру досягається централізоване керування логікою додатку, оскільки саме в ньому описується маршрутизація запитів та способи реагування на дії користувачів.

Переваги використання архітектури MVC включають чітку структуру коду та відокремлення різних аспектів функціонування вебдодатку, що дозволяє створити прозорий та зрозумілий програмний продукт, який легше супроводжувати та розширювати. Оскільки компоненти (модель, представлення, контролер) розділені, командна розробка спрощується: різні фахівці можуть працювати над окремими частинами незалежно один від одного, що прискорює створення якісного програмного продукту.

Ще однією значною перевагою MVC є простота тестування: кожен з компонентів (зокрема модель і контролер) можна перевіряти окремо, створюючи можливість проводити модульне тестування, яке значно збільшує надійність додатку та зменшує ризик помилок на пізніх етапах

розробки. Завдяки низькій зв’язаності компонентів, внесення змін в один з них зазвичай не потребує значної перебудови інших частин додатку, що робить додаток гнучким і легко масштабованим.

Також модель MVC широко підтримується сучасними вебфреймворками, зокрема ASP.NET Core, що й було обрано для реалізації серверної частини системи. Вбудовані засоби ASP.NET Core MVC, як-от маршрутизація, інтеграція залежностей (Dependency Injection) і підтримка механізмів Middleware, дозволяють швидко та ефективно створювати стабільні вебдодатки з чіткою структурою. Використання такого підходу значно скорочує час на розробку та покращує якість фінального продукту завдяки наявності багатьох перевірених та стандартизованих рішень.

2.2 Робота з базами даних через EF Core та PostgreSQL

Збереження та опрацювання даних є одним із центральних завдань будь-якого інформаційного вебдодатку. У проєкті для реалізації цієї складової обрана комбінація реляційної системи управління базами даних PostgreSQL та об’єктно-реляційного відображення Entity Framework Core (EF Core). Такий вибір зумовлений потребою у надійному, масштабованому і продуктивному механізмі зберігання даних у поєднанні з гнучким та зручним інструментом для доступу до них на рівні бізнес-логіки.

PostgreSQL – система управління базами даних з відкритим вихідним кодом, яка підтримує стандарт SQL і визнана однією з найкращих безкоштовних СУБД на ринку. PostgreSQL забезпечує повну транзакційність, підтримує складні типи даних, реляційні зв’язки, вкладені запити, агрегацію, розширювану систему індексації, а також можливість зберігання JSON-структур. Її стабільність і масштабованість роблять її оптимальним вибором для середовищ, де критичним є збереження цілісності даних і висока швидкість обробки запитів [10].

Вибір PostgreSQL для проєкту зумовлений не лише її функціональними можливостями, а й активною підтримкою в екосистемі ASP.NET Core. Завдяки бібліотекам (наприклад, Npgsql) забезпечується повна сумісність з Entity Framework Core, що дозволяє ефективно будувати реляційні моделі даних та зручно взаємодіяти з ними з рівня програмного коду.

Entity Framework Core – ORM-бібліотека від Microsoft, що полегшує та декларативно керує доступом до даних. З EF Core розробники оперують таблицями БД як наборами об’єктів, застосовуючи LINQ (Language Integrated Query) для формування запитів. Інструмент дозволяє абстрагуватися від SQL-синтаксису, підвищуючи читабельність та спрощуючи підтримку коду. Система підтримує два головні підходи: Code First, де модель формується на основі класів C# з наступним створенням структури БД, та Database First, коли модель генерується з вже існуючої бази [11].

У проєкті обрано Code First, оскільки це забезпечує максимальний контроль над структурою даних на всіх стадіях розробки. Зміни у схемі фіксуються за допомогою механізму міграцій, що дає змогу автоматично оновлювати структуру БД, зберігаючи поточні дані. Особливо це актуально при розширенні функціоналу або оновленні моделей в процесі розробки.

Перевагою EF Core є також підтримка зв’язків: «один до одного», «один до багатьох» і «багато до багатьох», що дозволяє ефективно реалізувати складні структури таблиць. В рамках проєкту реалізовано нормалізовану схему БД згідно з третьою нормальною формою (3НФ), що передбачає поділ сутностей на окремі таблиці з зовнішніми ключами, усуваючи дублювання інформації та покращує цілісність даних.

Однак, слід зауважити, що EF Core не завжди є найкращим вибором для задач, де ключовою вимогою є висока швидкодія при виконанні запитів, що потребують складних багатотабличних з’єднань, агрегацій або специфічних оптимізацій. У подібних сценаріях більш доцільним може бути

використання нативних SQL-запитів, через метод FromSqlRaw, або безпосереднє застосування представлень і збережених процедур безпосередньо в PostgreSQL. У розроблюваному додатку саме такий комбінований підхід передбачено для реалізації запитів, що стосуються обробки історичних даних про зміни цін монет, аналізу трендів та передбачення значень.

Використання Entity Framework Core у поєднанні з PostgreSQL дає можливість реалізувати надійну та продуктивну серверну частину, що дозволяє ефективно зберігати, обробляти та візуалізувати складні структуровані дані. Застосування такого поєднання характеризується високою масштабованістю, легкою інтеграцією з іншими компонентами системи і забезпечує підтримку високої якості даних при збереженні зручності розробки.

2.3 Динамічний вебінтерфейс з Razor та Bootstrap

Для реалізації сучасного, динамічного інтерфейсу вебдодатку, є доцільним використання серверних шаблонів Razor з фреймворком стилів Bootstrap, дозволяючи своїм функціоналом забезпечити як динамічну побудову наповнення сторінок на основі даних, так і адаптивність інтерфейсу до різноманітних розмірів екранів.

Razor – це синтаксис для розробки HTML-шаблонів у рамках ASP.NET Core. Він дає змогу поєднувати HTML-розмітку з C#-логікою в одному файлі (.cshtml), що значно спрощує генерацію динамічного наповнення на боці сервера. Razor дає можливість працювати з переданими моделями, будувати умовні блоки, цикли та опрацьовувати дані безпосередньо в шаблоні. Механізм зручний для зведення сторінок, які залежать від статусу користувача, наявних даних або результатів фільтрування [12].

Крім генерації HTML, Razor взаємодіє з механізмами представлення MVC чи Razor Pages, надаючи чітку структуру між логікою контролерів, моделями й інтерфейсом користувача. Завдяки цьому скорочується об’єм дублювання коду, а сам інтерфейс стає простіше підтримувати.

Для візуального оформлення сторінок та створення гнучкої адаптивної сітки зручно використовувати Bootstrap – популярний CSS-фреймворк з відкритим кодом. Bootstrap надає низку стилізованих компонентів: кнопки, форми, таблиці, картки, сповіщення, панелі навігації тощо. Також у ньому реалізована система класів для побудови адаптивного макету, що базується на flexbox-моделі й дозволяє створювати сторінки, які правильно відображаються на різних пристроях – від телефонів до широкоформатних екранів [13].

2.4 Збір даних із зовнішніх джерел

У системах, які взаємодіють з реальними об’єктами або ринковими показниками, ключовим етапом є отримання свіжої інформації з зовнішніх ресурсів. У контексті розробки вебдодатку, розумно передбачити автоматизований процес збирання даних про монети та пов’язаних з ними інформацію, що в свою чергу забезпечує оновлення інформаційної бази системи, підвищуючи її практичність та достовірність для кінцевого користувача.

Як інструмент для реалізації цієї задачі, часто використовують вебскрапінг – процес автоматичного вилучення структурованих даних зі сторінок вебсайтів. З технічної точки зору, вебскрапінг можна здійснити за допомогою мов програмування, які підтримують бібліотеки для HTTP-запитів та аналізу HTML-документів. Однією з найпопулярніших комбінацій є використання мови Python разом з бібліотеками requests та BeautifulSoup.

Бібліотека requests дає можливість надсилати HTTP-запити та отримувати вміст вебсторінок у форматі HTML. Водночас BeautifulSoup надає інструменти для синтаксичного аналізу отриманого HTML-документа, пошуку елементів за тегами, класами, атрибутами тощо. Такий підхід ефективний, якщо структура сторінки не змінюється та не вимагає виконання JavaScript-коду.

У випадку збору даних для нашого проєкту, потенційними джерелами можуть бути відкриті каталоги монет, аукціонні платформи, офіційні вебсайти центральних банків, інформаційні портали та API-сервіси. Зокрема, для створення навчальних вибірок та оновлення баз даних можуть використовуватися такі ресурси, як Numista, ua-coins.info, Violity тощо. Вибір джерел залежить від їх доступності, направленості, частоти оновлення, повноти та структурованості інформації.

Крім вилучення основних характеристик монет, важливим елементом є збір зображень. Їх наявність в системі дозволяє реалізовувати функції комп’ютерного зору, а також забезпечує візуальну ідентифікацію об’єкта користувачем. Тому парсинг повинен включати збереження як текстової, так і графічної інформації – з відповідним прив’язуванням до ID монети або іншого унікального ідентифікатора.

Зібрані дані, як правило, зберігаються у форматах CSV, JSON або безпосередньо імпортуються в базу даних. Далі їх можна використовувати для навчання моделей машинного навчання, формування аналітичних звітів, оновлення користувацького інтерфейсу або створення системи рекомендацій.

Для забезпечення стабільності парсера критично важливо враховувати ймовірні зміни на сайтах-джерелах. Отже, рекомендується реалізувати систему логування помилок та адаптивний підхід до вибору HTML-елементів. У випадку наявності API, доцільно віддати перевагу йому, як більш надійному джерелу інформації.

2.5 Інтелектуальні модулі штучного інтелекту

2.5.1 Класифікація стану монети за зображенням

Одним із прикладних завдань, важливих для нашої системи, є автоматизована оцінка стану монети на основі її зображення. Завдання належить до області комп’ютерного зору (computer vision), зокрема, класифікації зображень, і його можна розв’язати за допомогою методів машинного навчання, а саме, згорткових нейронних мереж (CNN).

Під станом монети звичайно розуміють її грейд – умовну класифікацію ступеня збереження об’єкта. Традиційно, класифікація виконується експертами вручну, що є суб’єктивним, повільним і не завжди репрезентативним процесом. Використання глибокого навчання дає змогу автоматизувати його, знизити похибки та забезпечити масштабованість при роботі з великими обсягами монет.

Для вирішення поставленої задачі рекомендується застосування згорткових нейронних мереж, зокрема, архітектур, оптимізованих для мобільних та швидких додатків, таких як MobileNetV3. Модель відрізняється невеликим об’ємом параметрів, високою продуктивністю та конкурентною точністю при класифікації зображень на обмежених ресурсах. Мережа складається з згорткових шарів, bottleneck-блоків та механізму attention (Squeeze-and-Excitation), що дозволяє адаптувати її до візуальних завдань середньої складності [14].

Перед тренуванням такої моделі, зазвичай, виконується попередня підготовка даних: зображення нормалізуються та аугментуються. Аугментація – це процес, що направлений на підвищення стійкості моделі до змін умов зйомки та покращує узагальнюючу здатність (уникаючи такі проблеми, як випадкове обертання, зміна яскравості, контрасту, дзеркальне відображення).

В процесі навчання модель отримує на вхід зображення та класифікує його у один із попередньо визначених класів збереження. Для створення якісної моделі потрібна розмічена вибірка, де кожне зображення має відповідну мітку грейду, встановлену експертом. У перспективі, така система може не тільки класифікувати, але й обчислювати ймовірності належності до кожного класу, що враховує невизначеності в прогнозах.

В якості формату експорту для подальшої інтеграції в серверну частину вебдодатку доцільно застосовувати ONNX (Open Neural Network Exchange) – відкритий стандарт обміну моделями, сумісний з .NET через ONNX Runtime та дозволяє запускати модель безпосередньо з C# без необхідності окремого Python-сервера, що значно спрощує архітектуру системи та скорочує затримки [15].

2.5.2 Прогнозування ціни монети

Коливання вартості монет обумовлені багатьма чинниками: унікальністю, з чого зроблено, ринковим попитом, а також історією продажів. У інформаційній системі, яка оперує з такими об’єктами, виглядає виправданим інтегрування модуля прогнозування ціни. Завдяки цьому створюється можливість не тільки показувати актуальні ціни, а й інформувати користувача про ймовірну динаміку змін вартості в майбутньому.

З технічної точки зору прогноз ціни може бути розглянутий як завдання регресії – передбачення числового значення на основі структурованих даних. Одним з найбільш ефективних інструментів для цих цілей сьогодні є алгоритм градієнтного бустингу, зокрема LightGBM. Фреймворк поєднує високу точність з хорошою масштабованістю, що дає змогу обробляти тисячі прикладів з великою кількістю ознак за короткий проміжок часу [16].

Модель прогнозу базується не тільки на останньому значенні ціни, але й на послідовності попередніх значень. Такий підхід передбачає використання лагів – тобто значень ціни, зафіксованих певний період часу тому. Наприклад, модель може отримати як вхід дані про ціну монети різного проміжку та інтервалу часів і дозволяє розпізнавати тренди, сезонність або циклічність у зміні вартості. Показники, як середнє значення по лагах, стандартне відхилення, сума змін, нахил (тенденція зростання або падіння) формують окремий блок статистичних характеристик, які підвищують інформативність вхідних даних.

Окрім часових факторів, у модель варто додати характеристики самих монет, зокрема фізичні та ринкові: вагу, діаметр, метал, номінал, тираж. Деякі з них можуть бути представлені у вигляді one-hot-кодування.

На відміну від класичних ARIMA або ETS-підходів, модель на основі бустингу не вимагає суворого припущення про стаціонарність ряду, а також здатна автоматично виявляти нелінійні залежності між ознаками, що робить її зручною у застосуванні до реальних ринкових даних, які часто є зашумленими, неповними або нерегулярними.

2.5.3 Рекомендаційна система на основі взаємодій

У системах, де на першому місці стоїть індивідуальний досвід користувача, механізм рекомендацій відіграє ключову роль. Він покликаний збільшити відповідність контенту, що відображається, скоротити час, витрачений на пошук, і зосередити увагу користувача на об’єктах, які найбільше відповідають його особистим зацікавленням. В системі для формування рекомендацій використано підхід колаборативної фільтрації, який ґрунтується на аналізі історії переглядів.

З технічної точки зору, модель реалізована через матричну факторизацію – метод, який дозволяє розкласти взаємодії між користувачами та об’єктами на приховані характеристики. Для цього

застосовується бібліотека ML.NET, що містить вбудований тренер MatrixFactorizationTrainer. Модель навчається на парах користувач-монета, де кожна взаємодія трактується як позитивний показник інтересу.

Однією з суттєвих переваг такого підходу є те, що він не потребує чітких характеристик монет чи профілів користувачів – система функціонує виключно на основі накопичених взаємодій, забезпечуючи швидкий запуск рекомендатора навіть за відсутності детальної предметної інформації. Модель також легко масштабується та дозволяють повторне використання в середовищі сервера без зовнішніх залежностей.

3 ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБДОДАТКУ

3.1 Опис технічного завдання на розробку

Мета реалізації вебдодатку – розробити експертну систему для вивчення, оцінки та аналітичної обробки інформації про монети. Передбачається, що платформа поєднає в собі функціонал довідника, інструмента для отримання фахової інформації та інтерфейсу, орієнтованого на інтерактивну роботу з даними.

Базова структура містить каталог монет із можливістю фільтрування. Користувач повинен мати можливість обирати монети за наступним переліком параметрів: країна походження, рік випуску, метал, валюта, номінал. Пошук повинен бути нечутливим до регістру, працювати як за назвами, так і за числовими значеннями, включаючи номінал у поєднанні з відмінюванням назви валюти. Результати пошуку та фільтрації відображаються у вигляді карток монет з аверсом, реверсом, характеристиками та ціною.

Під час розробки було сформульовано низку задач, що визначають логіку реалізації:

– створити зручний інтерфейс перегляду монет з коротким описом та доступом до повних характеристик;

– реалізувати динамічну систему фільтрів і сортування за кількома критеріями одночасно;

– забезпечити підтримку обраного для авторизованих користувачів, а також запис в історію переглядів монет;

– впровадити прогнозну модель, яка на основі історичних цін формує передбачення зміни вартості на обраний період часу;

– розробити систему візуальної оцінки стану монети, що працює з двома зображеннями і класифікує їх за нумізматичними стандартами;

– інтегрувати рекомендаційний модуль, що формує добірку монет на основі особистої історії переглядів;

– передбачити механізм автоматичного оновлення даних із зовнішніх джерел через скрипти парсингу.

Окремо було визначено перелік ролей користувачів системи, кожна з яких має свій набір доступних функцій:

– гостьовий користувач, котрий може переглядати каталог, виконувати пошук та ознайомлюватися з аналітикою, без слідів активності;

– авторизований користувач, який додатково отримує доступ до списку обраних монет, історії взаємодії з каталогом, функції оцінки стану монет і блоку персоналізованих рекомендацій;

– адміністратор, що має змогу запускати парсери для оновлення даних, отримувати службові повідомлення й стежити за загальним станом системи.

Парсери отримують інформацію з відкритих джерел, так і формують структуровані таблиці у вигляді CSV. Ці таблиці наступним кроком імпортуються до бази як на етапі первинної ініціалізації, так і вручну через інтерфейс адміністратора.

Важливою рисою системи є зміщення акценту з пасивного перегляду на активне аналітичне використання. Інтерфейс має не лише показувати дані, а й пояснювати їхню динаміку. Для цього використовуються прогнози, графіки зміни вартості, рекомендації на основі взаємодій користувача та оцінка стану за допомогою машинного навчання та методів штучного інтелекту. Дані кроки спрямовані на те, щоб зробити платформу корисною як для досвідченого колекціонера, так і для новачка.

3.2 Структура проєкту

Проєкт MintIQ розроблений на основі архітектурної моделі MVC, де функціональність розділена між моделями, представленнями та

контролерами. Також система включає інші додаткові рівні. Проєкт складається з наступних функціональних модулів, що згруповані відповідними папками (рисунок 3.1):

– controllers, що відповідають за обробку запитів, що надходять від користувача шляхом прийому параметрів з адресного рядка або форм, які наступним чином взаємодіють із сервісами та передають результат до представлень;

– models, які містять визначення об’єктів, які зберігаються в базі даних та задіяні для опису атрибутів, що властиві сутностям (монета, користувач, історія переглядів тощо);

– views, що виконують роль посередника у взаємодії користувача через інтерфейс шляхом відображення таких функціональних елементів, як: каталоги монет, графіки, результати оцінки, особистий профіль, форма входу;

– services, де реалізується бізнес-логіка, а також виступають допоміжними елементами для контролерів задля уникнення перенавантаження їх зайвим обчисленням та досягнення чистоти коду;

– DTOs, котрі є проміжними структурами даних, що покликані передавати лише необхідні дані від сервісів до представлень;

– ML, який використовується для взаємодії з методами машинного навчання та штучного інтелекту шляхом запуску Python-скриптів, що в свою чергу тренують та зберігають моделі, передають вхідні дані та обробляють отримані результати;

– parsers, де містяться окремі скрипти, що займаються отриманням даних з зовнішніх джерел та виконуються незалежно від основного додатку, зберігаючи інформацію у вигляді табличних даних;

– utils, що виступають допоміжним компонентом у реалізації функцій, які винесені із логіки сервісів через застосування реалізації на різних елементах проєкту;

– wwwroot, який містить в собі стилі, скрипти, що реалізовані мовою JavaScript та зображення для використання в рамках вебдодатку;

– конфігураційні файли Program.cs та appsettings.json, які відповідальні за налаштування підключення до бази даних, маршрутів, сесії, а також реєстрування всіх компоненти, які використовуються в програмі.

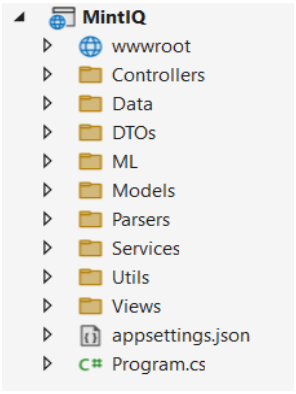

Рисунок 3.1 – Функціональна структура проєкту

В результаті наступної ізоляції компонентів було досягнуто чіткої визначеності в структурі, незважаючи на широкий спектр функціональних можливостей.

3.3 Моделі та база даних

Вебдодаток зберігає структуровані дані в базі PostgreSQL, де кожна таблиця репрезентує окрему сутність в межах предметної області. Для взаємодії з цими таблицями на рівні коду, у папці Models створено відповідні моделі. Вони віддзеркалюють структуру таблиць та використовуються Entity Framework Core для здійснення запитів, змін та реалізації зв’язків між даними (їх схема наведена на рисунку 3.2).

Усі зовнішні зв’язки реалізовано за допомогою зовнішніх ключів, а поля для зображень та результатів оцінювання зберігаються як шляхи до файлів, без використання BLOB-типів.

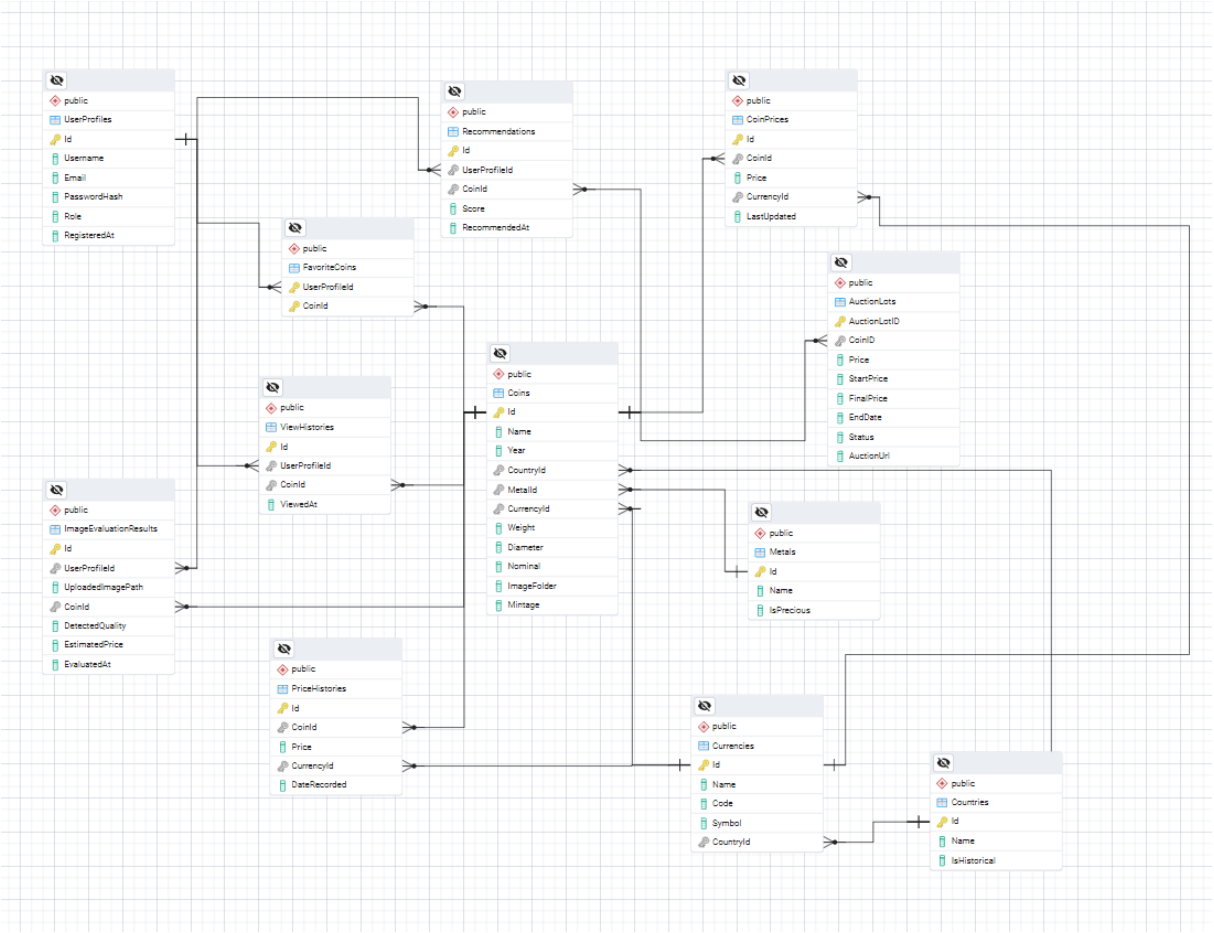

Рисунок 3.2 – ER-діаграма вебдодатку MintIQ

В проєкті використовується наступний перелік таблиць бази даних:

– auctionLots, таблиця з лотами, що містить інформацію про ціну, дату завершення, статус та зовнішнє посилання;

– coinPrices, таблиця з актуальними цінами на монети з інформацією про валюту та дату оновлення;

– coins, основна таблиця, що описує кожну монету назву, рік випуску, номінал, вагу, метал, країну, валюту, а також каталоговий ідентифікатор зображень;

– countries, довідник країн, що використовуються при описі монет та валют;

– currencies, перелік валют, включаючи код, символ та країну, якій вони належать;

– favoriteCoins, таблиця, що описує зв’язок між користувачами та монетами для збереження обраного;

– imageEvaluationResults, таблиця з результатами оцінювання монет за зображенням, з полем якості та датою оцінки;

– metals, довідник металів, що використовується для класифікації монет за матеріалом;

– priceHistories, історія змін вартості монет з прив’язкою до дати та валюти;

– recommendations, таблиця персоналізованих рекомендацій, пов’язаних із користувачами;

– user_profiles, облікові записи користувачів з базовими атрибутами: ім’я, пошта, хеш пароля, роль, дата реєстрації;

– view_histories, записи про перегляди монет користувачами з інформацією про час події.

Окрім моделей, що прямо представляють таблиці бази даних, у структурі проєкту наявний службовий клас, який задіяний у представлені. CoinListViewModel – це модель для відображення списку монет у каталозі, містить параметри фільтрування, перелік монет, номінали, роки та іншу інформацію, що створюється динамічно.

3.3.1 Таблиця Coins

Таблиця Coins є ключовим елементом в структурі бази даних. Вона зберігає основну інформацію про кожну монету, включаючи її назву, рік випуску, номінал, країну-емітента, матеріал, валюту, а також технічні параметри. Додатково, в таблиці міститься шлях до каталогу з фотографіями монети та загальний тираж випуску. Для кожного запису передбачені зовнішні ключі, що вказують на відповідні довідники: країн, металів та валют. Таблиця Coins є відповідником до однойменного класу Coin в проєкті, схема якого наведена на рисунку 3.3.

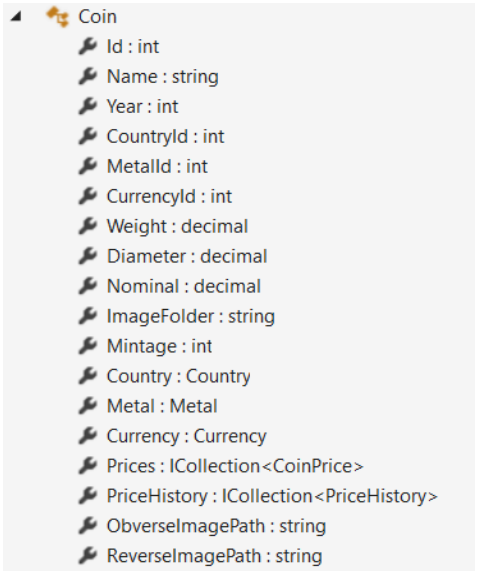

Рисунок 3.3 – Діаграма класу Coin

Нижче перелік полів таблиці Coins:

– id, ідентифікаційний номер монети, первинний ключ;

– name, найменування монети;

– year, рік карбування;

– country_id, зовнішній ключ, пов’язаний з таблицею Countries;

– metal_id, зовнішній ключ, що посилається на таблицю Metals;

– currency_id, зовнішній ключ для таблиці Currencies;

– weight, вага монети у грамах;

– diameter, діаметр монети в міліметрах;

– nominal, номінальна вартість монети;

– image_folder, назва теки, де розміщені зображення;

– mintage, тираж монет.

3.3.2 Таблиці Countries, Currencies, Metals

Для зберігання стандартизованої інформації, яка дублюється в основній таблиці монет, база даних містить окремі довідники. До них відносяться Countries, Currencies та Metals, які використовуються для

забезпечення зовнішніх зв’язків із таблицею Coins, зберігання метаінформації про країни, валюти та метали, а також мінімізації дублювання текстових значень у базі даних.

Таблиця Countries зберігає дані про країни, до яких належать монети (відповідна діаграма класу наведена на рисунку 3.4):

– id, ідентифікатор країни;

– name, назва країни;

– is_historical, ознака існування країни в минулому (логічне значення).

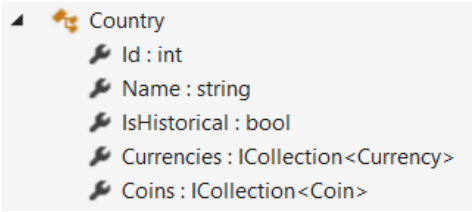

Рисунок 3.4 – Діаграма класу Countries

Таблиця Currencies містить інформацію про валюту (діаграма на рисунку 3.5):

– id, ідентифікатор валюти;

– name, назва валюти;

– code, літерний код валюти;

– symbol, символ чи скорочення валюти;

– country_id, зовнішній ключ до таблиці Countries.

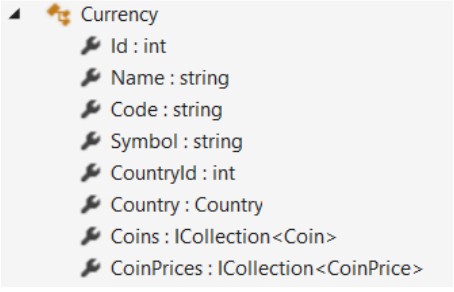

Рисунок 3.5 – Діаграма класу Currency

Таблиця Metals використовується для класифікації монет за матеріалом виготовлення (рисунок 3.6):

– id, ідентифікатор металу;

– name, назва металу;

– is_precious, ознака дорогоцінності (логічне значення).

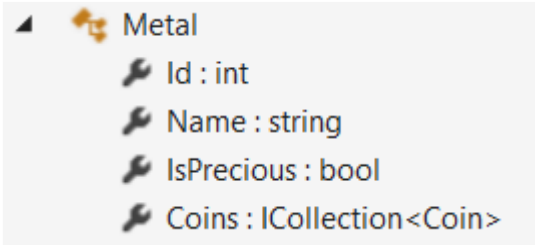

Рисунок 3.6 – Діаграма класу Metal

Кожна з цих таблиць має відповідний клас у програмній реалізації: Country, Currency, Metal. Усі вони використовуються для створення зв’язків типу «один до багатьох» із таблицею Coins.

3.3.3 Таблиці CoinPrices, PriceHistories

Зміни вартості монет є ключовим маркером ринку, тому для їх фіксації використовуються дві таблиці. CoinPrices зберігає поточні ціни монет, тоді як PriceHistories містить повну історію коливань вартості, що дозволяє швидко отримувати актуальну інформацію та одночасно аналізувати зміни в довгостроковій перспективі. Таблиця CoinPrices містить наступні характеристики (діаграма класу зображена на рисунку 3.7):

– id, первинний ключ;

– coin_id, посилання на таблицю Coins;

– price, числове значення вартості;

– currency_id, ідентифікатор валюти;

– last_updated, дата та час останнього оновлення.

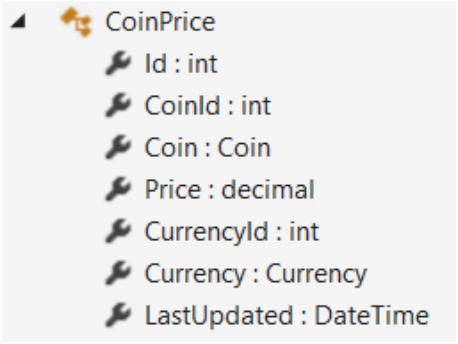

Рисунок 3.7 – Діаграма класу CoinPrice

Таблиця PriceHistories містить такі поля (рисунок 3.8):

– id, первинний ключ;

– coin_id, посилання на Coins;

– price, вартість, зафіксована на певну дату;

– currency_id, валюта, в якій вказано вартість;

– date_recorded, дата реєстрації ціни.

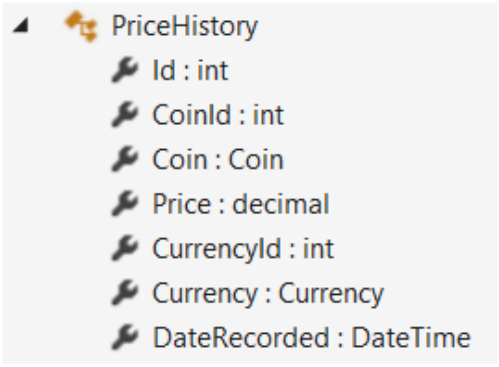

Рисунок 3.8 – Діаграма класу PriceHistory

Зважаючи на подібність даних, таблиці мають різне призначення. Таблиця PriceHistories веде запис коливання цін, що необхідно для вивчення трендів та формування передбачень. Натомість CoinPrices є зрізом, де зберігається лише актуальна ціна для кожної монети на поточний момент. Такий підхід було застосовано з метою оптимізації: завдяки таблиці

CoinPrices, система уникає повторних SQL-запитів для отримання останнього значення з історичних даних, що в свою чергу пришвидшує роботу інтерфейсу.

3.3.4 Таблиці UserProfiles, FavoriteCoins, ViewHistories

Для зберігання відомостей про користувачів, їхні дії та персоналізовану взаємодію з каталогом, у базі даних використовується три окремі таблиці. Основні дані про облікові записи зберігаються у UserProfiles, в той час як FavoriteCoins використовується для фіксації вподобаних монет, а ViewHistories – для ведення історії переглядів.

Таблиця UserProfiles містить такі поля та таку структуру класу (рисунок 3.9):

– id, ідентифікатор;

– username, ім’я, яке відображається користувачам;

– email, адреса електронної пошти;

– password_hash, шифрування пароля;

– role, роль користувача у системі;

– registered_at, дата реєстрації облікового запису.

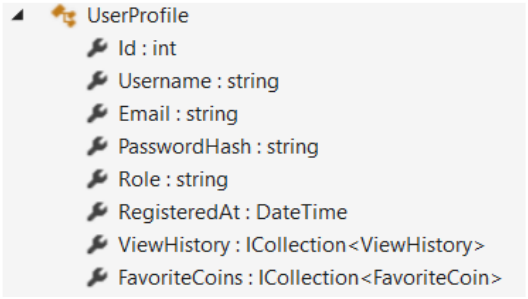

Рисунок 3.9 – Діаграма класу UserProfile

Таблиця FavoriteCoins реалізує зв’язок «багато-до-багатьох» між користувачами та монетами (рисунок 3.10):

– user_profile_id, зовнішній ключ, що посилається на UserProfiles;

– coin_id, зовнішній ключ, що посилається на Coins.

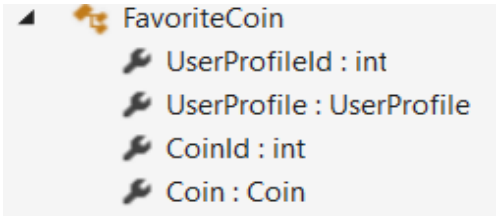

Рисунок 3.10 – Діаграма класу FavoriteCoin

Таблиця ViewHistories фіксує кожен перегляд монети авторизованим користувачем (рисунок 3.11):

– id, первинний ключ;

– user_profile_id, зовнішній ключ, що посилається на UserProfiles;

– coin_id, зовнішній ключ, що посилається на Coins;

– viewed_at, дата й час перегляду.

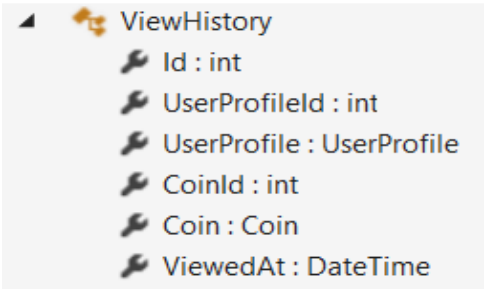

Рисунок 3.11 – Діаграма класу ViewHistory

Таблиця UserProfiles застосовується для автентифікації, керування ролями та надання доступу до персоналізованих функцій. Дані з FavoriteCoins використовуються для відображення списку обраних монет та оцінки їхньої популярності. ViewHistories служить для генерації персоналізованих рекомендацій, формування хронології переглядів та визначення трендових позицій.

3.3.5 Таблиці Recommendations, ImageEvaluationResults, AuctionLots

У структурі бази даних також передбачені таблиці, кожна з яких надає специфічні можливості системі. Вони охоплюють інтелектуальні підказки, автоматичний аналіз візуального стану монет, а також зберігання зовнішньої інформації, отриманої з відкритих джерел. Таблиця Recommendations призначена для зберігання персоналізованих пропозицій, сформованих для конкретного користувача (діаграма класу продемонстрована на рисунку 3.12):

– id, первинний ключ;

– user_profile_id, зовнішній ключ, що посилається на таблицю користувачів;

– coin_id, зовнішній ключ на рекомендовану монету;

– score, числова оцінка релевантності;

– recommended_at, дата формування рекомендації.

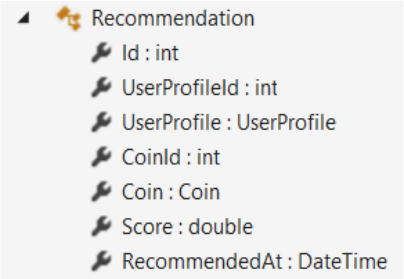

Рисунок 3.12 – Діаграма класу Recommendation

Таблиця ImageEvaluationResults зберігає результати оцінювання зображень, завантажених користувачами (рисунок 3.13):

– id, ідентифікатор запису;

– user_profile_id, користувач, який здійснив оцінювання;

– uploaded_image_path, шлях до файлу на сервері;

– coin_id, монета, якщо вдалося ідентифікувати;

– detected_quality, визначена якість;

– estimated_price, орієнтовна ціна;

– evaluated_at, час проведення оцінки.

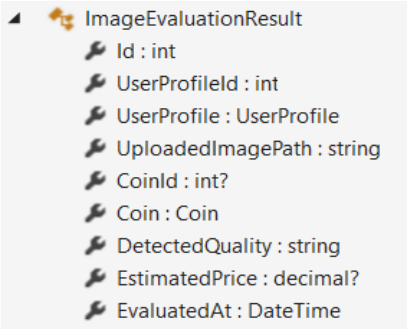

Рисунок 3.13 – Діаграма класу ImageEvaluationResult

Таблиця AuctionLots призначена для зберігання інформації про лоти, представлені на аукціонних платформах (рисунок 3.14):

– auction_lot_id, ідентифікатор лоту;

– coin_id, відповідна монета;

– price, поточна ціна;

– start_price, стартова вартість;

– final_price, фінальна ціна (або моментальна ставка);

– end_date, дата завершення торгів;

– status, результат (продано / не продано);

– auction_url, URL-адреса на джерело.

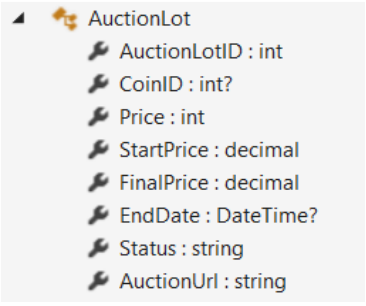

Рисунок 3.14 – Діаграма класу AuctionLot

Дані з таблиці Recommendations використовуються для формування персоналізованого відображення на головній сторінці та у відповідному розділі рекомендацій (рисунок 3.15). Результати з ImageEvaluationResult використовуються для оцінки монет за зображеннями та зберігаються в історії оцінювання. Таблиця AuctionLot надає доступ до прикладів актуальних або історичних пропозицій, що дає системі змогу оперувати реальними ринковими даними.

Вся логіка взаємодії з базою даних у вебдодатку організована у директорії Data. Ключовим компонентом є клас MintIQDbContext, що успадковує DbContext з бібліотеки Microsoft.EntityFrameworkCore. Клас діє як головний посередник між програмним кодом та реляційною базою даних PostgreSQL, з якою система взаємодіє. В ньому визначені всі таблиці як властивості типу DbSet<TEntity>, а також налаштована конфігурація полів та зв’язків між сутностями через метод OnModelCreating (рисунок 3.15).

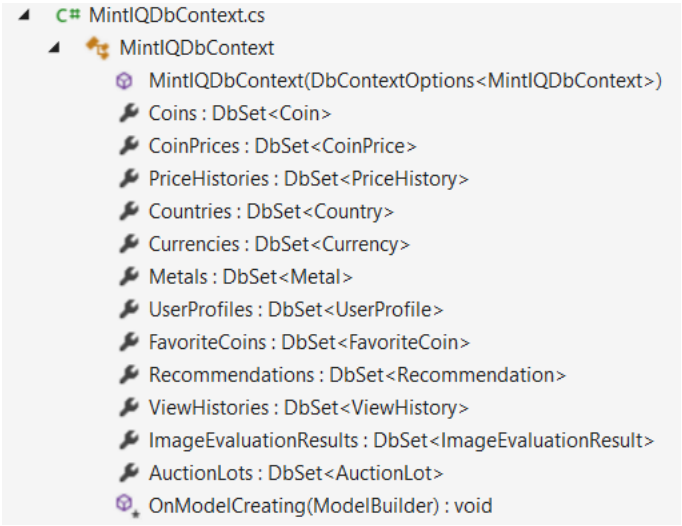

Рисунок 3.15 – Діаграма класу MintIQDbContext

Контекст визначає налаштування типів даних та розмір числових значень. Також тут описані складені ключі та взаємодії для таблиць зі складною структурою, наприклад, FavoriteCoins, що реалізовує взаємодію багато-до-багатьох між монетами та користувачами.

Крім власне контексту, у цьому каталозі міститься клас SeedData. Він містить логіку первинного наповнення бази даних з CSV-файлів, що дозволяє відокремити всю ініціалізацію та роботу з даними в межах однієї сфери відповідальності, дотримуючись принципів чистої архітектури (рисунок 3.16).

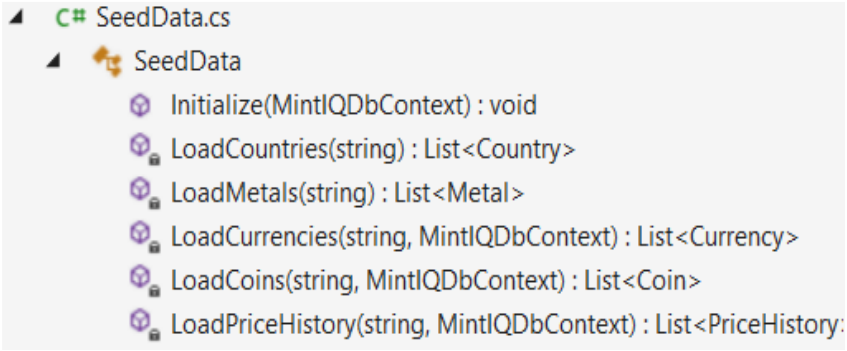

Рисунок 3.16 – Діаграма класу SeedData

Організація в такій формі надає зручну підтримку та можливість масштабування проєкту, що дозволяє централізовано управляти всіма аспектами взаємодії з базою даних. Більше того, чітке розділення логіки ініціалізації (SeedData) та роботи з контекстом (MintIQDbContext) сприяє поліпшенню тестованості та розширюваності системи.

3.4 Розробка контролерів

У вебдодатку контролери виконують надзвичайно важливу функцію: на них покладена відповідальність за перенаправлення запитів, взаємодію із сервісами, перевірку даних на відповідність та формування результату, що повертається як сторінка або JSON. З точки зору архітектури, це дає змогу розмежувати обов’язки між інтерфейсом, обробкою та доступом до даних, що значно полегшує масштабування та підтримку проєкту.

Кожен контролер у MintIQ має конкретну мету, зосереджену на окремій частині функціональності. Одні відповідають за взаємодію з користувачами, інші – за запуск алгоритмів машинного навчання чи зовнішніх сценаріїв.

3.4.1 AccountController

Контролер AccountController відповідає за автентифікацію користувачів, реєстрацію, показ профілю та вихід із системи. Всі взаємодії з базою даних здійснюються через UserService, що вміщує перевірки та операції з таблицями UserOrofiles і FavoriteCoins. Для автентифікації використовується механізм cookie, критичні методи доступні лише через POST-запити (рисунок 3.17).

Метод Login (GET) відображає форму для входу. У випадку, якщо в TempData є попередні сповіщення (наприклад, про помилку або вдалу реєстрацію), вони виводяться в інтерфейсі. Відповідний POST-метод обробляє вказані email та пароль. Пароль хешується за допомогою алгоритму SHA-256, після чого перевіряється в базі даних за допомогою UserService.AuthenticateAsync. Якщо користувача не знайдено – повертається повідомлення про помилку. У разі успішної автентифікації формуються клейми з ID, іменем та роллю користувача, які упаковуються у ClaimsPrincipal та записуються в cookie через SignInAsync.

Методи Register працюють аналогічно: GET-метод відображає форму, POST перевіряє унікальність email, хешує пароль і викликає RegisterUserAsync. Успішна реєстрація супроводжується перенаправленням на сторінку входу з відповідним сповіщенням/

Метод Profile доступний тільки для авторизованих користувачів. З клеймів отримується ID, на основі якого видобувається повний профіль. У разі невдачі повертається код 404. Вихід з системи (Logout, POST) викликає SignOutAsync і очищає cookie.

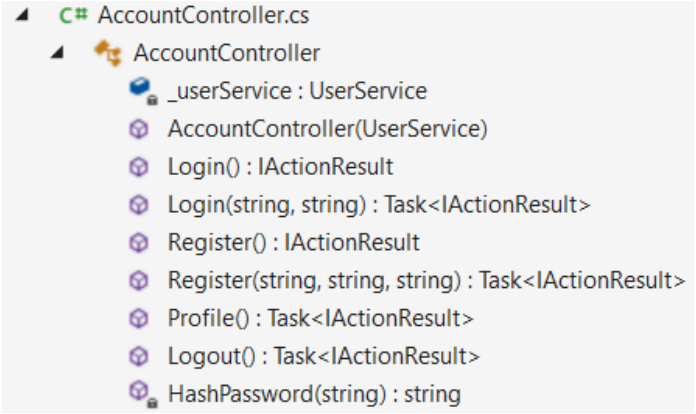

Рисунок 3.17 – Діаграма класу AccountController

Окремо реалізовано приватний метод HashPassword. Він конвертує пароль у масив байтів, хешує за допомогою алгоритму SHA-256 та повертає Base64-рядок. Це детерміноване хешування, яке не є найбезпечнішим, але є прийнятним для навчальних цілей.

3.4.2 AdminController

Контролер AdminController відіграє хоч і невелику за обсягом, проте значущу функцію надаючи адміністраторський інтерфейс для ручного старту парсерів. В межах додатку цей контролер виокремлюється, оскільки не пов’язаний з обробкою запитів користувачів в загальному розумінні, а спрямований на службову функціональність. Головна дія Index віддає представлення з кнопкою активації фонових процесів, а метод RunParsers відповідає за безпосереднє виконання запуску (рисунок 3.18).

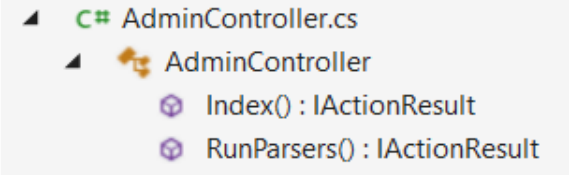

Рисунок 3.18 – Діаграма класу AdminController

Реальна логіка запуску парсерів розміщена в класі ParserScheduler, який викликається напряму з контролера. В разі успішного завершення парсингу користувачу висвічується сповіщення у TempData. Якщо виникає виключення, воно перехоплюється та відображається повідомлення про помилку.

3.4.3 AnalyticsController

Контролер AnalyticsController виступає ключовою ланкою, з'єднуючи інформацію про монету з бази даних з алгоритмами машинного навчання. Його функція – надання користувачеві аналітичного огляду, заснованого на історії ціни монети та її параметрах. Уся логіка сконцентрована у методі Index, який взаємодіє з базою даних через MintIQDbContext та використовує сервіси MLService і RecommendationService (рисунок 3.19).

Метод Index, доступний через HTTP GET, потребує параметр coinId. Спочатку виконується витяг з бази даних повної інформації про монету, з урахуванням зв'язків. Для цього використовується метод Include, що забезпечує «жадібне» завантаження пов'язаних об'єктів.

Далі відбувається окремий запит до таблиці CoinPrices, де визначається актуальна ціна монети. Значення передається до представлення як поле LatestPrice. Важливо підкреслити, що цей запит незалежний від історії змін, що допомагає зменшити обчислювальне навантаження та прискорює відповідь.

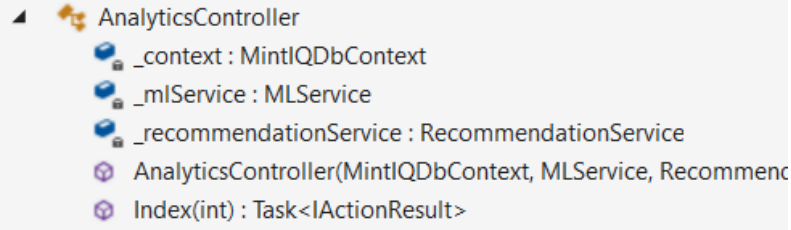

Рисунок 3.19 – Діаграма класу AnalyticsController

Після отримання основної інформації, запускається метод GetForecastWithScoreAsync з MLService, який відповідає за обчислення прогнозу цін. Алгоритм, працюючи з ID монети, витягує всі необхідні дані (історію цін, технічні характеристики) та повертає словник Forecasts, де ключем є горизонт прогнозу (у днях), а значенням – об’єкт з дельтою ціни, відсотковою зміною та текстовим описом тренду.

Всі зібрані дані об’єднуються у структуру CoinDetailsDTO, яка містить об’єкт монети, її актуальну ціну, прогноз та точність. На завершення, метод повертає оброблену сторінку «Index» у представлення, де інформація візуалізується для користувача. Графіки та табличні блоки для прогнозу формуються на фронтенді, використовуючи JavaScript.

3.4.4 CoinsController

Контролер CoinsController є одним з найбільш наповнених функціоналом в структурі MintIQ. Він забезпечує відображення каталогу монет, обробку фільтрів, демонстрацію деталей, взаємодію з обраним, історією переглядів та асинхронне оновлення рекомендацій. Основна логіка сконцентрована на взаємодії з базою даних через MintIQDbContext, а допоміжна – делегована відповідним сервісам (рисунок 3.20).

Метод Index обробляє головну сторінку каталогу. Він підтримує різні параметри: пошук, фільтрацію за країнами, металами, валютами, роками випуску, номіналами, діапазоном цін, а також сортування. Насамперед формується запит до бази даних з жадібним завантаженням пов’язаних таблиць (за допомогою Include), далі застосовуються фільтри. Обрані параметри, якщо користувач авторизований, доповнюються умовами для переглянутих або улюблених монет.

На етапі формування представлення, ViewBag наповнюється колекціями фільтрів, які обраховуються на базі вже відібраних монет і дозволяє одночасно здійснювати фільтрацію та підрахунок відповідних

кількостей (для інтерфейсу). Після цього відбувається підрахунок діапазону цін, та формування моделі CoinListViewModel, що передається до View.

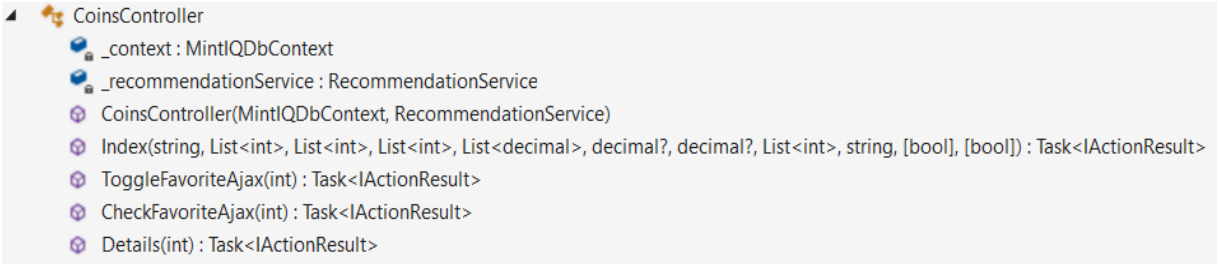

Рисунок 3.20 – Діаграма класу CoinsController

Метод ToggleFavoriteAjax, який обробляє POST-запити, відповідає за додавання або видалення монети з обраного. Спочатку перевіряється автентифікація, далі, в залежності від наявності монети у списку обраного, вона або видаляється, або додається. Після цього повертається результат у форматі JSON. Аналогічно працює CheckFavoriteAjax, що лише перевіряє, чи додана монета до обраного.

Метод Details, що відповідає за відображення окремої монети, реалізує ряд послідовних дій. Спершу через SQL-запит обирається монета з усіма необхідними зв’язками. Далі окремим запитом вибирається остання ціна, потім – повна історія зміни цін. До цього додається список схожих монет, отриманий через RecommendationService, і курс валют, завантажений з файлу за допомогою ExchangeRateLoader.

Після підготовки всіх основних даних відбувається запис перегляду: якщо користувач авторизований, у ViewHistories створюється новий запис. Далі в окремому асинхронному потоці, без використання поточного контексту бази, запускається генерація рекомендацій через GenerateRecommendationsForUserAsync. Завдяки такому підходу вся критична логіка – запис історії та оновлення рекомендацій.

3.4.5 EvaluationController

Контролер EvaluationController втілює в собі логіку взаємодії з модулем оцінювання стану монет за їхніми зображеннями. Його головна задача – забезпечити передачу необхідної інформації від користувача до сервісу машинного навчання, а також сформувати модель представлення з урахуванням вибраної монети, року випуску та країни (рисунок 3.21).

Метод Index (GET) відповідає за завантаження ієрархічної структури монет, впорядкованих за країнами, назвами та роками випуску. Вся інформація видобувається з бази даних і представляється у вигляді вкладеного словника, що робить зручною динамічну побудову каскадного списку вибору.

Метод UploadAndEvaluate (POST) займається обробкою завантажених зображень аверсу та реверсу, а також інформації про обрану монету. Після перевірки, інформація надсилається до методу EvaluateAsync сервісу ImageEvaluationService, який об’єднує зображення в єдиний вхідний параметр для моделі навчання. Результатом є об’єкт, що містить інформацію про визначений стан монети та дату оцінки.

Рисунок 3.21 – Діаграма класу EvaluationController

Результати оцінювання відображаються на інтерфейсі разом з обробленими зображеннями. Вся ML-логіка та робота із зображеннями

винесена на рівень сервісу, що дозволяє підтримувати чистоту контролера та легко розширювати функціональність.

3.4.6 HomeController

Контролер HomeController виконує роль головного обробника запитів для головної сторінки додатку. Його головна задача полягає у створенні початкового набору монет, що будуть запропоновані користувачеві. У випадку, коли користувач авторизований, його унікальний ідентифікатор отримується з ClaimTypes.NameIdentifier, і для нього формується персоналізований перелік рекомендованих монет (рисунок 3.22).

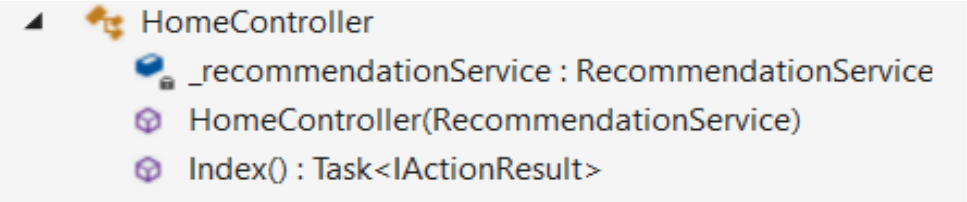

Рисунок 3.22 – Діаграма класу HomeController

Даний контролер застосовує RecommendationService для двох основних цілей: формування загального списку найпопулярніших монет, та окремої добірки монет, ціна на які зазнала зниження (знижки).

3.5 Застосування інструментів МН та ШІ

Інтелектуалізація у вебдодатку MintIQ є ключовим компонентом, що значно розширює можливості системи. Завдяки інтеграції моделей прогнозування, класифікації та рекомендацій, програма надає важливу та цінну для користувача інформацію. Структура побудована на залучені навчених моделей шляхом запуску Python-скриптів та інших елементів для отримання результату.

Клас MLService як раз відповідає за їх запуск, контроль за виконанням, отримання результатів та їх формування у DTO-структури для подальшої обробки у контролерах, відтак роль сервісу можна охарактеризувати як оркестраційну та формує центральну точку у логіці отримання аналітичних даних.

Донавчання відбувається лише у моделі рекомендації з тої простої причини, що на навчальних потужностях оновлення моделей прогнозування ціни та оцінки моделей є неможливою через великий обсяг та складність інформації. Результати навчання моделей зазначені у додатку А в таблицях А.1 та А.2.

3.5.1 Прогнозування динаміки цін монети

Окремий складник системи займається прогнозуванням коливання ціни монети, виходячи з історичних відомостей про неї. Тут застосовано підхід навчання з учителем, зокрема градієнтний бустинг – а саме LightGBM. Він дає змогу опрацьовувати великі масиви даних та брати до уваги як часові, так і структурні особливості об’єктів.

LightGBM (Light Gradient Boosting Machine) – це програмний каркас градієнтного бустингу від Microsoft, спеціалізований на створенні моделей на основі ансамблю рішень-дерев. Головною його перевагою є продуктивність – як у швидкості навчання, так і в обсязі споживаної пам’яті. На відміну від традиційних алгоритмів, LightGBM застосовує техніку «leaf-wise» побудови дерев з глибшим розгалуженням у багатообіцяючих гілках, що дозволяє досягти кращої точності за меншої кількості ітерацій.

Перш ніж розпочати навчання, модель отримує два основні джерела інформації: таблицю історії цін та набір атрибутів кожної монети. Перша таблиця містить часовий ряд, який служить основою для створення лагових ознак. Для кожної монети обирається щонайменше 10 попередніх значень з інтервалом у 7 днів – ці значення формують так звані лаги.

Додатково також формуються статистичні показники: середнє значення, стандартне відхилення, різниця між двома останніми значеннями, а також кумулятивний нахил. Мета полягає в тому, щоб сформувати ознаки, котрі відображають не лише значення, але й динаміку змін.

До загального переліку ознак додаються також властивості монети: вага, діаметр, тираж, номінал та тип металу. Тип металу кодується за допомогою one-hot кодування: кожен варіант («золото», «срібло», «інше») подається у вигляді вектора. Після підготовки даних застосовується навчання моделі з наступною серіалізацією в .pkl-файл, основний фрагмент якого зазначений у лістингу 3.1.

Лістинг 3.1 – Програмний код виклику алгоритму машинного навчання та збереження моделі

```
model = lgb.LGBMRegressor(n_estimators=100, max_depth=7)
model.fit(X, y)
with open("price_trend_model.pkl", "wb") as f:
    pickle.dump((model, LAGS), f)
```

У процесі роботи, коли потрібно створити прогноз, модель завантажується з файлу, а потім, ґрунтуючись на ній, поетапно прогнозуються ціни для горизонтів у 7, 30, 180 та 365 днів. На кожному етапі формується набір ознак, виходячи з останніх відомих даних. Після визначення прогнозу, результат коригується з використанням коефіцієнта чутливості (SENSITIVITY_FACTOR), який або послаблює, або підсилює тренд, залежно від різниці між попереднім та поточним значенням, що в свою чергу допомагає уникнути надмірно різких змін у прогнозі.

Отримані дані зберігаються у JSON-форматі, який далі обробляє сервіс MLService. Він аналізує структуру, перетворює передбачені дельти на текстові описи, і передає їх до інтерфейсу користувача для відображення.

3.5.2 Оцінювання стану монети

Оцінка стану монети за зображенням – одна з основних можливостей платформи MintIQ, що поєднує комп'ютерний зір з методами класифікації глибокого навчання. З технічної точки зору, реалізація базується на нейронній мережі MobileNetV3, навченій розпізнавати якісні характеристики монет за фотографіями їхнього аверсу та реверсу.

Навчальна вибірка формується з даних про минулі продажі, де для кожного запису збережено два зображення та клас стану (Grade). Ці зображення об'єднуються в одне, і в процесі тренування передаються в модель розміром 224×448 пікселів. Для зменшення зміщення класів застосовується метод RandomOverSampler, а для підвищення здатності до узагальнення – аугментація (наприклад, обертання, зміна кольору, віддзеркалення). Кожен клас стану монети кодується у вигляді мітки та вектора one-hot для обробки.

За основу моделі береться MobileNetV3-small, в якій останній шар замінено на повнозв'язний з кількістю виходів, рівною кількості класів. Навчання відбувається з використанням функції втрат CrossEntropyLoss, оптимізатора SGD та механізму ранньої зупинки на основі макро-F1-міри. Найкраща модель експортується у формат ONNX, як показано в лістингу 3.2.

Лістинг 3.2 – Навчання та експорт моделі класифікації якості

```
model=models.mobilenet_v3_small(weights=models.MobileNet_V3_Small_Weights.DEFAULT)
model.classifier[3]=nn.Linear(model.classifier[3].in_features,len(le.classes_))
model = model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer=torch.optim.SGD(model.parameters(),lr=0.01,momentum=0.9)
```

Продовження лістингу 3.2

```
dummy_input = torch.randn(1, 3, 224, 448).to(device)
torch.onnx.export(model, dummy_input, ONNX_PATH,
                  input_names=["input"],
                  output_names=["quality"],
                  opset_version=11)
```

Після того, як навчання буде завершено, найкращий варіант моделі експортується у форматі ONNX. Далі модель буде застосовуватися у робочому середовищі за допомогою скрипту на Python, який бере шлях до скомбінованого зображення, проводить інференс та повертає результат у форматі JSON.

3.5.3 Рекомендаційна система

Останнім ключовим елементом машинного навчання в архітектурі MintIQ є система персоналізованих рекомендацій, завдання якого є аналізування історії переглядів користувача, щоб формувати добірку монет, які з великою ймовірністю можуть його зацікавити (рисунок 3.23). На відміну від статичних механізмів, система адаптується до індивідуальної поведінки, з часом підвищуючи точність результатів.

Технічно, реалізація базується на методі матричної факторизації – одному з традиційних підходів у сфері рекомендованих систем. В цьому підході вся множина взаємодій користувачів з об’єктами представляється у вигляді матриці, де рядки відповідають користувачам, а стовпці – об’єктам. Якщо користувач взаємодіяв з конкретною монетою, у відповідній клітинці ставиться одиниця.

Розріджена матриця подається на вхід моделі, яка намагається знайти два менш розмірних латентних простори, множення яких наближає вихідну структуру. В результаті, кожен користувач та кожна монета отримують

числове представлення, а подальше обчислення скалярного добутку між ними дозволяє оцінити ймовірність взаємодії.

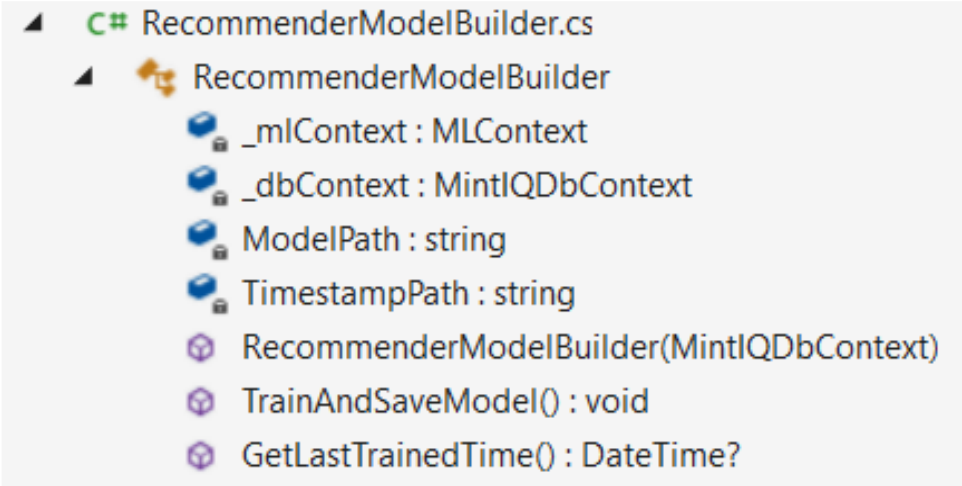

Рисунок 3.23 – Діаграма класу RecommenderModelBuilder

У реалізації застосовані засоби бібліотеки MatrixFactorizationTrainer. Модель навчається на основі таблиці переглядів ViewHistory, де кожен запис відповідає взаємодії між користувачем та монетою з фіксованою позначкою «1». Для навчання встановлено 6 ітерацій, а також латентний простір із 32 вимірами, що дозволяє досягти прийнятного балансу між точністю та швидкістю.

Після навчання модель зберігається у файл recommendation_model.zip, а разом з нею – дата останнього оновлення, що дозволяє не тільки використовувати її повторно, але й перевіряти актуальність. У випадку, якщо користувач здійснив нові перегляди, модель оновлюється у фоновому режимі без залучення основного потоку виконання.

В процесі генерації рекомендацій система перебирає всі можливі пари користувач-монета та оцінює їхню релевантність. В результаті виводяться ті монети, що мають найвищий рейтинг (Score) для поточного користувача.

3.6 Допоміжні елементи

3.6.1 Сервіси

Одним з наріжних принципів проєктування додатку MintIQ стало чітке розмежування функцій між компонентами (рисунок 3.24). Усі критично важливі обчислення, перевірки, логіка взаємодії з базами даних та зв’язки з ML або зовнішніми модулями покладено на спеціалізовані сервіси. Завдяки цьому контролери лишаються простими в своїй реалізації, зосередженими виключно на обробці HTTP-запитів.

UserService – сервіс, який опікується всім, що стосується реєстрації, входу, профілю користувача та улюблених монет. Зокрема, методи AuthenticateAsync і RegisterUserAsync перевіряють введені дані, хешують паролі, створюють нові облікові записи. Через цей же сервіс відбувається доступ до списку улюблених монет та історії переглядів. Таким чином, усі операції, що здійснюються з боку контролера AccountController, делегуються сюди.

CatalogService – ключовий елемент фільтрації та пошуку монет. Він виконує складні SQL-запити до бази, формує динамічні списки монет відповідно до заданих параметрів і повертає результат. Також на рівні сервісу реалізовано запити для створення фільтрів: для кожної групи повертається список з кількістю відповідних монет, що дає змогу показати користувачеві інформативні й точні варіанти вибору.

RecommendationService реалізує рекомендаційні алгоритми, зокрема отримання схожих монет, персоналізованих пропозицій, популярних або тих, що втратили у ціні. Він інтегрований з машинним навчанням і працює спільно зі збереженою моделлю. Усі рекомендації базуються на історії переглядів користувача, що дає змогу створити індивідуальні добірки.

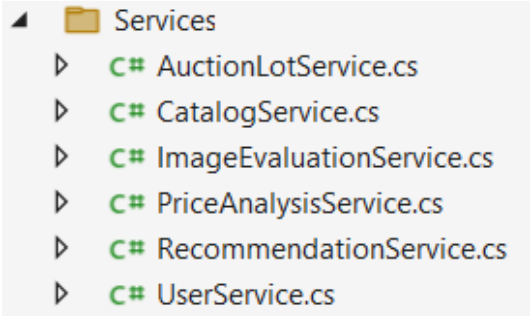

Рисунок 3.24 – Елементи модулю Services

PriceAnalysisService відповідає за базову цінову аналітику: середні значення, виведення історії змін, співвідношення до попередніх періодів. В його рамках також реалізовано метод для порівняння останньої ціни з передостанньою, що застосовується при формуванні списку монет зі знижками.

ImageEvaluationService – сервіс, який забезпечує взаємодію з моделлю класифікації якості монети за зображеннями. Після отримання двох фото (аверс і реверс), він створює об’єднане зображення та передає його до нейронної мережі. На виході користувач отримує клас, що дає змогу візуально оцінити стан монети. В свою ж чергу AuctionLotService обслуговує модуль з парсингом аукціонних лотів.

3.6.2 Утиліти

У проєкті MintIQ утиліти виконують функцію помічників, але значно полегшують роботу з форматами даних, відображенням результатів та локалізацією (рисунок 3.25). Ці компоненти не належать безпосередньо до контролерів чи сервісів, але активно застосовуються у процесі обробки даних або побудови інтерфейсу.

Першим заслуговує на увагу ExchangeRateLoader, який відповідає за зчитування архівних валютних курсів, сформованого за допомогою NbuExchangeParser. Він здійснює парсинг курсу гривні відносно долара та

євро з файлу, завдяки чому реалізується перерахунок історичних цін на монети в іноземні валюти та побудова динаміки в різних грошових одиницях.

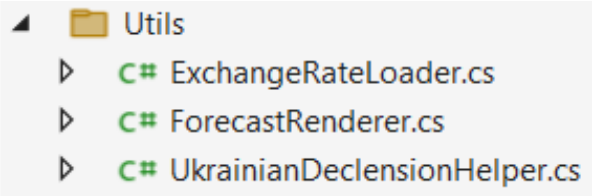

Рисунок 3.25 – Елементи модулю Utils

Другим є ForecastRenderer. Він перетворює об’єкт ForecastResultDTO, результат машинного прогнозування цін, на HTML-таблицю, що є зручним форматом для виведення у представленні. Клас реалізує два варіанти відображення: розгорнутий вигляд у вигляді таблиці з підсвічуванням трендів та стислий текстовий формат для аналітики.

Наостанок слід відзначити UkrainianDeclensionHelper – невеликий клас, який підбирає правильну форму відмінювання назва (гривня, гривні, гривень та ін.) залежно від числового значення номіналу. Він застосовується у відображенні назв монет у вигляді конструкцій типу та є важливим для зберігання правопису у відображенні інформації.

3.6.3 Парсери

Скрипти парсерів, написані мовою програмування Python, використовують бібліотеки requests та BeautifulSoup для отримання HTML-даних, а CSV – для збереження результатів. Додатково вони аналізують JavaScript-вставки на сторінці з метою збору та побудови історії цін. Нижче представлено фрагмент, який ілюструє один з аспектів роботи парсеру – витягування інформації із певних структур сторінки, що зображено у лістингу 3.3.

Лістинг 3.3 – Витяг характеристик монети зі сторінки HTML

```
    table=soup.select_one("table.coin-info")
    value_row=table.select("tr")[1] if table else None
    cells=value_row.find_all("td") if value_row else []
    year_val=extract_year(cells[0].text.strip())            if
len(cells)>0 else "n/a"
    nominal=cells[1].text.strip() if len(cells)>1 else "n/a"
    metal_raw=cells[2].text.strip() if len(cells)>2 else "n/a"
    metal=normalize_metal(metal_raw)
    weight=cells[3].text.strip() if len(cells)>3 else "n/a"
    diameter=cells[4].text.strip() if len(cells)>4 else "n/a"
    mintage=cells[5].text.strip().replace("шт.",   "").strip()
if len(cells)>5 else "n/a"
```

Скрипти працюють у зв’язці з механізмом запуску на сторінці (ParserScheduler), що дозволяє підтримувати актуальність бази даних без використання сторонніх API або ручного введення, завдяки чому досягається автономія в оновленні як метаінформації, так і цінової аналітики.

3.7 Представлення

У MintIQ візуалізація є ключовим елементом для взаємодії користувача з функціоналом вебдодатку. Представлення побудовані за допомогою Razor-розмітки в середовищі ASP.NET Core MVC, що дозволяє об’єднувати HTML та C# в одному шаблоні. Файли представлень організовані відповідно до логіки контролерів – кожна папка відповідає одному контролеру, а окремі сторінки реалізують конкретну візуальну дію (рисунок 3.26).

Сторінки облікового запису (Login, Register, Profile) містять інтерфейси для автентифікації користувача, створення нового профілю та

перегляду особистих даних. У профілі передбачено посилання на історію переглядів та обрані монети, а також кнопку виходу.

Представлення Index та Details, що знаходяться у розділі Coins, формують основу каталогу монет. На головній сторінці каталогу реалізована гнучка панель фільтрів, де користувач може здійснювати вибір монет за характеристиками. Кожна картка монети має опцію перемикання між аверсом і реверсом за допомогою анімації, а також кнопку для додавання монети до списку обраних. Детальна сторінка монети (Details.cshtml) містить не лише текстову інформацію, але й графік зміни ціни, таблицю прогнозів та динамічний блок з рекомендованими монетами.

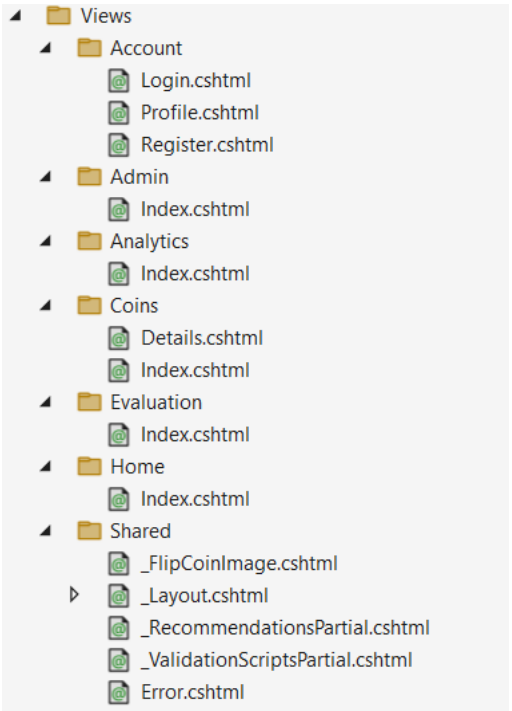

Рисунок 3.26 – Елементи модулю View

У папці Analytics розміщено представлення Index, яке відповідає за візуалізацію аналітичних даних щодо прогнозованої вартості конкретної монети. Тут реалізовано таблицю з числовими прогнозами та графік,

створений за допомогою Chart.js, де динаміка ціни відображається на визначених часових інтервалах.

Сторінка з оцінкою зображення (Evaluation/Index) надає користувачу інтерфейс для вибору монети та завантаження фотографій аверсу та реверсу. Після відправки зображень система здійснює класифікацію їх стану, а результат показується на тій же сторінці.

Окреме представлення Admin/Index забезпечує доступ до запуску парсерів – інтерфейс мінімалістичний, але функціональний: одна кнопка ініціює запит до сервера на запуск всіх фонових скриптів.

Домашня сторінка (Home/Index) виконує роль вітрини – тут представлені популярні монети, а також окремий блок з акційними пропозиціями. Якщо користувач авторизований, додатково з’являється розділ персональних рекомендацій. Всі спільні елементи (хедер, футер, стилі навігації, відображення сповіщень) реалізовані за допомогою окремих компонентів, таких як _Layout.cshtml, _RecommendationsPartial.cshtml тощо.

4 ДОСЛІДЖЕННЯ РОЗРОБЛЕНОГО ВЕБДОДАТКУ

4.1 Огляд інтерфейсу додатка

4.1.1 Головна сторінка

Головна сторінка вебдодатку (рисунок Б.1 в додатку Б) виконує ознайомчу функцію і є початковою точкою для користувача. У верхній частині розташована навігаційна панель, що дає змогу швидко переміщатися до основних розділів: каталогу, оцінок, особистого кабінету та пошукового рядка.

Основний простір заповнюють блоки з динамічним контентом – «Популярні товари» і «Пропозиції зі знижками», де показуються товари, відібрані на основі попередніх дій користувача та аналізу цін. Кожен товар містить короткий опис, зображення, вартість та кнопку для додавання до улюблених.

4.1.2 Каталог

Сторінка каталогу монет – важлива складова частина інтерфейсу користувача. Її розроблено з підтримкою різних фільтрів, які дозволяють звузити результати пошуку відповідно до декількох критеріїв: вартість, рік випуску, країна-емітент, номінал, тип металу та валюта. Для візуального представлення ціни використовується повзунок, який автоматично налаштовується відповідно до поточного діапазону результатів. З правого боку відображаються картки монет з ключовою інформацією. Сам фільтраційний блок переміщається паралельно скролингу сторінки. Реалізовано також сортування за ціною та назвою.

Кожна картка містить найменування, рік випуску, зображення аверсу, а також поточну ціну. Додатково реалізовано можливість додати монету до

обраного списку. Перелік результатів можна сортувати за заданими параметрами. Інтерфейс сторінки каталогу можна побачити в додатку Б (рисунок Б.2), а приклад використання користувацького фільтрування на рисунку Б.3.

4.1.3 Сторінка монети

Сторінка огляду монети надає доступ до її властивостей: зображення, найменування, рік карбування, сплав, вага, діаметр, номінал, країна випуску та тираж. У верхній частині розміщено повне ім’я монети, поточна її вартість та кнопка для збереження у списку вподобаних. Окремо виділено графічні зображення аверсу та реверсу, щоб дати користувачу уявлення про монету.

Під заголовком «Динаміка ціни» користувач може спостерігати графік коливання вартості у вибраній валюті. Доступний перегляд у гривні, доларі та євро. Якщо актуальних пропозицій в базі даних не виявлено, з’являється відповідне сповіщення. Внизу сторінки знаходиться блок «З цим часто переглядають», який реалізує рекомендаційний алгоритм та допомагає користувачу відшукати схожі монети на основі історії взаємодії. Візуальне представлення сторінки знаходиться на рисунку Б.4 у додатку Б.

4.1.4 Прогнозування ціни

Аналітична сторінка висвітлює прогноз майбутньої вартості монети, згенерований на основі моделі машинного навчання. Вона відкривається як окрема сторінка після переходу з картки монети, надаючи узагальнену інформацію щодо очікуваної зміни вартості у вигляді таблиці та графіка. Прогноз розповсюджується на інтервали часу – 7, 30, 180 та 365 днів, демонструючи прогнозовану ціну, абсолютне відхилення, відсоткову зміну та опис тренду (відносно поточної ціни).

Під таблицею знаходиться графік коливання ціни, що ілюструє майбутні ціни у розрізі часу та дозволяє швидко оцінити характер руху ціни, включаючи коливання, спади або стабільність. Також присутня можливість для повернення до сторінки монети. Інтерфейс наведений у додатку Б на рисунку Б.5.

4.1.5 Оцінка стану

На сторінці оцінки зображень створено інтерфейс для завантаження фотографій монет з метою їх подальшої автоматизованої класифікації стану. Користувачу пропонується вибрати країну походження монети, її назву та рік фіксації, після чого він завантажує два зображення: аверс і реверс. Кнопка «Оцінити» запускає процес обробки, результат якого користувач бачить на екрані (додаток Б, рисунок Б.6).

Після того, як зображення відправлені, модель машинного навчання, з якою взаємодіє сервер, здійснює прогноз категорії якості монети. Результат, супроводжуваний відповідною ймовірністю, відображається у текстовому форматі, а також візуально демонструються завантажені фотографії для підтвердження (додаток Б, рисунок Б.7).

4.1.6 Адміністраційна панель

Сторінка адміністратора виконує одну основну функцію – запуск усіх парсерів вручну (додаток Б, рисунок Б.8). Вона доступна лише користувачам з роллю адміністратора, що гарантує обмежений доступ до критичного функціоналу. Натискаючи на кнопку, адміністратор ініціює фоновий запуск скриптів збору даних, необхідних для оновлення каталогу, прогнозів і аналітики.

4.1.7 Автентифікація та авторизація

Сторінки автентифікації та авторизації надають користувачам можливість входу до системи, проходження реєстрації, а також перегляду даних свого профілю (див. додаток Б, рисунки Б.9–Б.11). Їх було реалізовано з використанням cookie-автентифікації, де ключова інформація зберігається у зашифрованому форматі у сесії браузера.

Сторінка реєстрації містить звичайну форму для введення імені користувача, електронної адреси та пароля. У разі коректного заповнення полів створюється новий запис у базі даних, а користувач отримує сповіщення про успішну реєстрацію. Після цього він може виконати вхід на відповідній сторінці, яка перевіряє відповідність адреси електронної пошти та пароля. Якщо дані вірні – користувачеві надається сесія, і його перенаправляють до особистого профілю. В профілі, окрім особистої інформації, присутні кнопки для перегляду вибраного контенту та історії дій, а також опція виходу із системи.

4.2 Загальний аналіз роботи вебдодатку

Під час розробки вебдодатку MintIQ особлива увага зосереджувалася на стабільності, швидкості та логічній організації частин системи. Кожен компонент, від шару представлення до машинного навчання, було втілено у вигляді незалежних модулів. Загальна взаємодія побудована за звичною архітектурою MVC: контролери управляють перенаправленням запитів, представлення відповідають за відображення сторінок, а бізнес-логіка прихована у сервісах.

Система демонструє надійну роботу навіть при великій кількості запитів до бази даних, зокрема при пошуку, сортуванні, роботі з фільтрами або виклику моделей прогнозування. Усі запити до бази оптимізовано за

допомогою методів Include, Select, AsNoTracking та попереднього агрегування даних.

Окремо варто відзначити модуль MLService, який ефективно взаємодіє з Python-скриптами для обробки зображень та прогнозів. Обчислення, що вимагають значних обчислювальних ресурсів, виконуються асинхронно, не блокуючи роботу інтерфейсу.

Автентифікація користувачів базується на механізмі CookieAuthentication, що дає змогу ефективно керувати сесіями та обмеженням доступу. Відповідно до ролі, користувачі бачать лише ті сторінки, які їм дозволено. Уся система підтримує зберігання історії переглядів, улюбленого контенту та персоналізованих рекомендацій, що дає можливість персоналізувати взаємодію з платформою, а також використовувати накопичені дані про поведінку для поліпшення точності прогнозів та якості рекомендацій.

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи було розроблено та впроваджено вебдодаток, призначений для вивчення, аналізу та експертної оцінки нумізматичної продукції. Функціональність системи включає в себе розширений каталог монет, засоби фільтрації, огляд інформації, прогнозування ціни з використанням машинного навчання, а також персоналізовані рекомендації на основі історії взаємодії користувачів.

Технічною платформою додатку виступає ASP.NET Core в комбінації з СУБД PostgreSQL та модулями машинного навчання, реалізованими на Python і ML.NET. Реалізовано чіткий поділ прав доступу для різних категорій користувачів: гостя, зареєстрованого користувача та адміністратора. Інформація автоматично оновлюється з використанням парсерів, які збирають дані з відкритих джерел. Особливу увагу приділено системі оцінки стану монет на основі зображень, в якій використовується модель класифікації, розроблена на основі нейронної мережі.

Представлена робота демонструє приклад комплексної інтеграції веб-технологій, систем обробки даних та методів машинного навчання в єдиній системі. Під час розробки враховано принципи масштабованості, розширюваності та гнучкості, що дає проєкту потенціал для подальшого вдосконалення.

Отримані результати можуть служити основою для проведення поглиблених досліджень у сфері нумізматики, вебаналітики або інтелектуальних систем підтримки прийняття рішень. З практичної точки зору, система може бути адаптована для використання у комерційних цілях задля полегшення в отриманні інформації про монети як унікального екземпляра.

Отже, поставлені завдання було повністю виконано, а результати роботи підкреслюють її практичну та навчальну цінність у контексті університетської освітньої програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Нумізматичний циркуляр. *Національна історична бібліотека України*. URL: https://nibu.kyiv.ua/exhibitions/365/ (дата звернення: 12.05.2025).

2. Пасічник Н., Ріжняк Р. Еволюція оцінки колекційної вартості монет (друга половина XIX – кінець XX століття). *Наукові записки з української історії. Збірник наукових статей.* Вип. 33. Переяслав-Хмельницький, 2013. С. 175–180

3. Інтернет-магазин нумізматичної продукції Національного банку України. *bank.gov.ua*. URL: https://coins.bank.gov.ua (дата звернення: 12.05.2025).

4. Violity.com Маркет. *Violity*. URL: https://violity.com/ua/ (дата звернення: 12.05.2025).

5. Hotline – порівняти ціни в інтернет-магазинах України. *Hotline.ua*. URL: https://hotline.ua (дата звернення: 14.05.2025).

6. Melnik M., Alm J. Seller Reputation, Information Signals, and Prices for Heterogeneous Coins on eBay. *Southern Economic Journal*. 2005. Vol. 72, no. 2. P. 305–328.

7. Coin Grading | Numismatic Guaranty Company | NGC. *NGC*. URL: https://www.ngccoin.com (дата звернення: 15.05.2025).

8. ASP.NET documentation. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/uk-ua/aspnet/core/?view=aspnetcore-9.0 (дата звернення: 15.05.2025).

9. Overview of ASP.NET Core MVC. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/uk-ua/aspnet/core/mvc/overview?view=aspnetcore-9.0 (дата звернення: 16.05.2025).

10. PostgreSQL: Documentation. *PostgreSQL: The world's most advanced open source database*. URL: https://www.postgresql.org/docs/ (дата звернення: 17.05.2025).

11. Overview of Entity Framework Core - EF Core. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/uk-ua/ef/core/ (дата звернення: 17.05.2025).

12. Razor syntax reference for ASP.NET Core. *Microsoft Learn: Build skills that open doors in your career*. URL: https://learn.microsoft.com/uk-ua/aspnet/core/mvc/views/razor?view=aspnetcore-9.0 (дата звернення: 18.05.2025).

13. Introduction. *Bootstrap | The most popular HTML, CSS, and JS library in the world.* URL: https://getbootstrap.com/docs/5.0/getting-started/introduction/ (дата звернення: 19.05.2025).

14. Qian S., Ning C., Hu Y. MobileNetV3 for Image Classification. *2021 IEEE 2nd International Conference on Big Data, Artificial Intelligence and Internet of Things Engineering (ICBAIE)*. 2021. P. 490–497.

15. How to use ONNX Runtime. *ONNXRuntime*. URL: https://onnxruntime.ai/docs/ (дата звернення: 23.05.2025).

16. Features – LightGBM 4.6.0.99 documentation. *LightGBM 4.6.0 documentation*. URL: https://lightgbm.readthedocs.io/en/latest/Features.html (дата звернення: 24.05.2025).

17. Sagarra M., Vici L., Zanola R. Returns from rare coins: a machine learning approach. *Journal of Cultural Economics*. 2025. URL: https://doi.org/10.1007/s10824-025-09530-8 (дата звернення: 26.05.2025).

18. Easton S. Valuing coins as the sum of the underlying asset and a perpetual American put option. *Global Finance Journal*. 2007. Т. 17, № 3. С. 397–402. URL: https://doi.org/10.1016/j.gfj.2006.08.001 (дата звернення: 24.05.2025).

19. W3Schools.com. *W3Schools Online Web Tutorials | JavaScript*. URL: https://www.w3schools.com/js/ (дата звернення: 25.05.2024).

20. W3Schools.com. *W3Schools Online Web Tutorials | CSS*. URL: https://www.w3schools.com/css/ (дата звернення: 26.05.2024).