Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

«РОЗРОБКА МЕТОДІВ І ЗАСОБІВ СИСТЕМИ УПРАВЛІННЯ ОБІГОМ АНТИКВАРІАТУ»

Виконав: студент II курсу
групи 2ПІ-22м спеціальності
121 – Інженерія програмного забезпечення
Сафо В.В.

Керівник: д. т. н., проф. каф. ПЗ
Ліщинська Л.Б.
«08» грудня 2023р.

Опонент: к.т.н., доц. каф. ЗІ
Дудатьєв А. В.
«08» грудня 2023р.

Допущено до захисту
Завідувач кафедри ПЗ
д.т.н., проф. Романюк О. Н.
(прізвище та ініціали)
«08» грудня 2023 р.

ВНТУ – 2023

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Рівень вищої освіти ІІ-й (магістерський)
Галузь знань 12 – Інформаційні технології
Спеціальність 121 – Інженерія програмного забезпечення
Освітньо-професійна програма – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
«19» вересня 2023 р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Сафо Віктору Володимировичу

1. Тема роботи – розробка методів і засобів системи управління обігом антикваріату.

Керівник роботи: Ліщинська Людмила Броніславівна, д.т.н., професор кафедри ПЗ, затверджені наказом вищого навчального закладу від «18» вересня 2023 р. № 247.

2. Строк подання студентом роботи
5 грудня 2023 р.

3. Вихідні дані до роботи:
Платформа: Microsoft Azure, .NET.
Мова програмування: C#.
Технології: Microservices, Azure Functions, REST API, GraphQL API, SQL, SQL, Cosmos DB, Azure Service Bus, Azure Blob Storage, Redis.

4. Зміст текстової частини: предметна область програмного забезпеченняб методи і засоби проектування програмного забезпечення, реалізація програми створення програмного забезпечення, тестування програмного додатку, економічна частина, висновки, додатки.

5. Перелік ілюстративного матеріалу: мета, об'єкт, предмет, основні задачі, новизна отриманих результатів, аналоги, технологічні вимоги, архітектура системи, діаграма пакетів, діаграма розгортання, архітектура системи аналізу даних, блок-схема алгоритму для пошуку оптимальних товарів, прототип системи, результат роботи.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Ліщинська Л. Б., д.т.н., професор кафедри ПЗ	19.09.2023	05.12.2023
5	Кавецький В. В., к.е.н., доц. каф. ЕПВМ	19.09.2023	18.11.2023

7. Дата видачі завдання 19 вересня 2023 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасного стану питання та обґрунтування задачі	20.09.2023-6.09.2023	Вик.
2	Розробка архітектури програмного додатку	6.09.2023-30.09.2023	Вик.
3	Проектування бази даних	1.10.2023 – 7.10.2023	Вик.
4	Обґрунтування вибору мови програмування та середовища розробки додатку	7.10.2023 – 10.10.2023	Вик.
5.	Розробка загального алгоритму роботи програмного додатку	10.10.2023 – 1.10.2023	Вик.
6.	Розробка графічного інтерфейсу	1.11.2023 – 10.11.2023	Вик.
7.	Економічна частина	10.11.2023-13.11.2023	Вик.
8.	Оформлення матеріалів магістерської кваліфікаційної роботи	13.11.2023 – 01.12.2023	Вик.

Студент _____ **Сафо В. В.**
(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ **Ліщинська Л.Б.**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.5

Сафо В. В. Розробка методів і засобів системи управління обігом антикваріату. Магістерська кваліфікаційна робота зі спеціальності 121 – Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2023. 100 с. На укр. мові. Бібліогр.: 34 назв; рис.: 29; табл. 15.

Сучасний ринок антикварних товарів стає все більше конкурентним, вимагаючи від бізнесу ефективних рішень у сфері управління обігом цих цінних предметів. Магістерська кваліфікаційна робота спрямована на розробку системи управління обігом антикваріату, яка використовує мікросервісну архітектуру, .NET платформу та нейронні мережі для ефективного пошуку клієнтських потреб. Ця робота пропонує комплексний підхід до управління антикварними товарами, забезпечуючи швидкий і точний пошук товарів для клієнтів за допомогою нейронних мереж. Мікросервісна архітектура дозволяє розділити систему на невеликі незалежні сервіси, що полегшує розвиток і підтримку системи.

В рамках цієї роботи було проаналізовано сучасного стану ринку антикварних товарів та існуючих систем управління обігом, розроблено мікросервісну архітектуру для системи управління обігом антикваріату, використано .NET платформу для реалізації різних компонентів системи, впроваджено нейронні мережі для аналізу клієнтських потреб та рекомендацій.

Результати цієї дослідницької роботи можуть бути корисні для підприємств, які займаються торгівлею антикваріатом та бажають покращити свою конкурентоспроможність на ринку.

Ключові слова: система управляння, антикваріат, мікро сервісна архітектура.

ABSTRACT

Safo V. V. Development of methods and means of the system for managing the circulation of antiques. Master's thesis on the specialty 121 - Software engineering, educational program - software engineering. Vinnytsia: VNTU, 2023. 100 p. In Ukrainian language. Bibliography: 34 titles; Fig.: 29; table 15.

The modern market of antique goods is becoming more and more competitive, demanding from businesses effective solutions in the field of managing the circulation of these valuable items. The master's qualification work is aimed at the development of an antiquities circulation management system that uses microservice architecture, the .NET platform and neural networks to efficiently search for customer needs. This work offers a comprehensive approach to managing antique goods, providing fast and accurate product searches for customers using neural networks. Microservice architecture allows you to divide the system into small independent services, which facilitates the development and maintenance of the system.

As part of this work, the current state of the market of antique goods and existing circulation management systems was analyzed, a microservice architecture was developed for the antiquities circulation management system, the .NET platform was used to implement various components of the system, and neural networks were implemented to analyze client needs and recommendations.

The results of this research work can be useful for enterprises that are engaged in the trade of antiques and wish to improve their competitiveness in the market.

Keywords: control system, antiques, micro service architecture.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний бізнес у сфері торгівлі стикається з ростом конкуренції та постійно зростаючими вимогами клієнтів. Для успішної роботи систем управління обігом антикваріату важливо не лише мати широкий асортимент цінних предметів, але й забезпечити ефективне управління товарообігом та високу якість обслуговування клієнтів.. Дослідження ринку та оптимізація роботи системи можуть допомогти виявити конкурентні переваги, вдосконалити пропозицію товарів та послуг, залучити більше клієнтів та підвищити ефективність бізнесу. Існують декілька методів які використовуються для побудови таких систем, таких як метод побудови системи на основі мікросервісної архітектури, методи пошуку оптимального шляху за допомогою алгоритмів маршрутизації та геолокації, методи пошуку знижок і акцій відповідно до вимог клієнта. Відповідні аналоги не використовують вищевказані методи або використовують не в повному обсязі. Тому є доцільним розробити нову систему з використанням вищевказаних методів. Також є доцільним вдосконалити метод побудови архітектури системи, так як технології розвиваються, потреби і запити клієнтів зростають що спонукає в вдосконаленні продуктивності і ефективності системи [3,4].

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася відповідно до плану науково-дослідних робіт кафедри програмного забезпечення.

Мета та завдання дослідження. Метою дослідження підвищення ефективності системи для управління обігом антикваріату.

Основними задачами дослідження є:

- проаналізувати предметну область поставленої задачі, існуючі аналоги системи та поставити задачі, які треба виконати;
- розробити модель архітектури
- модель архітектури системи для управління обігом антикваріату

- провести архітектурне і детальне проектування системи з запропонованими новими підходами та рішеннями:
  - проектування системи на базі мікросервісної архітектури та новітніх підходів і сервісів;
  - метод проектування системи аналізу даних використовуючи новітні підходи і сервіси;
  - проектування методу оптимального пошуку з використанням нейронних мереж.
- провести дослідження у виборі ресурсів для розробки і провести розробку прототипу програмного забезпечення.
- провести тестування прототипу розробленої системи.
- проаналізовати конкурентоспроможність програмного продукту і здійснити розрахунки ефективності вкладень.

Об’єктом дослідження є процеси управляння обігом антикваріату.

Предметом дослідження є методи та засоби управляння обігом антикваріату.

Методи дослідження. Можна виділити Інтернет ресурси, сучасні методи розробки і вдосконалення що використовуються для розробки програмних засобів сучасних потреб. А саме методи побудови системи з використанням мікросервісної архітектури використовувались як основа для побудови архітектури системи, методи пошуку оптимального шляху, алгоритми машинного навчання, методи глибокого навчання, інструменти аналітики і бізнес-інтелекту для аналізу результатів рекомендацій та персональних пропозицій, дослідження теми і методів розробки з використанням нейронних мереж використовувались як основа для побудови системи аналізу даних і як результат побудови моделі оптимального пошуку.

Наукова новизна отриманих результатів. Як новизна отриманих результатів можна виділити наступне:

- удосконалено модель побудови системи з використанням мікросервісної архітектури, з використанням асинхронного спілкування між

сервісами за допомогою повідомлень і орхестрацію для контролю над сервісами, що дозволило підвищить продуктивність і ефективність системи в цілому як у використанні так і у підтримці;

- удосконалено модель побудови системи аналізу даних за рахунок агрегації даних для подальшого використання. Що дозволило ефективніше формувати дані для подальшого аналізу і покращить продуктивність побудови відповідних звітів;
- удосконалено метод пошуку оптимального шляху з використанням нейронних мереж, що дозволить зробити пошук оптимального шляху доставки товару більш ефективним.

Практична цінність отриманих результатів. Практична цінність одержаних результатів полягає в тому, що на основі отриманих в магістерській кваліфікаційній роботі теоретичних положень запропоновано методи та розроблено програмні засоби системи управління обігом антикваріату.

Особистий внесок здобувача. Усі наукові результати, викладені у магістерській кваліфікаційній роботі, отримані автором особисто. У друкованих працях, опублікованих у співавторстві, автору належать такі результати: архітектура системи в цілому; архітектура системи аналізу даних; алгоритм пошуку з використанням нейронних мереж; прототип системи для управління обігом антикваріату [1,2].

Апробація матеріалів магістерської кваліфікаційної роботи. Основні положення магістерської кваліфікаційної роботи доповідалися та обговорювалися на конференціях: міжнародна науково-практична інтернет-конференція «Молодь в науці: дослідження, проблеми, перспективи» (Вінниця, 2023), Всеукраїнська науково-практичної Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (Вінниця, 2023).

Публікації. Деякі результати дослідження опубліковані в 2 наукових працях у тому числі на міжнародна науково-практична інтернет-конференція та Всеукраїнської науково-практичної Інтернет-конференції [1,2].

1 АНАЛІЗ МЕТОДІВ І ПРОГРАМНИХ ЗАСОБІВ ДЛЯ СТВОРЕННЯ СИСТЕМИ УПРАВЛІННЯ ОБІГОМ АНТИКВАРІАТУ

1.1 Галузь застосування, призначення і вимоги до створення програмного забезпечення

Програмне забезпечення для магазину антикваріату може використовуватися для автоматизації різних аспектів діяльності магазину, забезпечення ефективної роботи та поліпшення обслуговування клієнтів. Основні області застосування та призначення програмного забезпечення магазину антикваріату включають:

- Управління інвентарем: Програмне забезпечення може допомогти веденню обліку товарів, їх категоризації та опису, включаючи фотографії, історію походження та іншу релевантну інформацію. Це допомагає контролювати наявність товарів, уникати подвійних продажів і забезпечувати точність інформації про товари;
- Продаж і обробка замовлень: Програмне забезпечення може забезпечити зручний процес продажу товарів, включаючи оформлення замовлень, виписку рахунків, оплату та стеження за станом замовлення. Воно може також автоматично розраховувати податки, знижки та вартість доставки;
- Клієнтський облік: Програмне забезпечення може зберігати інформацію про клієнтів, включаючи контактні дані, історію покупок і замовлень. Це допомагає покращити обслуговування клієнтів, розуміючи їхній вибір, нагадуючи про спеціальні пропозиції та забезпечуючи персоналізований підхід;
- Управління цінами і знижками: Програмне забезпечення може допомагати встановлювати ціни на товари, здійснювати розрахунки маржі, керувати акціями та знижками. Воно також може надавати звіти про прибутковість та ефективність різних товарів і категорій;

- Інтелектуальний облік клієнтських потреб: Збереження історії покупок, переглядів клієнта. На базі зібраної інформації формування списку потреб, реклами, знижок;
- Онлайн-продажі і електронна комерція: Програмне забезпечення може інтегруватися з онлайн-платформами або допомагати створювати власний веб-магазин для продажу антикварних товарів через Інтернет. Це розширює географію магазину і забезпечує доступ клієнтів до товарів та послуг у будь-який час.

Вимоги до створення програмного забезпечення магазину антикваріату можуть включати:

- Інтерфейс користувача: Зручний і простий у використанні інтерфейс, який дозволяє персоналу магазину ефективно працювати з програмою, швидко знаходити інформацію про товари та обробляти замовлення;
- Інтеграція з обладнанням: Програмне забезпечення повинно бути сумісним з різними типами обладнання, такими як касові апарати, сканери штрих-кодів, принтери для друку чеків та інше;
- Безпека даних: Важливо забезпечити захист персональних даних клієнтів, включаючи інформацію про платежі та історію покупок. Програмне забезпечення повинно мати механізми шифрування даних та забезпечувати обмежений доступ до конфіденційної інформації;
- Звіти та аналітика: Програмне забезпечення повинно забезпечувати можливість створення звітів та аналітики щодо продажів, запасів, прибутковості та інших ключових показників діяльності магазину. Це допоможе управлінню приймати обґрунтовані рішення та розуміти ефективність бізнесу;
- Масштабованість: Програмне забезпечення повинно бути здатне масштабуватися і працювати з ростом бізнесу. Воно повинно підтримувати розширення кількості товарів, клієнтів та магазинів, якщо такі є.

Урахування цих вимог допоможе створити програмне забезпечення, яке

задовольнить потреби магазину антикваріату і сприятиме його успішній діяльності.

1.2 Аналіз методі і засобів для побудови системи для управління обігом антикваріату

Побудова системи для управління обігом антикваріату за допомогою мікросервісної архітектури може бути важливим завданням, яке вимагає аналізу існуючих методів і засобів. Давайте розглянемо кілька ключових аспектів, які потрібно врахувати під час створення такої системи:

- Аналіз бізнес-процесів: Перш за все, потрібно ретельно проаналізувати бізнес-процеси, пов'язані з управлінням антикваріатом. Це включає в себе приймання товарів, зберігання, продаж, оцінку та інші аспекти. Важливо визначити всі етапи цих процесів.
- Вибір технологій: Для створення мікросервісної системи важливо вибрати технології, які відповідають потребам вашого проекту. Це може включати в себе мови програмування, бази даних, фреймворки та інші інструменти.
- Розробка мікросервісної архітектури: Мікросервіси - це невеликі, незалежні компоненти програмної системи, які взаємодіють між собою за допомогою API. Вони можуть бути використані для реалізації окремих функцій, таких як каталог товарів, керування замовленнями, облік клієнтів тощо.
- Керування даними: Антикваріат може вимагати обробки великої кількості історичних даних. Для цього вам потрібно буде вибрати інструменти для збереження, оновлення та доступу до даних. Реляційні та NoSQL бази даних можуть бути корисними в цьому відношенні.
- Безпека: Забезпечення безпеки даних та транзакцій - це надзвичайно важлива частина будь-якої системи управління обігом

антикваріату. Розгляньте використання автентифікації, авторизації, шифрування та інших методів для захисту інформації.

- Інтеграція з іншими системами: Якщо у вас є інші системи, такі як онлайн-магазини або системи управління запасами, важливо забезпечити можливість їх інтеграції з вашою системою для уникнення дублювання даних та оптимізації робочих процесів.
- Моніторинг і управління: Після впровадження системи важливо надавати можливість моніторингу та управління мікросервісами. Використовуйте інструменти моніторингу та логування для виявлення та вирішення проблем.
- Тестування та впровадження: Перш ніж вводити систему в експлуатацію, необхідно провести тестування на різних етапах розробки. Тестування включає функціональне, інтеграційне, навантажувальне і security тестування.
- Підтримка і розвиток: Після впровадження системи вирішіть питання підтримки та подальшого розвитку. Система має бути гнучкою і готовою до внесення змін з плином часу.
- Забезпечення дотримання законодавства: Враховуйте вимоги законодавства щодо обігу антикваріату, зокрема стосовно податків, ліцензій та інших правових аспектів.

Для побудови мікросервісної системи для управління обігом антикваріату може бути корисним вивчення існуючих практик у сфері розробки мікросервісів, а також розгляд та порівняння різних технологій та рішень, що вже використовуються в схожих проектах.

Існують ряд практик у сфері розробки мікросервісів, які допомагають забезпечити успішну і ефективну розробку та експлуатацію мікросервісних систем. Ось декілька з них:

- Розбиття на дрібні сервіси: Основною ідеєю мікросервісної архітектури є розділення функціональності системи на невеликі, самостійні

сервіси. Кожен сервіс повинен виконувати обмежену кількість функцій та бути незалежним від інших сервісів.

- Використання API: Мікросервіси повинні взаємодіяти один з одним та з іншими системами через добре визначені API. REST або GraphQL часто використовуються для цієї мети.
- Контейнеризація: Використання контейнерів, таких як Docker, допомагає забезпечити консистентність та переносимість середовищ виконання для мікросервісів.
- Оркестрація: Використання оркестраторів, таких як Kubernetes, дозволяє автоматизувати розгортання, масштабування та керування мікросервісами.
- Моніторинг і логування: Збір і аналіз логів та метрик допомагають виявляти проблеми та вдосконалювати продуктивність мікросервісів. Інструменти, такі як Prometheus, Grafana, ELK Stack, допомагають у цьому.
- Автоматизоване розгортання: Використання процесів автоматизованого розгортання (наприклад, Continuous Integration/Continuous Deployment - CI/CD) спрощує впровадження змін у мікросервісах та забезпечує надійність та швидкість розгортання.
- Централізований конфігураційний управління: Використання централізованих рішень для керування конфігураціями допомагає забезпечити консистентність параметрів та налаштувань мікросервісів.
- Тестування: Тестування кожного мікросервісу окремо та на рівні системи допомагає виявляти помилки та забезпечувати високу якість продукту.
- Документація: Зберігайте докладну документацію для кожного мікросервісу, включаючи API, параметри конфігурації та вимоги до інтеграції.
- Моніторинг виробництва: Після впровадження системи, використовуйте моніторинг виробництва для слідкування за продуктивністю, доступністю та безпекою мікросервісів.

Реалізація пошуку оптимального шляху доставки є важливим завданням

для багатьох логістичних та транспортних систем. Існують різні методи та практики, які можна використовувати для досягнення цієї мети. Ось декілька з них:

- Алгоритми маршрутизації: Використовуйте алгоритми маршрутизації, такі як алгоритми Дейкстри або алгоритми A* для знаходження найкоротшого шляху між точками. Розгляньте використання графів доріг або інших структур даних для представлення мережі шляхів.
- Геолокація та GPS: Використовуйте GPS та геолокаційні дані для визначення поточного місцезнаходження автомобілів або доставки та відстеження їх руху в реальному часі.

Прогноз популяції та трафіку:

- Враховуйте дані щодо популяції та трафіку, щоб передбачити оптимальний час доставки та уникнути пікових годин руху.

Оптимізація маршруту:

- Використовуйте програми для оптимізації маршруту, які дозволяють враховувати обмеження, такі як обмеження на вагу, об'єм, час, та інші фактори. Такі програми можуть автоматично обирати оптимальний маршрут для кожного замовлення доставки.
- Розглядайте альтернативні маршрути: Розгляньте можливість вибору альтернативних маршрутів, які можуть бути швидшими або дешевшими, в залежності від конкретних умов.
- Додаткові сервіси: Розгляньте можливість використання додаткових сервісів, таких як навігація в режимі реального часу, інтеграція з системами трафіку та інші інструменти для поліпшення доставки.
- Аналітика та відстеження: Використовуйте аналітичні інструменти для відстеження та аналізу даних про доставку. Це допоможе виявити патерни та можливості для оптимізації.
- Реакція на зміни: Забезпечте систему, яка може адаптуватися до непередбачуваних змін, таких як затори, зміни в розкладі, погодні умови та інші фактори, які можуть вплинути на доставку.

- Інтеграція з ГІС: Використовуйте геоінформаційні системи (ГІС) для відображення дорожньої мережі, трафіку та інших географічних даних, що може бути корисним для пошуку оптимальних шляхів.
- Оцінка витрат та вибір рішень: Розгляньте витрати та користь від різних маршрутів та стратегій доставки для прийняття оптимального рішення.

Реалізація пошуку оптимальних товарів та скидок для клієнтів в електронних магазинах та різних електронних платформах є важливою для залучення та утримання клієнтів. Існують різні методи та практики, які можна використовувати для цього. Ось декілька з них:

- Персоналізований підхід: Використовуйте дані про клієнта, такі як попередні покупки, браузерна історія та профіль користувача, для надання індивідуальних рекомендацій та скидок. Можна використовувати алгоритми рекомендацій, такі як колаборативне фільтрування та контент-фільтрування.
- Аналіз даних: Використовуйте аналіз даних та машинне навчання для виявлення патернів та тенденцій в покупках клієнтів. Наприклад, можна аналізувати динаміку продажів та використовувати цю інформацію для створення пропозицій.
- A/B тестування: Використовуйте A/B тестування для оцінки ефективності різних пропозицій, скидок та акцій. Це допомагає визначити, які пропозиції найбільше привертають увагу клієнтів.
- Динамічне ціноутворення: Використовуйте динамічне ціноутворення, яке змінює ціни в реальному часі в залежності від попиту, запасів та інших факторів. Це може стимулювати клієнтів придбати товари та послуги.
- Купони та промо-коди: Надавати клієнтам знижки та пропозиції через купони та промо-коди. Це може бути зручним інструментом для залучення нових клієнтів та стимулювання повторних покупок.
- Групові знижки: Пропонуйте знижки на товари, які купуються великими партіями або разом з іншими товарами. Це може стимулювати більші обсяги продажів.

- Контент-маркетинг: Використовуйте цікавий та корисний контент, такий як блоги, відео та інші ресурси, для привертання уваги та надання інформації про продукти та акції.
- Програми лояльності: Створюйте програми лояльності, які нагороджують клієнтів за покупки та участь у різних акціях. Це допомагає залучати та утримувати постійних клієнтів.
- Використання інструментів електронної комерції: Використовуйте спеціальні платформи для електронної комерції та платіжних систем для зручного та ефективного впровадження різних акцій та скидок.
- Збільшення споживчої цінності: Враховуйте, які додаткові послуги або товари можуть збільшити споживчу цінність для клієнтів та робити вашу пропозицію більш привабливою.

Сучасні системи для пошуку оптимальних товарів і скидок для клієнтів використовують різні технології та підходи для надання персоналізованих рекомендацій та підвищення задоволення клієнтів. Ось декілька сучасних систем та технологій, які використовуються для цих цілей:

- Рекомендаційні системи: Рекомендаційні системи використовують алгоритми машинного навчання, щоб пропонувати клієнтам товари та скидки, які відповідають їхнім інтересам та попереднім покупкам. Такі системи можуть аналізувати десятки або сотні факторів, включаючи рейтинги, додані до кошика товари, історію переглядів тощо.
- Динамічне ціноутворення: Електронні магазини та торгові платформи використовують динамічне ціноутворення для зміни цін в реальному часі в залежності від різних факторів, таких як попит, запаси та конкуренція. Це дозволяє надавати клієнтам знижки та пропозиції, які привертають їхню увагу.
- Використання штучного інтелекту: Штучний інтелект (ШІ) використовується для аналізу даних та прогнозування покупок клієнтів. ШІ може виявляти патерни та встановлювати зв'язки між різними товари та

покупками, що допомагає пропонувати більш цікаві та контекстуальні пропозиції.

- Глибоке навчання: Глибоке навчання використовує нейронні мережі для аналізу та моделювання клієнтських поведінок. Воно може допомагати покращити точність рекомендацій та прогнозувати майбутні покупки.
- Аналітика та бізнес-інтелект: Використовуйте інструменти аналітики та бізнес-інтелекту, щоб аналізувати результати рекомендацій та ефективність акцій. Це допомагає розуміти, які стратегії найкраще працюють і які потребують оптимізації.
- Інтерактивні інтерфейси: Розроблюють інтерактивні інтерфейси для користувачів, які дозволяють клієнтам налаштовувати свої персоналізовані пропозиції та фільтрувати товари та скидки відповідно до своїх потреб.
- Мобільні додатки та платформи: Мобільні додатки і онлайн-платформи розроблюються з урахуванням зручності користувача та можливостей рекомендацій для забезпечення зручного та ефективного пошуку товарів та скидок.
- Аналіз відгуків та відгуків клієнтів: Використовуйте аналіз текстових відгуків та відгуків клієнтів для визначення їхніх смаків та побажань. Це може допомогти в покращенні рекомендацій та відображенні релевантних товарів.

1.3 Порівняльний аналіз аналогів програмних засобів для систем управління обігом антикваріату

Існує кілька аналогів програмного забезпечення, які можуть бути використані як магазин антикваріату. Ось декілька популярних аналогів:

- OLX є однією з найбільших і популярних онлайн-платформ у світі для розміщення оголошень про продаж різноманітних товарів, включаючи

антикваріат. OLX працює в багатьох країнах, включаючи Україну, і забезпечує можливість встановлення контакту між продавцями та покупцями безпосередньо;

- Violity є онлайн-платформою, спеціалізованою на продажу антикваріату, предметів колекціонування та різноманітних товарів вторинного ринку в Україні. Ця платформа дозволяє користувачам розміщувати оголошення про продаж антикварних товарів, встановлювати ціни та взаємодіяти з потенційними покупцями;
- eBay є одним з найбільших і найпопулярніших онлайн-майданчиків для купівлі та продажу антикваріату. Він має велику кількість лотів з різних категорій, включаючи антикварні предмети, мистецтво, колекціонерські речі тощо;
- Catawiki є онлайн-аукціонною платформою, де можна знайти антикварні та колекційні предмети. Покупці можуть придбати товари на аукціоні, який організовується для різних категорій, включаючи антикваріат, мистецтво, монети, листівки та інше;

Багато простих online-сайтів які використовуються для продажу різних товарів антикваріату. Кожен сайт спеціалізується на продажу одного окремого товару.

Ці аналоги надають широкий вибір антикварних товарів та можливості для покупців знаходити унікальні предмети. Вибір платформи залежить від вашого регіону, типу антикваріату, який вас цікавить, та особистих вподобань. Але всі ці системи є онлайн магазинами які використовуються для продажу різноманітних товарі або окремого напрямку антикваріату, чи є онлайн аукціонами. Провівши дослідження аналогів можна зробити висновок що не можна виділити якусь окрему платформу яку можна використовувати як систему для управління магазином антикваріату.

Система для управління магазином антикваріату націлена на повне управління магазином, який спеціалізується на антикваріаті. Але система є гнучкою і може бути адаптована на будь яку сферу. Перевагами цієї системи

над вищевказаними аналогами можна виділити наступні:

Систему можна використовувати як онлайн магазин так і як програмне забезпечення для магазину;

Гнучкість. Систему можна адаптувати і використовувати в будь якій сфері торгівлі;

Вдосконалена каталогізація та організація товарів: Система управління дозволяє створити цифрову базу даних антикварних товарів, включаючи їхні фотографії, описи, ціни та інші важливі деталі. Це допомагає зручно каталогізувати товари і швидко знаходити потрібні предмети, спрощує процес пошуку для клієнтів і персоналу магазину;

Моніторинг запасів: Система дозволяє вести точний облік антикварних товарів на складі магазину. Вона допомагає контролювати кількість товарів, виявляти недостачі або перебільшення запасів і сприяє управлінню запасами для ефективного замовлення та поповнення асортименту;

Оптимізація процесу продажу: Система управління може автоматизувати процес продажу антикварних товарів, включаючи ведення обліку замовлень, виставлення рахунків, обробку платежів та відстеження доставки. Це спрощує роботу персоналу і зменшує можливість помилок;

Вдосконалений клієнтський сервіс: Система може забезпечувати зручність для клієнтів, надаючи їм можливість швидко знайти та переглянути доступні антикварні товари онлайн, робити покупки в Інтернеті або резервувати товари для подальшої покупки. Крім того, система може включати інструменти для збору відгуків клієнтів та вирішення їхніх питань, що сприяє поліпшенню якості обслуговування;

Аналітика та звітність: Система може забезпечувати аналітичні інструменти та звіти про продажі, запаси, популярність певних товарів та інші ключові показники. Це допомагає зробити обґрунтовані рішення щодо управління магазином, маркетингу та стратегії розвитку;

Можливість інтегруватись з різними зовнішніми системами доставки., оплати тощо;

Можливість реалізації різних клієнтів які можуть використовувати системи і управляти магазином (Mobile додатки, Web додатки, Desktop додатки).

1.4 Постановка задачі

Постановка задачі для розробки програмного забезпечення системи управління обігом антикваріату є важливим кроком у процесі розробки.

Опис функціональності:

- Управління каталогом товарів: Система повинна дозволяти додавати, оновлювати каталог антикваріату. Для кожного товару повинна бути доступна інформація про його опис, фотографії, ціну, наявність;
- Управління запасами: Система повинна вести облік запасів товарів, автоматично оновлювати кількість товарів після продажу та відображати інформацію про кількість товарів у наявності;
- Управління замовленнями: Система повинна дозволяти клієнтам розміщувати замовлення на товари, а також обробляти ці замовлення, включаючи підтвердження, відправку та відстеження статусу доставки;
- Управління клієнтськими даними: Система повинна зберігати інформацію про клієнтів, їх контактні дані, історію покупок;

Технологічні вимоги:

- Розробка програмного забезпечення з використанням мови програмування C# та середовища розробки Visual Studio;
- Використання бази даних для зберігання даних про товари, клієнтів, замовлення та іншу інформацію;
- Вимоги до забезпечення безпеки даних, обмеження доступу до певних функцій та інформації в залежності від ролей користувачів.

Система має низку технічних характеристик та впроваджених технологій які роблять систему гнучкою і конкурентною:

- Система будується на основі мікросервісної архітектури, де різні функціональні частини реалізовані як окремі незалежні сервіси. Кожен сервіс

відповідає за конкретну функціональність, таку як каталог товарів, обробка замовлень, управління клієнтами тощо;

- Система побудована з використанням .NET 8 та основних патернів ООП;
- GraphQL використовується як механізм для виконання запитів до сервісів та отримання даних. Він дозволяє клієнтам точно вказувати необхідні дані, зменшуючи надлишковість та підвищуючи продуктивність;
- Isolated Azure Functions використані для виконання окремих завдань та функцій в системі, таких як обробка подій, розсилка сповіщень, запуск автоматичних процесів тощо;
- Azure Service Bus використовується для надійної та масштабованої комунікації між сервісами, обміну повідомленнями та керування чергами;
- Azure Blob Storage використовується для зберігання великих обсягів даних, таких як фотографії антикварних товарів або інші медіафайли;
- MongoDB використовується як база даних для зберігання та керування інформацією про антикварні товари, замовлення, клієнтів тощо. MongoDB є гнучкою та масштабованою NoSQL базою даних;
- Azure App Configuration використовується для керування конфігурацією API, такою як параметри підключення до бази даних, сервісів, безпеки тощо. Це дозволяє легко налаштувати та керувати системою;
- Azure Key Vault використовується для безпечного зберігання та керування ключами, сертифікатами та секретами, такими як паролі, токени доступу тощо;
- Active Directory використовується для керування аутентифікацією та авторизацією користувачів системи, а також для керування ролями та доступом;
- Безпека та аутентифікація:
    - JWT (JSON Web Tokens) для забезпечення аутентифікації та авторизації користувачів;

- OAuth для інтеграції зовнішніх служб аутентифікації;
- HTTPS та SSL/TLS для шифрування комунікації між клієнтом та сервером.

- Інтеграція з платіжними системами:
  - Платіжні шлюзи: Stripe, PayPal, Braintree, Authorize.Net, і т.д;
  - Інтеграція з платіжними API для обробки онлайн-платежів та розрахунків.
- Для хостингу, масштабування та управління інфраструктурою використовується Microsoft Azure;
- Інструменти для збору, аналізу та візуалізації даних, такі як Microsoft Power BI, Elasticsearch, Kibana, Grafana;
- Інструменти для моніторингу продуктивності, логування помилок, трасування даних, такі як Azure Application Insights , Grafana, Splunk, Kibana;
- Фреймворки для автоматизованого тестування, такі як Specflow, Selenium, NUnit, Jmeter;
- CICD з використанням Azure DevOps.

1.5 Висновки

Дослідження підтверджує, розробка системи для управління антикваріатом з використанням сучасних технологій може значно покращити управління цінними предметами, забезпечити їхню автентичність та цілісність, а також залучити більше клієнтів та забезпечити конкурентну перевагу на ринку антикваріату.

2 МЕТОДИ І ЗАСОБИ ПРОЕКТУВАННЯ СИСТЕМИ ДЛЯ УПРАВЛІННЯ ОБІГОМ АНТИКВАРІАТУ

2.1 Удосконалення моделі побудови системи з використанням мікросервісної архітектури на основі системи для управління обігом антикваріату

Модель архітектури системи для управління обігом антикваріату може бути побудована на основі мікросервісної архітектури, яка дозволяє розділити функціональність системи на невеликі незалежні компоненти (мікросервіси). Така архітектура полегшує розробку, масштабування та підтримку системи. Ось загальна модель архітектури для системи управління обігом антикваріату:

- Користувацький інтерфейс: Користувацький інтерфейс системи, який може бути веб-сайтом або мобільним додатком, де клієнти можуть шукати, переглядати та купувати антикварні товари.
- Мікросервіси: Кожен мікросервіс відповідає за конкретний аспект системи.:
    - Мікросервіс для управління інвентарем антикваріату, включаючи додавання нових товарів, оновлення стану товарів та видалення товарів.
    - Мікросервіс для обробки замовлень та платежів, який дозволяє клієнтам робити замовлення та здійснювати оплату.
    - Мікросервіс для рекомендацій, який надає персоналізовані рекомендації клієнтам на основі їхнього попереднього вибору та інтересів.
    - Мікросервіс для обробки оцінок та відгуків, який дозволяє клієнтам залишати відгуки про товари та визначає їхні рейтинги.

- Мікросервіс для аналітики, який збирає та аналізує дані про популярність товарів, динаміку продажів та інші статистичні дані.

- База даних: Централізована або розподілена база даних, яка забезпечує зберігання інформації про антикварні товари, клієнтів, замовлення, відгуки та інші дані, які використовуються в системі.
- Сервіси доставки та логістики: Система може інтегруватися з сервісами доставки та логістики, які дозволяють виконувати доставку антикваріату клієнтам, враховуючи різні параметри, такі як місце доставки, тип доставки та інше.
- Автентифікація та безпека: Механізми автентифікації та захисту даних для забезпечення безпеки клієнтських даних та операцій.
- Споживачі та інтеграція: Інші системи та інтеграції, такі як операційні системи для обробки замовлень, оплати та оновлення інвентаря.
- Моніторинг і аналітика: Система моніторингу та аналітики для відстеження продуктивності, виявлення проблем та вдосконалення роботи системи.
- Спілування між мікросервісами: Для покращення продуктивноті системи буде використано асинхронний підхід спілкування між мікросервісами за допомогою повідомлень. Орхестрація буде використана для відслідковування процесів що відбуваються між мікросервісами.

Ця архітектурна модель може бути розвинута та адаптована в залежності від конкретних вимог та особливостей системи для управління обігом антикваріату. Мікросервіси дозволяють розширювати функціональність та покращувати систему з часом, що робить її більш гнучкою та масштабованою.

Архітектурне проектування програмної системи є важливим етапом у процесі розробки. На цьому етапі визначається загальна структура та організація системи, її компоненти, взаємозв'язки та принципи взаємодії. Основною метою архітектурного проектування є забезпечення ефективності, масштабованості, надійності та розширюваності програмної системи.

У сучасному світі, де інформаційні технології стають необхідним атрибутом успішної діяльності в різних галузях, обіг антикваріату не є винятком. Антикварні товари завжди цінувалися за їхню унікальність, історичну цінність та естетичний вигляд. Однак, управління таким обігом може бути важким завданням через різноманітність товарів, їхню рідкість і специфічність покупців.

У цьому контексті мікросервісна архітектура набула великого значення як сучасний підхід до розробки та управління системами. Вона надає можливість створювати більш гнучкі, масштабовані та ефективні рішення для управління обігом антикваріату, забезпечуючи спрощення ряду завдань та підвищення якості обслуговування клієнтів.

В першу чергу хотілось б розглянути що ж таке мікросервісна архітектура і її переваги. Отже, мікросервісна архітектура - це підхід до розробки та управління програмними системами, при якому програмне забезпечення розбивається на невеликі, незалежні сервіси, які функціонують окремо один від одного і співпрацюють через API (інтерфейси програмування додатків). Кожен мікросервіс виконує конкретну функцію або набір функцій і може бути розроблений, впроваджений та масштабований окремо від інших сервісів.

Основні характеристики мікросервісної архітектури включають:

- Компактність та незалежність. Кожен мікросервіс є компактним та має обмежений функціонал, що дозволяє йому працювати незалежно від інших сервісів. Це полегшує розробку, тестування та впровадження.
- Складна система з менш простими частинами. Замість створення великої монолітної програми, мікросервісна архітектура розбиває систему на менші частини, які можна розробляти та масштабувати окремо. Це полегшує управління та підтримку системи.
- Спілкування. Мікросервіси спілкуються один з одним через API або повідомлення, що дозволяє їм обмінюватися даними та запитами. Це забезпечує легку інтеграцію та взаємодію сервісів.

- Незалежне впровадження та масштабування. Кожен мікросервіс може бути впроваджений та масштабований окремо, в залежності від потреб системи. Це дозволяє оптимізувати ресурси та витрати.
- Підтримка. Завдяки розбиттю системи на невеликі сервіси, управління та підтримка стають більш ефективними та прозорими.

Мікросервісна архітектура має багато переваг, завдяки чому вона стала популярною і знаходить застосування в різних галузях. Основні переваги мікросервісної архітектури включають:

- Гнучкість і масштабованість. Мікросервіси можна розробляти, впроваджувати та масштабувати окремо. Це дозволяє більш точно відповідати потребам системи і легко реагувати на зміни обсягу роботи.
- Незалежність розробки та оновлень. Кожен мікросервіс може бути розроблений, оновлений та підтриманий незалежно від інших сервісів. Це полегшує розробку, випуск оновлень та управління версіями.
- Скорочення ризиків. Якщо один мікросервіс виходить з ладу або потребує оновлення, інші сервіси можуть продовжувати працювати нормально. Це зменшує ризик великих системних збоїв.
- Легка інтеграція. Мікросервіси спілкуються між собою за допомогою API або повідомлень, що полегшує їхню інтеграцію з іншими системами та сервісами.
- Використання різних технологій. Різні мікросервіси можуть бути написані з використанням різних технологій та мов програмування, що дозволяє використовувати найкращі інструменти для конкретних завдань.
- Підвищення продуктивності розробників. Розробники можуть працювати над окремими мікросервісами, не вдаючись до складностей всієї системи. Це покращує продуктивність і дозволяє швидше розробляти та впроваджувати новий функціонал.
- Спрощене управління витратами. Витрати на інфраструктуру та ресурси можуть бути більш точно спрямовані на окремі мікросервіси залежно від їхньої навантаженості та вимог.

- Більша стабільність та надійність. В разі виходу з ладу одного мікросервіса інші можуть продовжувати працювати, що забезпечує вищу доступність системи.
- Покращена моніторинг та аналітика. Мікросервісна архітектура полегшує моніторинг і збір аналітичних даних для кожного сервісу окремо, що дозволяє швидше виявляти проблеми та оптимізувати продуктивність.
- Підтримка гібридних та хмарних інфраструктур. Мікросервіси можуть бути розподілені на різних серверах або хмарних платформах, що спрощує використання гібридних або хмарних рішень [1,2].

Усі ці переваги роблять мікросервісну архітектуру привабливим вибором для багатьох організацій, особливо в тих галузях, де потрібна висока гнучкість, масштабованість та надійність систем.

Мікросервісна архітектура дозволяє створювати гнучкі та легко масштабовані системи, а також полегшує розробку та підтримку програмного забезпечення. Вона особливо корисна в сферах, де системи повинні бути високонавантаженими, масштабованими та відновлюваними, таких як системи управління обігом антикваріату.

Система управління обігом антикваріату націлена на повне управління магазином, який спеціалізується на антикваріаті. Використання мікросервісної архітектури для системи управління обігом антикваріату може мати численні переваги і привести до покращення її ефективності та функціональності. Ось кілька основних причин, чому варто розглянути цей підхід:

- Складність та різноманітність даних. В сфері антикваріату існують різні типи товарів, кожен з яких може мати власні характеристики та деталі. Мікросервісна архітектура дозволяє створити окремі сервіси для обробки різних типів антикваріату, що спрощує роботу з різноманітними даними.
- Складність та різноманітність акторів системи. Система має велику кількість ймовірних акторів які виконують різні функції в системі.

Мікросервісна архітектура дозволяє створити окремі сервіси які будуть відображати роботу різних акторів не залежно один від одного.

- Гнучкість в розробці та розширенні. Завдяки незалежності мікросервісів, розробники можуть швидко створювати та впроваджувати новий функціонал для системи, не впливаючи на роботу інших частин системи.
- Скорочення ризиків. В разі виникнення помилок або збоїв в одному мікросервісі, це не призведе до відмови всієї системи. Решта сервісів може продовжувати працювати нормально.
- Масштабованість. Якщо потрібно обробляти більше даних або більше користувачів, окремі мікросервіси можуть бути масштабовані незалежно від інших. Це дозволяє забезпечити високий рівень обслуговування в пікові навантаження.
- Легка інтеграція з іншими системами. Важливо мати можливість легко інтегрувати систему управління обігом антикваріату з іншими сервісами, такими як платіжні шлюзи, системи доставки та інші. Мікросервіси дозволяють створювати чіткі API для інтеграції.
- Швидкий розвиток та інновації. Мікросервіси дозволяють командам розробників працювати над окремими частинами системи швидше та інноваційніше, що може допомогти залучати більше клієнтів і покращувати обслуговування [1,2, 10,11].

Основна загальна архітектура системи зображена на рисунку 2.1. Діаграма пакетів містить всі основні пакети які задіяні в системі (рисунок 2.2). Це на сам перед загальні пакети які будуть використовуватись у всіх інших пакетах і в системі в цілому. Це пакети які відповідають за роботу з різними ресурсами, такими як: управління кешом, управління базою даних, управляння репозиторієм медіа ресурсів, управління відправкою повідомлень. Також слід виділити пакети компонентів самого додатку, такі як пакети що відповідають і містять всі дані стосовно продуктів, клієнтів, заказів, репортів. Ну і відповідно окремо пакет що відповідає за інтерфейс користувача.

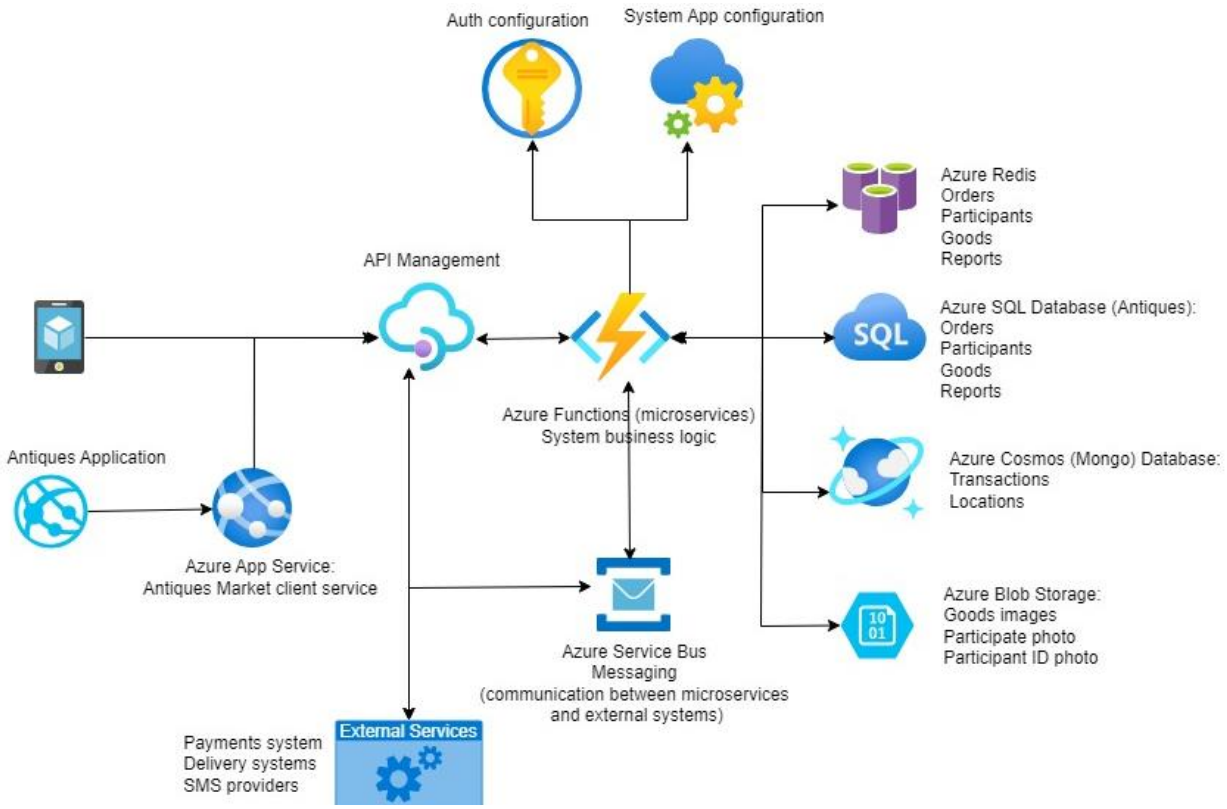


Рисунок 2.1 – Архітектура системи.

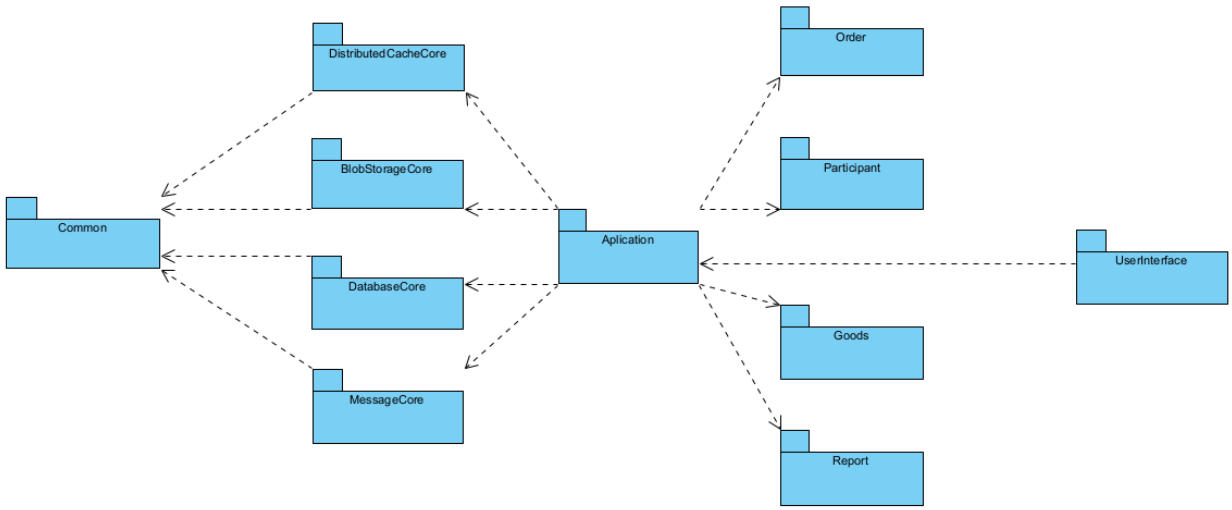


Рисунок 2.2 – Діаграма пакетів

Діаграма розгортання містить основні компоненти: сервіс інтерфейсу користувача, сервіси кожного компоненту системи, база даних, медіа і кеш репозиторії (рисунок 2.3).

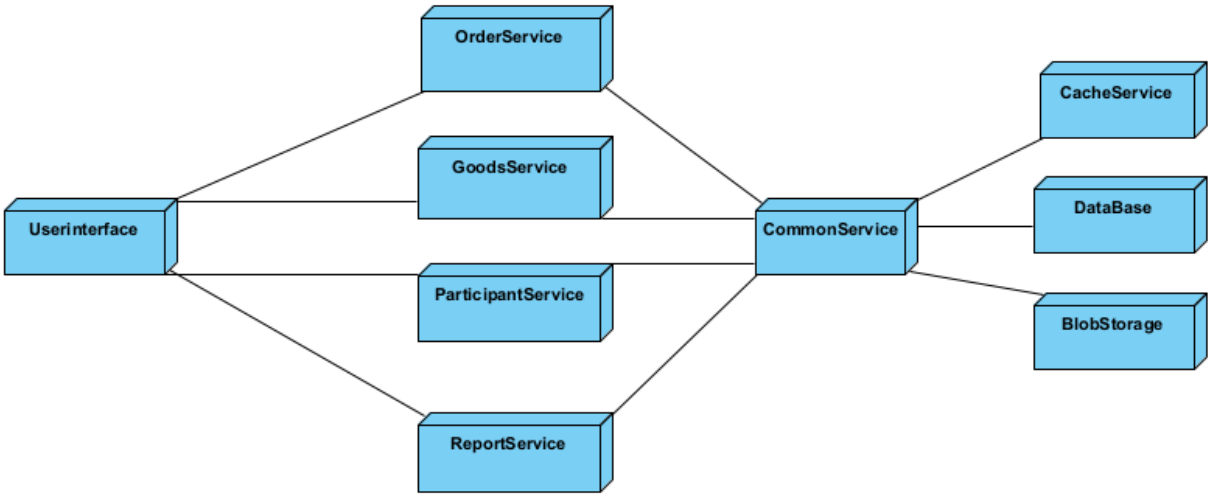


Рисунок 2.3 – Діаграма розгортання

Система буде складатись з таких модулів, це на сам перед загальних модулі які будуть використовуватись у всіх інших пакетах і в системі в цілому. Це модулі які відповідають за роботу з різними ресурсами, такими як: управління кешом, управління базою даних, управляння репозиторієм медіа ресурсів, управління відправкою повідомлень. Також слід виділити модулі компонентів самого додатку, такі як модулі що відповідають і містять всі дані стосовно продуктів, клієнтів, заказів, репортів. Ну і відповідно окремо модулі що відповідає за інтерфейс користувача.

2.2 Детальне проектування програмного забезпечення

Першим етапом проектування було створення діаграми прецедентів, так як діаграма прецедентів є корисним інструментом, який дозволяє візуалізувати функціональність системи та взаємодії з її користувачами. Тому, включення створення діаграми прецедентів на ранньому етапі може бути корисним для розуміння основних потреб та вимог користувачів системи. На рисунку 2.4 зображено діаграму прецедентів програмного забезпечення магазину антикваріату.

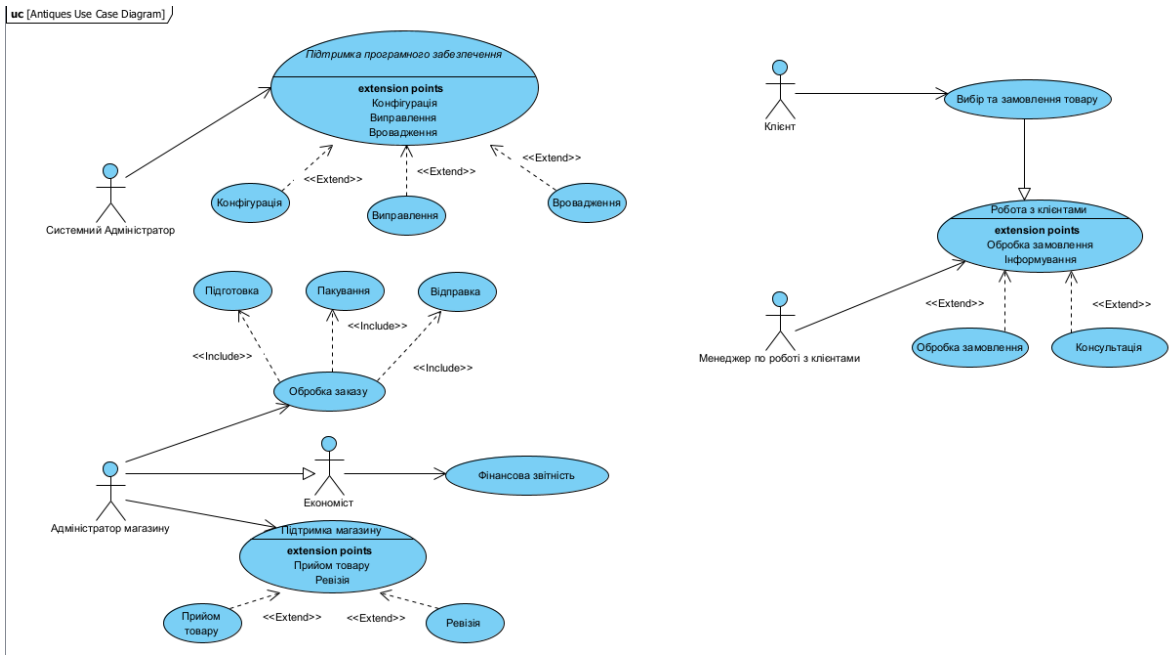


Рисунок 2.4 – Діаграма прецедентів

Актори:

- Системний адміністратор – Займається підтримкою програмного забезпечення (конфігурація додатку, підтримка та виправлення інцидентів, впровадження нових можливостей);
- Адміністратор магазину – займається обробкою заказів (підготовкою, пакуванням, відправкою), підтримкою магазину, закупівлею і прийомом нового товару, ревізія існуючого товару;
- Економіст – займається фінансовою звітністю;
- Менеджер по роботі з клієнтами – оформляє замовлення клієнта, консультація клієнта;
- Клієнт – переглядає, вибирає товар і робить замовлення.

Прецеденти:

- Підтримка програмного забезпечення – виконується системним адміністратором, дозволяє конфігурувати додаток;
- Підтримка програмного забезпечення – виконується системним адміністратором, дозволяє підтримувати і виправляти інциденти щодо несправності додатку;

- Підтримка програмного забезпечення – виконується системним адміністратором, дозволяє оновлювати додаток;
- Обробка заказу – запускається адміністратором магазину, дозволяє підготовити замовлений товар;
- Обробка заказу – запускається адміністратором магазину, дозволяє запакувати і оформити замовлений товар;
- Обробка заказу – запускається адміністратором магазину, дозволяє надати чи відправити клієнту замовлений товар;
- Підтримка магазину – запускається адміністратором магазину, дозволяє закуповувати і приймати новий товару;
- Підтримка магазину – запускається адміністратором магазину, дозволяє проводити ревізія існуючого товару;
- Фінансова звітність – запускається економістом за вимоги адміністратором магазину, дозволяє проводити фінансову звітність;
- Вибір та замовлення товару – запускається клієнтом, дозволяє вибирати, оцінювати і робити замовлення товару;
- Робота з клієнтами – запускається менеджером по роботі з клієнтами, дозволяє переглядати і обробляти замовлення клієнта;
- Робота з клієнтами – запускається менеджером по роботі з клієнтами, дозволяє надавати відповідну інформацію клієнту.

Потік подій для прецеденту «Вибір та замовлення товару»:

- Передумови: Для вибору та оформлення замовлення, клієнт має зареєструватися в додатку, заповнити форму з персональними даними та згенерувати пароль для доступу в додаток;
- Головний потік: Прецедент починає виконуватись коли клієнт входить в систему ввівши персональний пароль. Система перевіряє правильність ведених даних. Далі можливі варіанти дій: переглянути контент, зробити замовлення, замовити консультацію. Якщо вибрана операція консультація тоді в роботу вступає інший прецедент «Робота з клієнтами» і в

роботу вступає інший потік який запускається «Менеджер по роботі з клієнтами», дозволяє надавати відповідну інформацію клієнту.

Під-потоки:

- Відповідь клієнту і надання консультації в письмовому вигляді;
- Відповідь клієнту і надання консультації в телефонному вигляді;

Якщо вибрана операція замовлення тоді запускається під-потік замовлення:

Під-потоки:

- Клієнтом обирається товар(и) і заповнюється відповідна форма;
- Клієнт може додати або видалити товар з замовлення;
- Клієнт переглядає і підтверджує замовлення.

Потік подій для прецеденту «Робота з клієнтами»:

Після того як клієнт підтвердив замовлення в роботу вступає інший прецедент «Робота з клієнтами» і в роботу вступає інший потік який запускається «Менеджер по роботі з клієнтами», дозволяє переглядати і обробляти замовлення клієнта.

Головний потік: Менеджер по роботі з клієнтами опрацьовує замовлення і може виконати наступні дії: відхилити замовлення, підтвердити замовлення і передати на роботу адміністратору магазину. Під-потоки:

- Відхилення замовлення з відповідною причиною в коментарях;
- Зв'язок з клієнтом для уточнення або заміні замовлення;
- Підтвердження замовлення;

Потік подій для прецеденту «Обробка заказу»:

Після того як менеджер по роботі з клієнтами підтвердив замовлення в роботу вступає інший прецедент «Обробка заказу» і в роботу вступає інший потік який запускається адміністратором магазину.

Головний потік: Адміністратор магазину опрацьовує замовлення і може виконати наступні дії: відхилити замовлення, пустити замовлення в обробку.

Під-потоки:

- Відхилення замовлення;
- Підготовка замовлення;
- Відправка замовлення.

Переходимо до наступного важливого кроку і на основі побудованої діаграми прецедентів будуємо концептуальну діаграму класів, так як це початковий етап розробки програмного забезпечення і деталі всіх класів повністю не відомі. Але ця діаграма мстить всі сутності які відносяться до магазину антикваріату, які також описані на прецедентів. Діаграма класів представлена на рисунку 2.5.

IParticipant – інтерфейс який описує актора програмного забезпечення. Містить основні загальні данні актора.

Participant – клас який реалізує IParticipant інтерфейс (зв'язок узагальнення на діаграмі) і описує операції над актором, описує деталі актора. Це є базовий клас актора від якого унаслідуються всі інші класи які відповідають різним ролям акторів (Client, ClientManager, MarketManager, Economist, SystemManager)

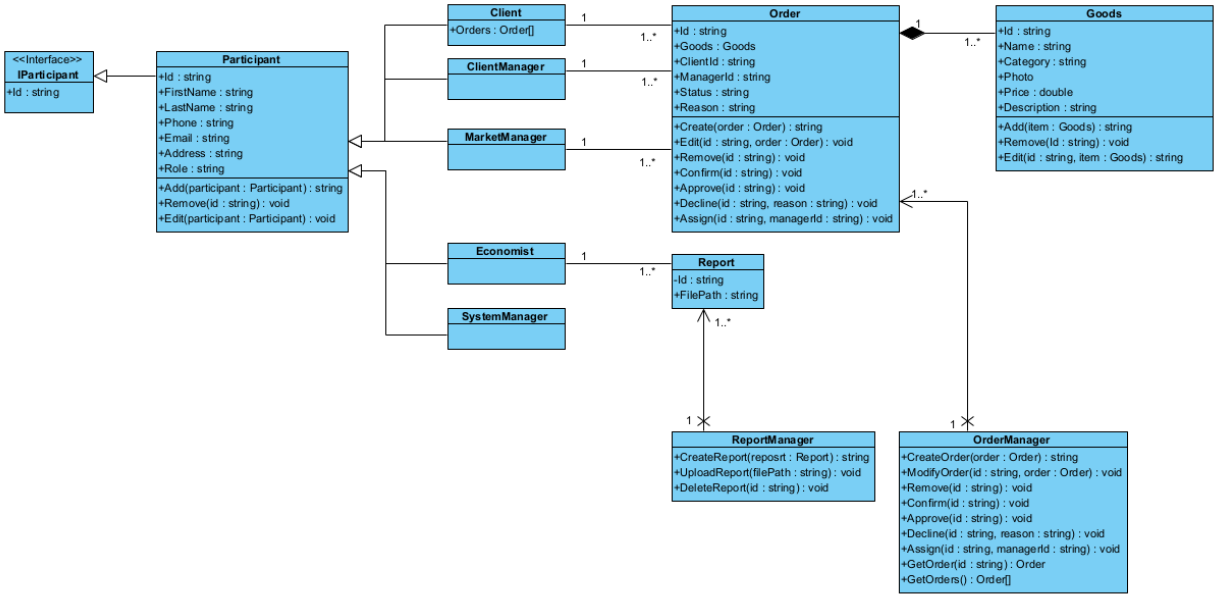


Рисунок 2.5 – Діаграма класів

Client – клас який описує поведінку і інформацію про клієнта. Цей клас є нащадком Participant класу, тому на діаграмі показаний зв'язок узагальнення.

Клієнт може зробити декілька заказів, тому на діаграмі зображено відповідна асоціація один до багатьох.

ClientManager – клас який описує поведінку і інформацію про менеджера по роботі з клієнтами. Цей клас є нащадком Participant класу, тому на діаграмі показаний зв'язок узагальнення. Менеджер по роботі з клієнтами може обробити декілька заказів, тому на діаграмі зображено відповідна асоціація один до багатьох.

MarketManager – клас який описує поведінку і інформацію про менеджера магазину. Цей клас є нащадком Participant класу, тому на діаграмі показаний зв'язок узагальнення. Менеджер магазину може обробити декілька заказів, тому на діаграмі зображено відповідна асоціація один до багатьох.

Economist – клас який описує поведінку і інформацію про економіста. Цей клас є нащадком Participant класу, тому на діаграмі показаний зв'язок узагальнення. Економіст може обробити декілька репортів, тому на діаграмі зображено відповідна асоціація один до багатьох.

SystemManager – клас який описує поведінку і інформацію про системного адміністратора. Цей клас є нащадком Participant класу, тому на діаграмі показаний зв'язок узагальнення.

Order – клас який описує поведінку і інформацію про заказ.

Goods – клас який описує поведінку і інформацію про товар. Цей клас є частиною Order класу тому на діаграмі показаний відповідний зв'язок композиції у відношені один до одного так як заказ відноситься до одного замовлення (в майбутньому може змінитись один до багатьох).

- ReportManager – відповідає за роботу над репортами (створення, завантаження і т.п.)
- OrderManager– відповідає за роботу над замовленнями (створення, обробка)

Відповідно до нашої діаграми класів ми можемо виділити наступні пакети які зображені на рисунку 2.6.

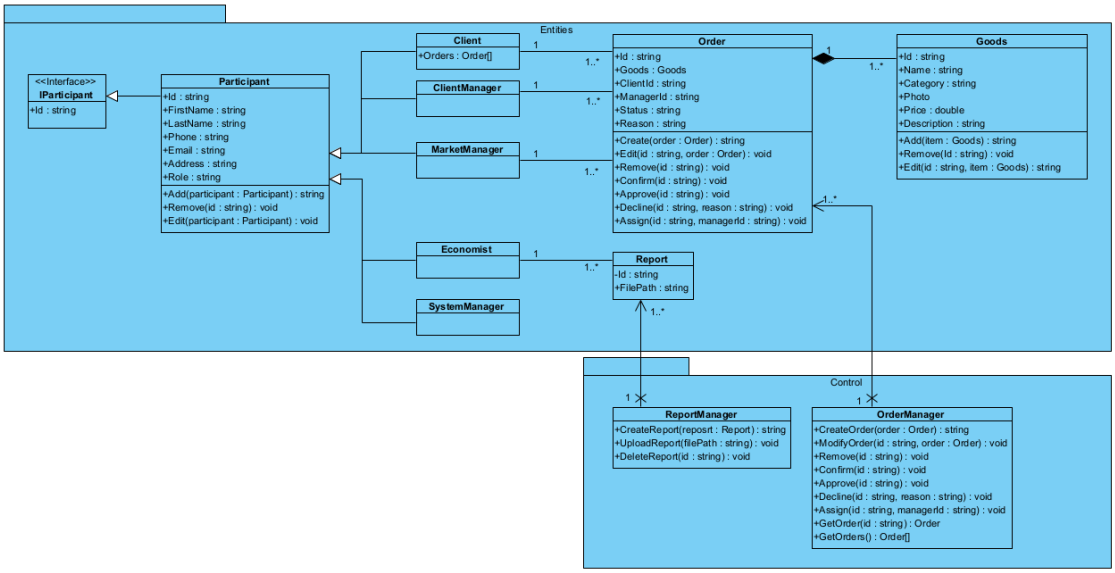


Рисунок 2.6 – Діаграма класів розподілених по пакетах

Entities – куди входять класи які описують сутності продукту: Participant, Client, ClientManager, MarketManager, Economist, SystemManager, Order, Report, Goods.

Control – куди входять класи які описують поведінку продукту: ReportManager, OrderManager.

Для опису послідовності виконання операцій та внутрішніх процесів системи використовуються діаграми послідовності та кооперацій. Було вирішено розробити детальну діаграму послідовності для процесу, що містить найбільшу кількість кроків. Це одна з головних дій клієнта, створення замовлення. Список повідомлень діаграми представлено у таблиці 2.1. Діаграми послідовності і кооперацій представлені на рисунку 2.7 і рисунку 2.8 відповідно.

На основі раніше побудованої діаграми прецедентів і діаграми послідовностей будуємо діаграму активностей. Першим етапом створимо діаграму активностей яка описує бізнес-процес. Для побудови візьмемо один із основних процесів додатку, замовлення і оформлення замовлення. Діаграма активностей бізнес процесу, що описує роботу клієнта і менеджера по роботі з клієнтом в рамках оформлення заказу, зображена на рисунку 2.9.

Таблиця 2.1 – Повідомлення у діаграмі послідовностей

Номер повідомлення	Об’єкт-відпавник	Об’єкт-одержувач	Назва
1	Client	LoginManager	Login
1.1	LoginManager	Participant	Check participant
1.2	Participant	LoginManager	Participant is confirmed
1.3	LoginManager	LoginManager	Save login history
1.4	LoginManager	Client	Client is logged in
2	Client	GoodsManager	Get list of goods
2.1	GoodsManager	Goods	Get list of goods
2.2	Goods	GoodsManager	List of goods
2.3	GoodsManager	GoodsManager	Save to cache
2.4	GoodsManager	GoodsManager	Get from cache
2.5	GoodsManager	Client	Return list of goods
3	Client	GoodsManager	Select goods
3.1	GoodsManager	Goods	Mark goods as selected
3.2	Goods	GoodsManager	Ok
3.3	GoodsManager	Client	Ok
4	Client	OrderManager	Create Order
4.1	OrderManager	OrderManager	Save to cache
4.2	OrderManager	Client	Ok
5	Client	OrderManager	Confirm order
5.1	OrderManager	Order	Add order
5.2	Order	OrderManager	Ok
5.3	OrderManager	OrderManager	Remove from cache
5.4	OrderManager	Client	Ok

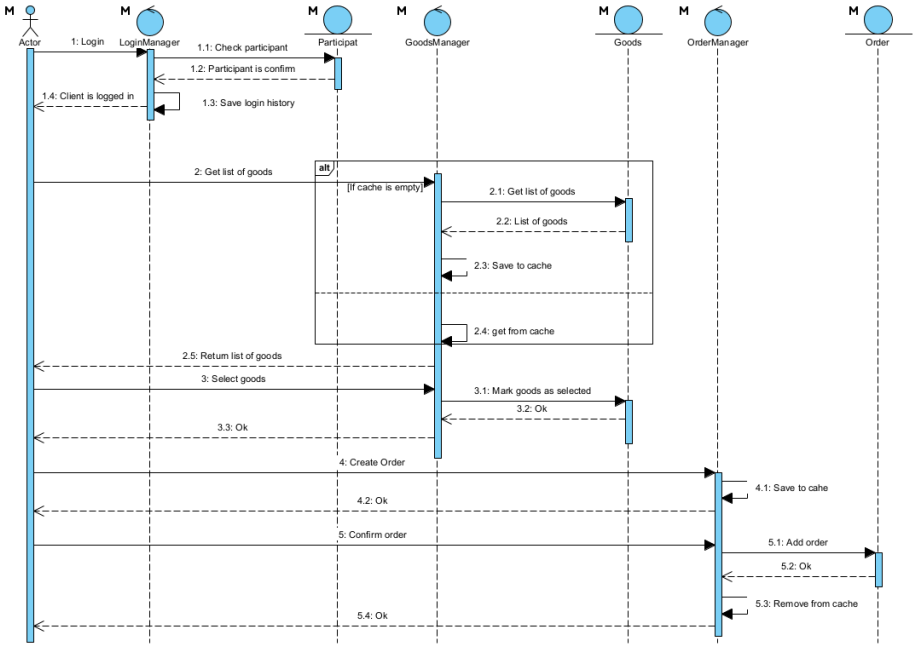


Рисунок 2.7 – Діаграма послідовностей

Процес починається з входом в систему клієнтом і закінчується отриманням заказу. Також бізнес процес можна розширити ще одним актором (Менеджер магазину), який описаний в попередніх діаграмах. В даній діаграмі весь процес за який відповідає менеджер магазину описаний одним блоком активності «Оформлення, підготовка і відправка товару»

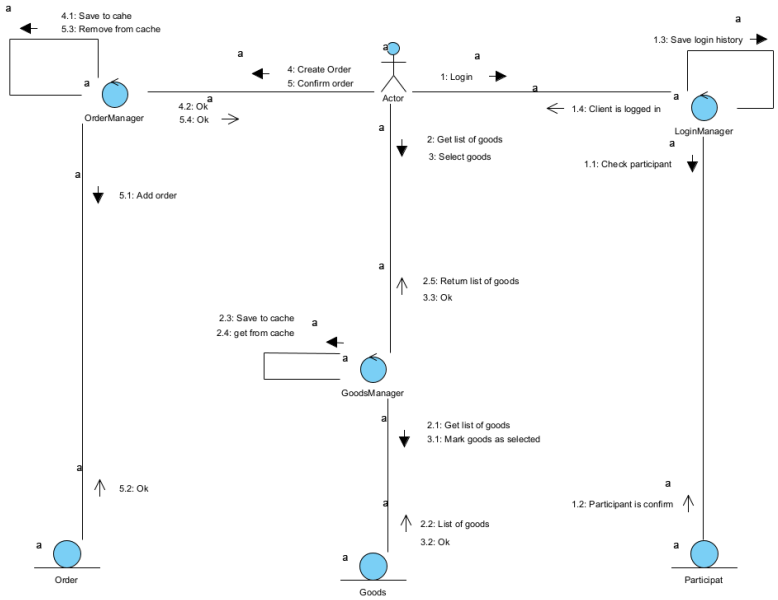


Рисунок 2.8 – Діаграма кооперацій

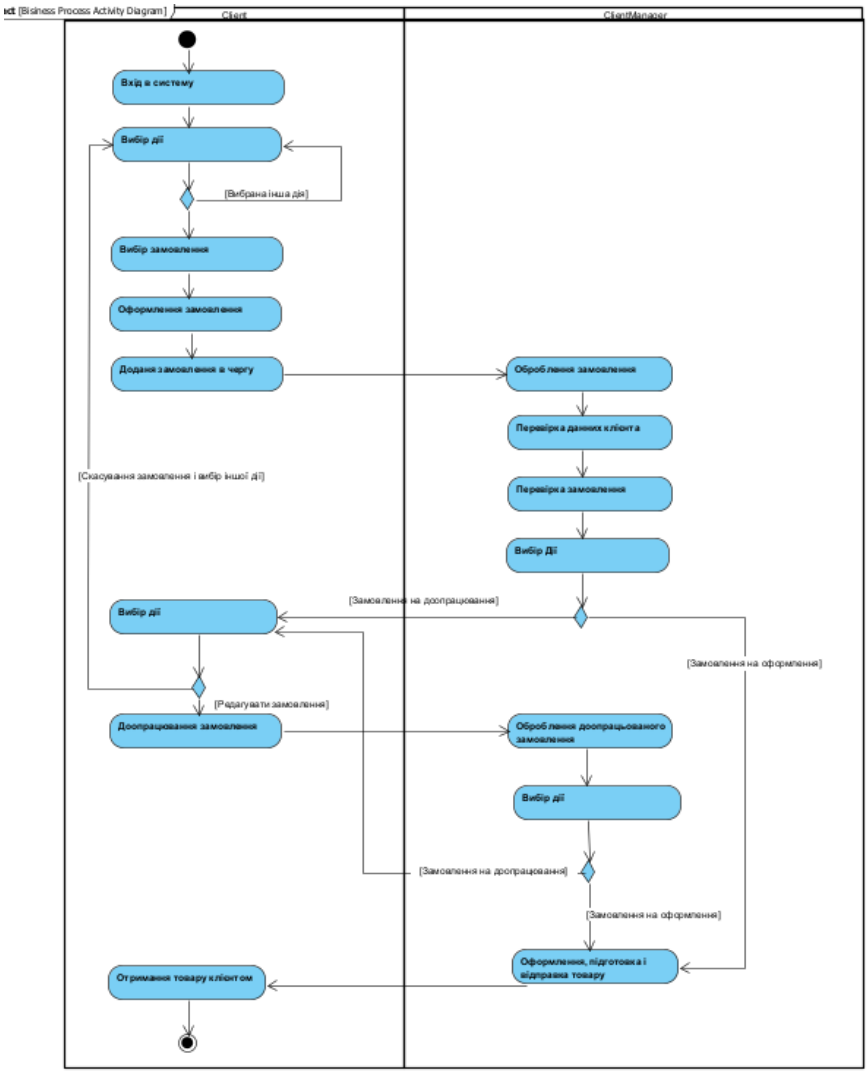


Рисунок 2.9 – Діаграма активності процесу оформлення замовлення

Це процес є складним і потребує більш детального опису кожної дії яка описана в бізнес процесі. Тому створимо більш детальну діаграму діяльності клієнта. Для створення діаграми візьмемо основний процес клієнта. Діаграмі діяльності клієнта в системі зображена на рисунку 2.10.

Старт процесу починається з входу в систему клієнтом, далі у клієнта є вибір: вийти з системи чи авторизуватися і продовжити роботу. Після того як клієнт вибрав авторизацію, система починає перевірку користувача. І якщо користувач пройшов перевірку тоді клієнт може виконувати наступні дії в системі. Але якщо клієнт не пройшов перевірку, тоді система пропонує клієнту пройти авторизацію ще раз або вийти з системи.

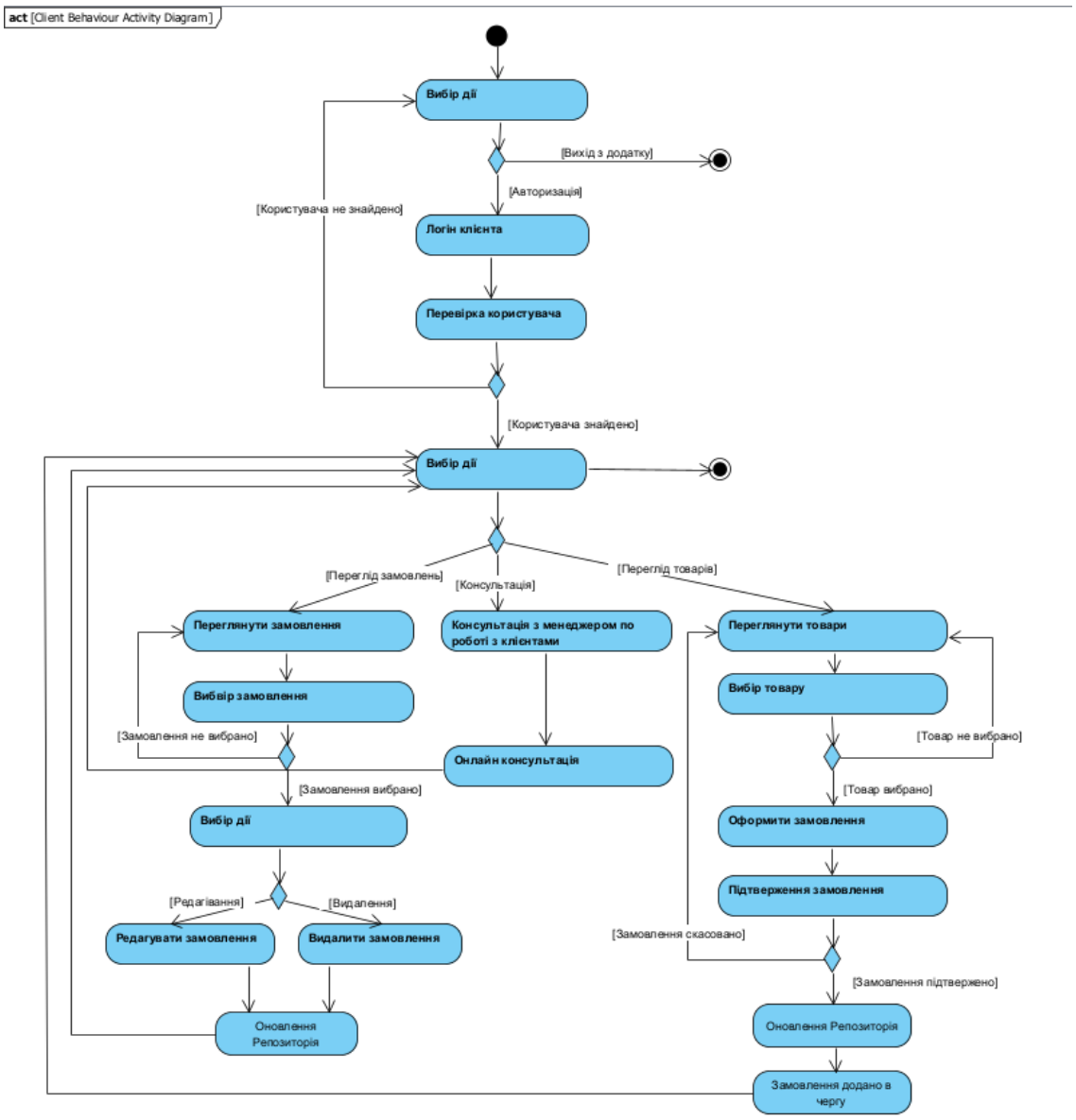


Рисунок 2.10 - Діаграма діяльності клієнта в системі

Після того як клієнт пройшов авторизацію він може вибрати декілька опцій наступних кроків:

- Переглянути замовлення – вибравши цю опцію, клієнт може переглядати всі свої, раніше створені, замовлення. Також клієнт може видалити своє замовлення, доповнити чи редагувати на вимогу магазину або вийти з системи чи вибрати іншу опцію дій;
- Консультація з менеджером по роботі з клієнтами – вибравши цю опцію, клієнт замовляє онлайн консультацію з менеджером по роботі з

клієнтами. Після консультації клієнт може вибрати іншу дію або вийти з системи;

- Переглянути товари – вибравши цю опцію, клієнт може переглядати всі наявні в магазині товари. Також клієнт вибрати відповідний товар і оформити замовлення або вийти з системи чи вибрати іншу опцію дій. Після оформлення замовлення клієнту надає можливість продовжити дії в системі або вийти з системи.

Наступним етапом можна виділити побудову діаграми станів. Першим етапом створимо діаграму станів системи в цілому, яка зображена на рисунку 2.11. Опис кожного стану описано в таблиці 2.2.

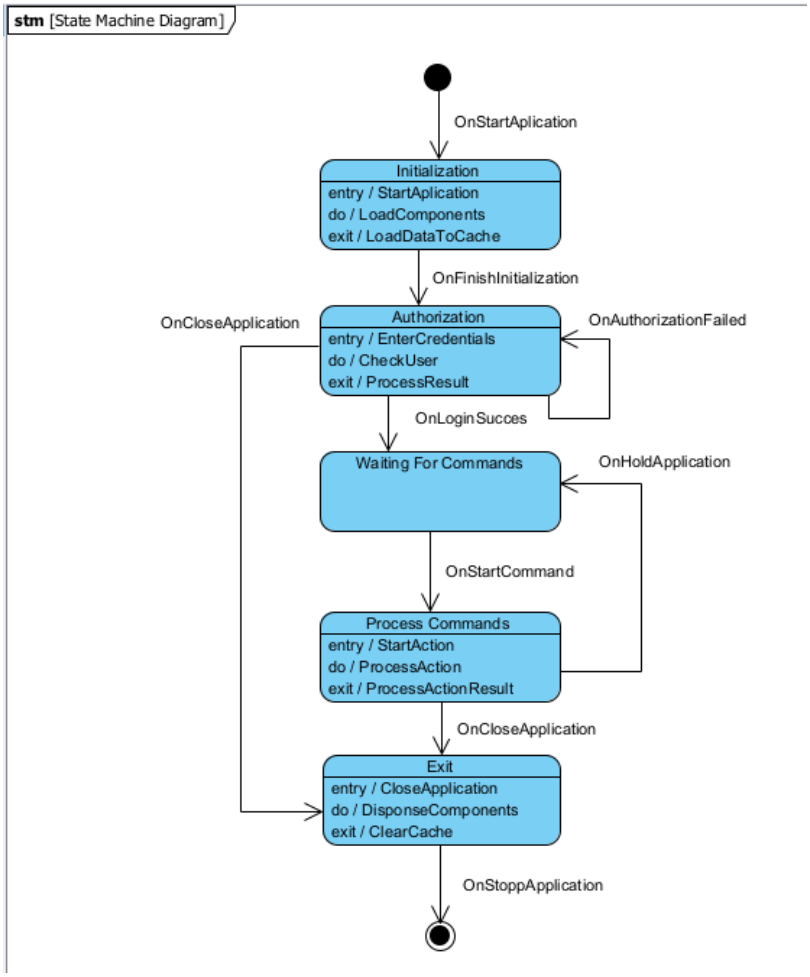


Рисунок 2.11 – Діаграма станів системи в цілому

Таблиця 2.2 – Опис станів системи в цілому

Стан	Опис стану
Initialization	Стан ініціалізації додатку відбувається при запуску системи. Коли додаток знаходиться в цьому стані, система завантажує всі необхідні для роботи компоненти. Виходом з цього стану буде подія завершення ініціалізації системи і збереження відповідних даних відображення в кеші
Authorization	Стан авторизації відбувається при вході в систему. Відбувається перевірка даних користувача. Виходом з цього стану можуть буди дві події. Перша це авторизація пройшла успішно, у цьому випадку система переходить в наступний стан. Друга це авторизація не пройшла успішно, у цьому випадку система повертається в цей самий стан
Waiting For Commands	Стан очікування системи
Process Commands	Стан обробки команди системою. Відбувається стан при початку виконані дії і завершується після завершення виконання. Стан може перейти в попередній стан або в стан виходу з додатку
Exit	В стан виходу система переходить після завершення роботи в системі. У цьому стані відбувається завершення всіх компонентів системи

Наступним етапом буде створення діаграми станів основного компонента системи – це діаграма станів замовлення. Замовлення в системі буде проходити через декілька акторів, тому замовлення може мати багато станів. Діаграма станів замовлення зображено на рисунку 2.12. Опис кожного стану описано в таблиці 2.3.

На основі раніше побудованих діаграм будуємо діаграму компонентів. Основними компонентами системи є головний компонент (сам додаток, який взаємодіє з усіма іншими компонентами), інтерфейс користувача, бізнес логіка і компоненти які відповідають за роботу з репозиторіями даних. На початковій стадії бізнес логіка буде реалізована як суцільний окремий компонент, але нових версіях продукту це будуть окремі компоненти (мікросервіси) які будуть відповідати за окрему частину роботи. Кожен компонент бізнес логіки

буде взаємодіяти з іншими компонентами. Діаграма компонентів зображена на рисунку 2.13.

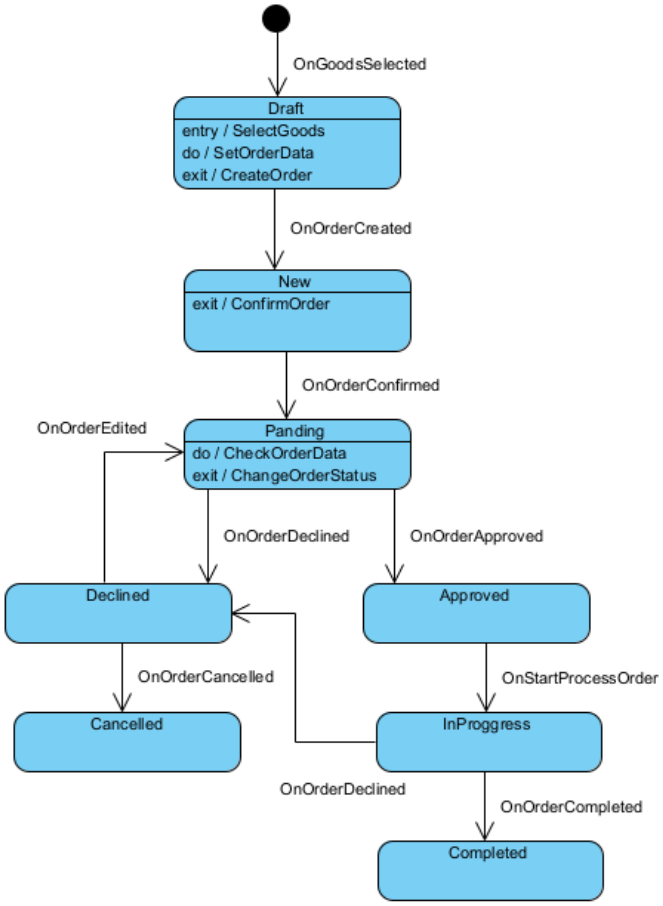


Рисунок 2.12 – Діаграма станів об’єкту замовлення

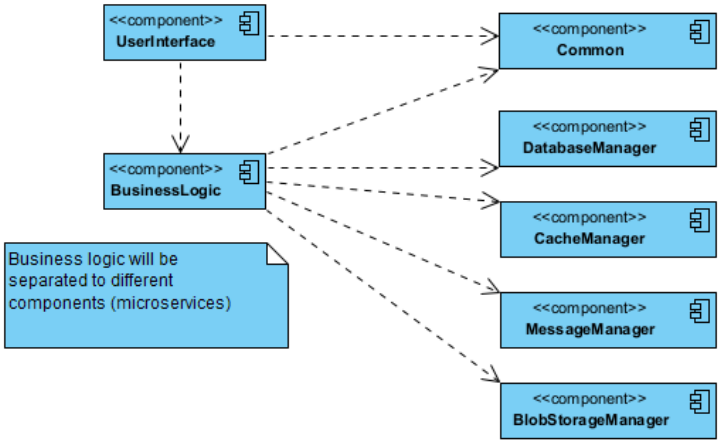


Рисунок 2.13 – Діаграма компонентів

Таблиця 2.3 – Опис станів замовлення

Стан	Опис стану
Draft	В стан Draft замовлення переходить після того як клієнт вибрав товар і почав оформляти замовлення.
New	В стан New замовлення переходить після того як клієнт створив замовлення.
Pending	Стан Pending замовлення переходить після того як клієнт підтвердив створене замовлення.
Approved	Стан Approved замовлення переходить після того як менеджер по роботі з клієнтами продивився данні замовленні і підтвердив наявність продукту і відповідність даних.
Declined	Стан Approved замовлення переходить після того як менеджер по роботі з клієнтами продивився данні замовленні і відхилив замовлення з відповідною причиною (немає в наявності продукту, не відповідність даних, то що). З цього стану замовлення може перейти в стан Pending після редагування клієнтом або в стан Canceled якщо клієнт вирішив скасувати замовлення.
InProggress	Стан InProggress замовлення переходить після того як менеджер магазину продивився данні замовленні, замовлення і взяв замовлення в роботу. З цього стану замовлення може перейти в стан Declined, після того як менеджер замовлення обробив замовлення і відхилив з деякої причини, або в стан Completed якщо замовлення було опрацьовано успішно менеджером магазину.
Completed	Стан InProggress замовлення переходить після того як замовлення було опрацьовано успішно менеджером магазину.
Canceled	Стан Canceled замовлення переходить після того як клієнт вирішив скасувати замовлення.

Отже ми провели детальне проектування поставлених задач і побудували відповідні діаграми що відображають основні бізнес процеси. Виходячи з цього можна виділити основні задачі в реалізації програмного забезпечення.

2.3 Удосконалення моделі побудови системи аналізу даних на основі системи для управління обігом антикваріату

Аналіз даних в системі управління антикваріатом може бути корисним для оптимізації управління асортиментом, прогнозу попиту, встановлення цін та покращення загальної продуктивності. Ось деякі аспекти, які можна аналізувати:

Аналіз асортименту:

- Визначення популярних та менш популярних товарів в асортименті.
- Виявлення сезонних та тематичних тенденцій.
- Аналіз категорій товарів, що продаються найкраще.

Аналіз попиту та клієнтського поведінки:

- Вивчення динаміки попиту на різні види антикваріату.
- Визначення профілю клієнтів та їхніх вподобань.
- Аналіз змін у попиті залежно від часу року, свят та інших факторів.

Фінансовий аналіз:

- Моніторинг прибутковості різних товарів.
- Розрахунок маржі та валового прибутку.
- Аналіз витрат та оптимізація управління запасами.

Ціноутворення та стратегії знижок:

- Визначення оптимальних цін для товарів на основі аналізу конкурентів та попиту.
- Розроблення стратегій знижок та акцій для привертання клієнтів.

Прогнозування запасів та попиту:

- Використання методів прогнозування, таких як часові ряди або машинне навчання, для передбачення попиту та потреб у запасах.
- Планування запасів та поставок товарів на підставі прогнозів.

Ефективність маркетингу та реклами:

- Вимірювання результатів маркетингових кампаній та рекламних заходів.
- Визначення, які канали та стратегії найбільше впливають на продажі.

Споживча зворотність:

- Вивчення відгуків клієнтів та реакція на них.
- Покращення обслуговування клієнтів та якості товарів на основі зворотнього зв'язку.

Конкурентний аналіз:

- Аналіз діяльності конкурентів та їхніх стратегій.
- Визначення переваг та недоліків власного бізнесу порівняно з конкурентами.

Аналіз даних може бути виконаний за допомогою різних інструментів, включаючи статистичний аналіз, машинне навчання, бізнес-інтелект і програми для обробки даних. Результати аналізу допоможуть вдосконалити стратегії управління антикваріатом, збільшити ефективність та задоволеність клієнтів, а також підвищити прибутковість вашого бізнесу.

Проектування системи в Azure для аналізу даних системи управління антикваріатом може бути важливим завданням для ефективного управління і оптимізації бізнес-процесів. Нижче наведено кроки та компоненти, які можуть бути включені у таку систему:

Збір та зберігання даних:

- Спроектуйте схему бази даних для зберігання історичних та поточних даних про антикваріатні товари.
- Використовуйте Azure SQL Database або Azure Cosmos DB для зберігання структурованих та неструктурованих даних.

Інтеграція з джерелами даних:

- Налаштуйте процеси збору даних з різних джерел, таких як магазини, онлайн майданчики або аукціони.

- Використовуйте Azure Data Factory або Azure Logic Apps для автоматизації інтеграції даних.

Аналітичні інструменти:

- Використовуйте Azure Synapse Analytics для аналізу великих обсягів даних та створення звітів і аналітики.
- Застосуйте Power BI для візуалізації даних та створення інтерактивних звітів.

Машинне навчання:

- Використовуйте Azure Machine Learning для прогнозування цін на антикваріатні товари або виявлення трендів у сфері антикваріату.
- Впроваджуйте моделі машинного навчання у виробництво для автоматичного аналізу даних.

Захист даних:

- Застосовуйте Azure Security Center та Azure Active Directory для забезпечення безпеки даних та керування доступом.
- Шифруйте дані в покої для зберігання та передачі за допомогою Azure Key Vault.

Моніторинг і оптимізація:

- Використовуйте Azure Monitor для моніторингу працездатності системи та її складових.
- Оптимізуйте ресурси за допомогою Azure Cost Management та Azure Advisor.

Інтеграція з розподільчими системами:

- Якщо є потреба, інтегруйте систему управління антикваріатом з іншими бізнес-системами, такими як система обліку товарів або система замовлень.

Скальованість:

- Забезпечте горизонтальну та вертикальну скальованість системи для відповіді на зростаючий обсяг даних та завдань обробки.

Резервне копіювання та відновлення:

- Налаштуйте щоденне резервне копіювання даних та відновлення в разі втрати інформації.

Документація та навчання:

- Забезпечте документацію для адміністраторів та користувачів системи.
- Забезпечте навчання для персоналу щодо користування системою та її компонентами.

Це загальний план для проектування системи в Azure для аналізу даних системи управління антикваріатом. Кожен бізнес може мати свої власні вимоги і варіації в цьому плані, тому важливо провести детальний аналіз потреб та обмежень вашої конкретної організації перед розробкою та впровадженням системи.

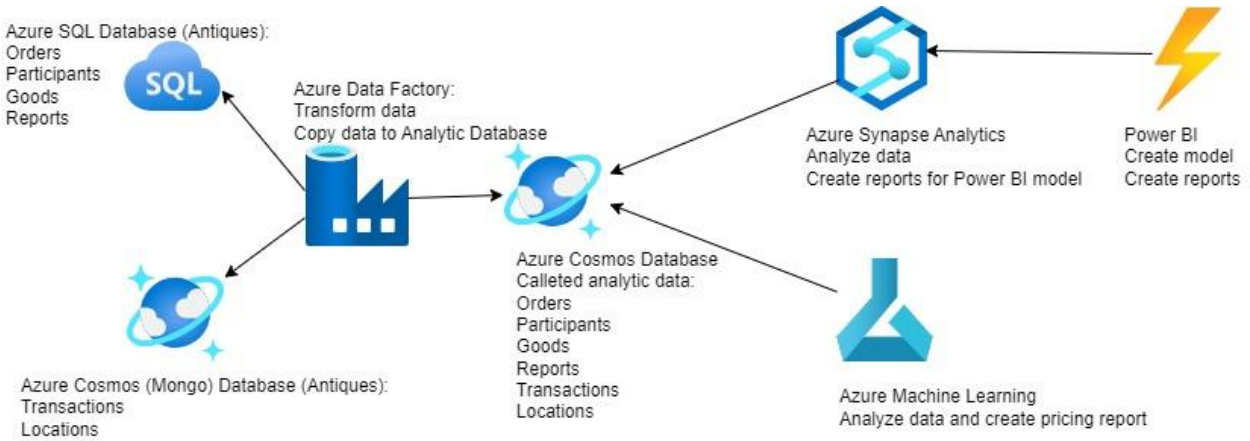


Рисунок 2.14 – Архітектура системи аналізу даних

Отже, враховуючи вищесказане можна спроектувати систему аналізу даних системи управління антикваріатом. На рисунку 2.14 зображено схему архітектури системи аналізу даних. Цей підхід дозволить більш ефективніше та швидше формувати данні які потрібні для аналізу та у реалізації такої функціональності як пошуку оптимального шляху доставки та пошуку ф формуванню цін.

2.4 Удосконалення методу пошуку оптимального шляху на основі системи для управління обігом антикваріату

Реалізація пошуку оптимальних антикварних товарів, знижок є важливою складовою для різних сторін сучасного ринку і має значний вплив на різні групи людей і суспільство в цілому. Також оптимальний пошук має місце в пошуку оптимального шляху доставк товару. Алгоритм пошуку має бути сучасним і постійно вдосконалюватись так як ринок і технології постійно оновлюються. Щоб вирішити цю задачу найкращим кандидатом є підхід з використанням нейронних мереж. Нейронні мережі можуть допомогти в автоматизації процесу пошуку і фільтрації антикварних товарів на основі ваших власних вимог та уподобань. Нижче наведено загальний план, який можна використовувати для створення такої системи:

- Збір даних: Одним із перших кроків є створення бази даних з антикварними товари, локацій складів, локацій клієнтів та вже існуючих шляхів які були виявлені за допомогою геолокації та алгоритмів маршрутизації. Це може включати інформацію про товари, таку як назва, опис, вартість, роки виготовлення, категорія і фотографії.
- Побудова нейронної мережі: Вам потрібно буде створити нейронну мережу, яка зможе аналізувати ці дані і робити рекомендації. Один із можливих варіантів - це використання моделі глибокого навчання, такої як рекурентна нейронна мережа (RNN) або мережа прямого поширення (feedforward neural network). Мережа може вивчати кореляції між вимогами користувачів та антикварними товари.
- Попередня обробка даних: Важливо обробити дані перед подачею їх на вхід до нейронної мережі. Це включає в себе нормалізацію, видалення відсутніх даних та кодування категоріальних змінних.
- Тренування мережі: Ви повинні навчити нейронну мережу на вашому наборі даних. Під час навчання модель навчиться робити прогнози і

робити рекомендації щодо антикварних товарів, які можуть зацікавити користувача.

- Інтеграція з інтерфейсом користувача: Побудуйте інтерфейс користувача, який дозволить користувачам встановлювати свої уподобання та запити для пошуку антикварних товарів.
- Результати і рекомендації: Після введення вимог користувача система може повертати рекомендації щодо оптимальних антикварних товарів, які відповідають їхнім вимогам.
- Зворотний зв'язок та покращення: Постійно вдосконалюйте вашу систему, враховуючи зворотний зв'язок від користувачів і навчання моделі на нових даних.
- Забезпечення конфіденційності і безпеки даних: Важливо забезпечити безпеку та конфіденційність особистих даних користувачів, коли вони користуються вашою системою.
- Масштабування системи: Після успішної реалізації можливо розширити систему та впровадити її на ринку.

Спершу розглянемо модель прямого поширення (feedforward neural network), а потім рекурентну нейронну мережу (RNN). Обидві моделі є популярними для різних завдань машинного навчання.

Модель прямого поширення (Feedforward Neural Network). Модель прямого поширення (FFNN), також відома як штучний нейронний шар, - це нейронна мережа, де дані рухаються в одному напрямку, без циклів або зворотніх зв'язків. Основними компонентами FFNN є шари нейронів, які приймають вхідні дані, виконують обчислення і передають результати на наступний шар. Ось кроки для створення моделі FFNN:

- Імпортування бібліотек: Спершу вам потрібно імпортувати необхідні бібліотеки, такі як TensorFlow або PyTorch, для створення та навчання моделі.
- Створення моделі: Визначте архітектуру моделі, включаючи кількість і розміри прихованих шарів та кількість нейронів в кожному шарі.

- Визначення функції втрат: Виберіть функцію втрат, яка визначає, наскільки модель точно передбачає вихідні значення. Це може бути, наприклад, середньоквадратична помилка для регресії або категоріальна крос-ентропія для класифікації.
- Компіляція моделі: Вказіть оптимізатор і метрики, які ви хочете використовувати для навчання моделі.
- Тренування моделі: Використовуйте навчальні дані для тренування моделі. Це включає в себе передачу вхідних даних через модель, обчислення втрат, оновлення ваг та повторення цього процесу протягом декількох епох.
- Оцінка та тестування моделі: Після навчання оцініть модель за допомогою тестових даних, щоб переконатися в її точності та ефективності.

Рекурентна нейронна мережа (RNN). Рекурентна нейронна мережа (RNN) - це мережа, яка має зв'язки, які утворюють цикл, що дозволяє передавати інформацію з попередніх кроків часу в майбутнє. Ця особливість робить RNN корисною для обробки послідовних даних, таких як текст або часові ряди. Ось кроки для створення RNN:

- Імпортування бібліотек: Як і в моделі FFNN, спершу імпортуйте необхідні бібліотеки.
- Створення моделі RNN: Визначте архітектуру моделі RNN, включаючи кількість шарів та кількість нейронів в кожному шарі. Важливо визначити, чи ця мережа буде однорідною або мається на увазі механізм забування (як у LSTM або GRU).
- Визначення функції втрат і оптимізатора: Оберіть функцію втрат і оптимізатор для вашої моделі, так само, як і в FFNN.
- Компіляція моделі: Визначте оптимізатор та метрики для навчання RNN.
- Тренування моделі: Використовуйте тренувальні дані та навчіть RNN. Зауважте, що в RNN важливо враховувати часові залежності даних.

- Оцінка та тестування моделі: Після навчання оцініть модель за допомогою тестових даних.

При використанні RNN для обробки послідовних даних, рекомендується також розглянути розширені архітектури, такі як LSTM або GRU, які допомагають вирішити проблему зникнення та вибування градієнтів.

Рекурентні нейронні мережі (RNN) і моделі прямого поширення (feedforward neural networks) відрізняються за своєю архітектурою та способом роботи. Ось порівняльна характеристика цих двох типів нейронних мереж:

Архітектура:

- RNN: RNN має циклічну структуру, яка дозволяє інформації переходити від одного часового кроку до іншого. Це робить RNN добре підходящою для обробки послідовних даних, таких як текст або часові ряди.
- FFNN: Модель прямого поширення має пряму структуру без циклів. Вхідні дані пересуваються в одному напрямку через різні шари нейронів, і немає зворотніх зв'язків для передачі інформації в минуле.

Застосування:

- RNN: Використовується для завдань, які вимагають обробки послідовних даних, таких як обробка тексту, машинний переклад, розпізнавання мови, генерація тексту і прогнозування часових рядів.
- FFNN: Зазвичай використовується для завдань, де вхідні дані не мають послідовності або залежностей в часі. Це може бути завдання класифікації, регресії або розпізнавання об'єктів на зображеннях.

Робота з послідовностями:

- RNN: RNN призначена для роботи з послідовностями і може враховувати залежності в часі. Вона здатна зберігати попередні стани та використовувати їх для аналізу нових вхідних даних.
- FFNN: FFNN не має здатності обробляти послідовності без додаткових зусиль. Якщо потрібно аналізувати послідовності, використовуються RNN або інші рекурентні архітектури.

Проблема зникнення та вибування градієнтів:

- RNN: RNN має тенденцію до проблеми зникнення та вибування градієнтів, особливо на довгих послідовностях. Це може ускладнювати тренування глибоких RNN на деяких завданнях.
- FFNN: FFNN, зазвичай, має менше проблем з градієнтами, оскільки вона не має циклічних зв'язків і, таким чином, немає довгих залежностей.

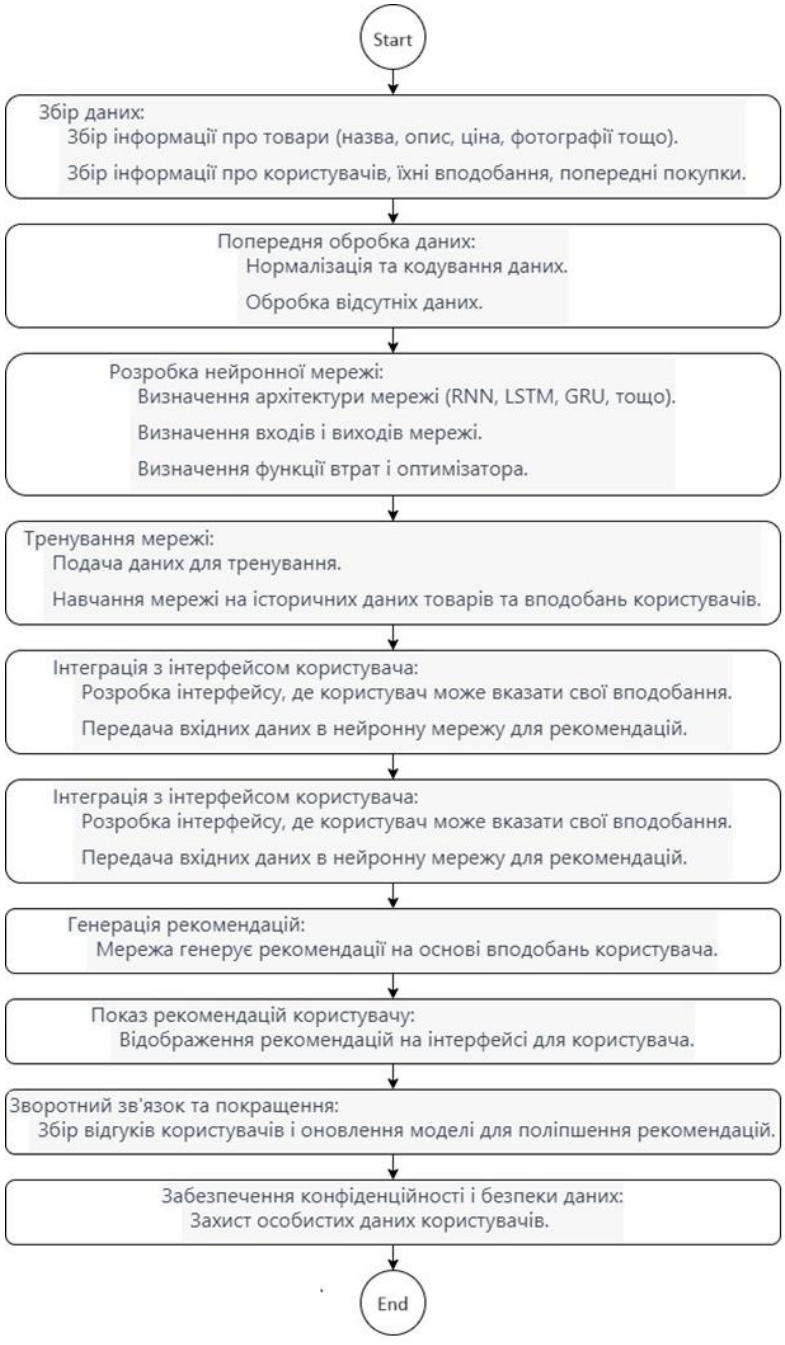


Рисунок 2.15 – Блок-схема алгоритму для пошуку оптимальних товарів з використанням нейронних мереж

Створення блок-схеми алгоритму для оптимального пошуку з використанням нейронних мереж може бути складним завданням через велику кількість деталей і фаз, які включаються у процес. Також у задачі пошуку оптимального шляху доставки можна використовувати різні типи нейронних мереж, наприклад:

- Звичайні нейронні мережі (ANN): Для прогнозування часу доставки можна використовувати мережі з декількома шарами, що допоможе зв'язати різні вхідні фактори з часом доставки.
- Згорткові нейронні мережі (CNN): Можуть бути використані для аналізу карт наочних даних, таких як мапи, для визначення оптимального маршруту.
- Рекурентні нейронні мережі (RNN): Для аналізу часових рядів (наприклад, трафік на дорогах) та передбачення часу доставки в залежності від часу доби.

Отже загальна блок-схема, яка ілюструє основні кроки і компоненти алгоритму для оптимального пошуку з використанням нейронних мереж зображано на рисунку 2.15. Використання такої моделі допоможе прогнозування часу доставки в залежності від різних факторів, таких як відстань, тип транспорту, трафік, погода тощо. Також у визначенні оптимального маршруту з урахуванням різних обмежень, таких як об’єм товару, доступність доріг або точок доставки.

Нейронні мережі потребують великої кількості даних для тренування та налаштування. Тому є важливим добре спроектувати систему аналізу данних.

3 РОЗРОБКА ПРОТОТИПУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Варіантний аналіз та обґрунтування вибору способів реалізації програмного засобу

Варіантний аналіз та обґрунтування вибору способів реалізації програмного засобу є важливою частиною процесу розробки програмного продукту. Цей процес допомагає визначити найкращий спосіб для досягнення поставленої мети з урахуванням обмежень та вимог проекту. Нижче наведено кроки, які можна виконати при варіантному аналізі та обґрунтуванні вибору способів реалізації програмного засобу:

- Визначення вимог до програмного засобу. Почніть з ретельного аналізу вимог, які ставляться до програмного засобу. Це включає в себе функціональні, нефункціональні, технічні та економічні вимоги. Ретельне розуміння вимог є ключовим етапом.
- Визначення альтернативних рішень. Розгляньте різні можливі шляхи реалізації програмного засобу. Наприклад, це може бути вибір мови програмування, платформи, архітектури, технологій тощо.
- Оцінка альтернатив. Для кожного варіанту реалізації проведіть оцінку, враховуючи такі фактори, як складність реалізації, технічні можливості, швидкість роботи, масштабованість, вартість розробки та підтримки, зручність для користувачів тощо.
- Порівняння та вибір оптимального варіанту. Порівняйте оцінки для різних альтернатив та оберіть оптимальний варіант реалізації. При цьому обов'язково враховуйте пріоритети проекту, обмеження та вимоги замовника.
- Обґрунтування вибору. Відповідно до результатів аналізу поясніть, чому було обрано конкретний варіант реалізації. Вказуйте на переваги та недоліки інших альтернатив і пояснюйте, як обраний варіант відповідає вимогам проекту найкраще.

- Розробка деталей плану. Після обрання оптимального варіанту реалізації розробіть деталізований план робіт, який включає в себе кроки, терміни, ресурси та відповідальних осіб.
- Моніторинг та коригування. Процес вибору способу реалізації повинен бути піддано моніторингу на протязі всього проекту. Якщо з'являються нові вимоги або змінюються обставини, може знадобитися коригування обраного варіанту.

Варіантний аналіз та обґрунтування вибору способів реалізації програмного засобу допомагають забезпечити ефективну та успішну реалізацію проекту, зменшити ризики та максимізувати відповідність вимогам замовника.

Порівняння різних платформ розробки може бути складним завданням, оскільки вибір платформи зазвичай залежить від конкретних потреб та вимог проекту. Однак я можу навести загальні порівняльні характеристики двох популярних платформ розробки: Java та .NET (C#).

Таблиця 3.1 – Порівняння популярних платформ розробки

Java	.NET (C#)
Кросплатформенність: Java розроблялася з орієнтацією на кросплатформенність, що дозволяє запускати Java-програми на різних операційних системах без модифікацій.	Кросплатформенність: .NET також став кросплатформеним, починаючи з версії .NET Core (пізніше перейменовано в .NET 5 і .NET 6), дозволяючи розробляти на Windows, macOS і Linux.
Мова програмування: Основна мова програмування для Java - це Java, але також існують інші мови, такі як Kotlin, що можуть бути використані на платформі JVM.	Мова програмування: Основною мовою для .NET є C#, але також підтримуються F# і VB.NET.

Продовження табл. 3.1

Java	.NET (C#)
Велика спільнота та екосистема: Java має велику та активну спільноту розробників, багато бібліотек і фреймворків.	Інтеграція з екосистемою Microsoft: .NET має гарну інтеграцію з іншими продуктами Microsoft, такими як Microsoft Azure, Visual Studio і Windows Server.
Використання в різних сферах: Java добре підходить для веб-розробки, корпоративних застосунків, мобільних додатків, інтернету речей (IoT) та інших сфер.	Велика бібліотека: .NET має обширну бібліотеку класів, включаючи ASP.NET для веб-розробки, Entity Framework для роботи з базами даних, і WPF для створення настольних додатків.

Обираючи між Java і .NET, важливо враховувати конкретні потреби проекту, наявність навичок розробників і інші фактори. Обидві платформи мають свої переваги і застосування в різних областях розробки.

Обґрунтування вибору платформи розробки .NET може бути обумовлене рядом переваг і факторів, які варто враховувати при розгляді цієї платформи для вашого проекту:

- Мови програмування: .NET підтримує кілька мов програмування, включаючи C#, F#, і VB.NET. Це дає можливість вибору мови, яка найкраще підходить для вашого проекту та команди розробників.
- Кросплатформенність: Із введенням .NET Core (пізніше перейменованим в .NET 5 і .NET 6) .NET став кросплатформеним рішенням, що дозволяє розробляти програми для Windows, macOS і Linux. Це важливо, якщо вам потрібно підтримувати різні операційні системи.
- Велика бібліотека: .NET має велику бібліотеку класів (Base Class Library), яка містить численні готові рішення для різних завдань, що може значно спростити розробку.

- Висока продуктивність: .NET відомий своєю високою продуктивністю та оптимізаціями. Він підтримує Just-In-Time компіляцію та може бути швидким у виконанні.
- Інтеграція з екосистемою Microsoft: Якщо ви вже використовуєте інші продукти Microsoft, такі як Microsoft Azure або Visual Studio, то вибір .NET може бути логічним, оскільки він легко інтегрується з іншими рішеннями Microsoft.
- Розвинута спільнота: .NET має велику та активну спільноту розробників, що дозволяє швидко отримувати допомогу, знаходити рішення та спільно розвивати ресурси.
- Підтримка майбутнього розвитку: Microsoft активно розвиває платформу .NET, випускаючи нові версії та функціональні оновлення. Це означає, що ваше програмне забезпечення може бути актуальним у майбутньому.

Обґрунтування вибору платформи розробки .NET буде залежати від конкретних потреб вашого проекту, але вищезазначені фактори свідчать про те, що .NET є потужною та гнучкою платформою для багатьох видів програмних розробок.

Обґрунтування мови програмування для реалізації програмного забезпечення магазину антикваріату залежить від кількох факторів, таких як вимоги проекту, експертиза розробників, наявні ресурси та інші технічні чинники. В таблиці 3.2 наведено порівняння деяких популярних мов програмування за різними критеріями

Таблиця 3.2 – Порівняння популярних мов програмування

Мова програмування	Використання	Впровадження	Продуктивність	Спільнота
Java	Універсальна	JVM	Середня	Велика
Python	Універсальна	Інтерпретація	Нижча	Велика
C#	Microsoft	.NET	Висока	Велика

Виходячи з вище сказаного вибір мови програмування C# для розробки системи для управління обігом антикваріату може бути обґрунтований декількома причинами:

- Платформа .NET: C# є основною мовою програмування для платформи .NET, яка надає широкий набір інструментів і бібліотек для розробки різноманітних програмних рішень. Це включає можливості роботи з базами даних, мережами, веб-розробкою та багато іншого. Використання C# та платформи .NET дозволяє ефективно використовувати існуючі ресурси та інфраструктуру;
- Об'єктно-орієнтоване програмування: C# є мовою, яка підтримує об'єктно-орієнтоване програмування (ООП), що сприяє структуруванню програмного коду та полегшує розробку та обслуговування складних систем. ООП дозволяє моделювати реальні об'єкти та взаємодію між ними, що є корисним для розробки магазину антикваріату, де можуть бути різноманітні сутності, такі як товари, клієнти, замовлення тощо;
- Широке співтовариство розробників: C# має широке співтовариство розробників, що сприяє доступності ресурсів, підтримки та знань. Існує велика кількість документації, підручників, форумів та інших ресурсів, які допоможуть вирішити проблеми та отримати поради в процесі розробки;
- Інтеграція з іншими технологіями Microsoft: Якщо магазин антикваріату використовує інші технології Microsoft, такі як Microsoft SQL Server для бази даних або ASP.NET для веб-розробки, використання C# забезпечить більшу сумісність та інтеграцію з цими технологіями;
- Розширені можливості: C# має ряд розширених можливостей, таких як LINQ (Language-Integrated Query), який спрощує роботу з даними, асинхронне програмування, підтримку паралельного програмування та багато іншого. Ці можливості можуть бути корисними для оптимізації та поліпшення продуктивності системи.

Аналіз та обґрунтування вибору інструментального програмного засобу

є важливим етапом у розробці програмного забезпечення. При виборі інструментального програмного засобу для розробки програмного забезпечення, можна враховувати такі фактори:

- Функціональність: Вибраний інструмент повинен мати всі необхідні функції та можливості для розробки програмного забезпечення. Наприклад, можливість моделювання бази даних, створення діаграм класів та діаграм станів, генерація коду тощо;
- Легкість використання: Інструмент повинен бути зручним у використанні, мати інтуїтивний інтерфейс користувача та добре документовані можливості. Це дозволить розробникам швидко оволодіти інструментом і ефективно використовувати його в процесі розробки;
- Підтримка мов програмування: Якщо у вашому проекті використовуються певні мови програмування, важливо переконатися, що вибраний інструмент підтримує ці мови програмування. Наприклад, якщо ви плануєте використовувати мову C#, інструмент повинен мати підтримку для роботи з цією мовою;
- Інтеграція з іншими інструментами: Якщо ви вже використовуєте інші інструменти або середовища розробки, такі як Visual Studio, важливо переконатися, що вибраний інструмент може інтегруватися з ними. Це дозволить забезпечити зручний і безперешкодний обмін даними та роботу з іншими інструментами;
- Підтримка та документація: Важливо враховувати рівень підтримки виробника і доступність документації. Якщо виникають проблеми або питання, важливо мати доступ до швидкої підтримки виробника або документації, яка допоможе вирішити проблеми та зрозуміти функціональні можливості інструменту.

Порівняльна характеристика баз даних допоможе вам зрозуміти різницю між різними типами баз даних. Ось декілька основних характеристик, за якими можна порівнювати бази даних:

Тип даних:

- Реляційні бази даних: Використовують таблиці для збереження даних, де кожен запис представлений рядком, а кожен стовпець - це поле або атрибут даних.
- NoSQL бази даних: Можуть використовувати різні структури для збереження даних, такі як документи, ключ-значення, графи, стовпчаті тощо.

Модель даних:

- Реляційні бази даних використовують структуровані схеми з фіксованими таблицями та зв'язками між ними.
- NoSQL бази даних надають більше гнучкості, дозволяючи зберігати структуровані, напівструктуровані або невструктуровані дані.

Мови запитів:

- Реляційні бази даних використовують SQL (Structured Query Language) для взаємодії з даними.
- NoSQL бази даних можуть використовувати різні мови запитів або навіть API для доступу до даних.

Масштабованість:

- Реляційні бази даних можуть бути обмежені у масштабуванні і вимагати складних конфігурацій для розширення.
- NoSQL бази даних, зазвичай, більше підходять для горизонтального масштабування (додавання нових серверів) через розподілені системи.

Доступ до даних:

- Реляційні бази даних зазвичай використовують транзакції для гарантування цілісності даних.
- NoSQL бази даних можуть мати інші моделі доступу до даних, такі як "елементарність" (кожен елемент доступний окремо) або "полегшена консистентність".

Призначення:

- Реляційні бази даних часто використовуються для транзакційних систем і бізнес-застосунків, де важлива цілісність даних.

- NoSQL бази даних можуть бути корисні для великих обсягів даних, аналітики, веб-застосунків і сценаріїв, де змінливість схеми важлива.

Підтримка:

- Реляційні бази даних мають довгу історію та багато різних систем, таких як MySQL, PostgreSQL, Oracle, SQL Server, тощо.
- NoSQL бази даних включають MongoDB, Cassandra, Redis, Neo4j, і багато інших, кожна з яких призначена для конкретних завдань.

Дуплікація даних:

- У реляційних базах даних дуплікація даних зазвичай уникається, оскільки дані зазвичай зберігаються в нормалізованій формі.
- У NoSQL базах даних дуплікація даних може бути допустимою для підвищення продуктивності і доступності.

Ось порівняльна таблиця 3.6 основних характеристик чотирьох популярних реляційних баз даних: MySQL, PostgreSQL, Oracle і SQL Server. Зазначені характеристики можуть допомогти вам визначити, яка з них найкраще відповідає вашим потребам.

Таблиця 3.6 – Порівняльний аналіз основних характеристик реаліційних баз даних

Характеристика	MySQL	PostgreSQL	Oracle	SQL Server
Ліцензія	Open Source	Open Source	Proprietary	Proprietary
Підтримка SQL	Так	Так	Так	Так
Підтримка транзакцій	Так	Так	Так	Так
ACID-сумісність	Так	Так	Так	Так
Географічна реплікація	Так	Так	Так	Так
Підтримка JSON	Так	Так	Так	Так
Підтримка XML	Так	Так	Так	Так

Продовження табл. 3.6

Характеристика	MySQL	PostgreSQL	Oracle	SQL Server
Підтримка геоданих	Так	Так	Так	Так
Підтримка графових даних	За допомогою додаткових розширень	Так	Так	За допомогою додаткових розширень
Розподілені бази даних	Так	Так	Так	Так
Резервне копіювання	Так	Так	Так	Так
Повнотекстовий пошук	Так	Так	Так	Так
Підтримка географічних запитів	За допомогою додаткових розширень	Так	Так	За допомогою додаткових розширень
Інтеграція з програмними мовами	Різні мови, включаючи C++, Java, Python	Різні мови, включаючи C++, Java, Python	Різні мови, включаючи C++, Java, Python, .NET, PL/SQL	C++, Java, Python, .NET, Ruby, PHP, Perl, PowerShell

Також порівняльна таблиця 3.7 основних характеристик декількох популярних NoSQL баз даних.

Таблиця 3.7 – Порівняльний аналіз основних характеристик NoSQL баз даних

Характеристика	MongoDB	Cassandra	Redis	Cosmos DB
Тип бази даних	Документоорієнтована	Ключ-значення	Ключ-значення	Мультимодельна
Модель даних	Багатоструктурна	Багатоструктурна	Багатоструктурна	Багатоструктурна
Спосіб запиту	JSON, BSON, деякі інші формати	CQL (Cassandra Query Language)	N/A	SQL (T-SQL), Gremlin, MongoDB Query Language та інші
Підтримка ACID	За допомогою транзакцій (у деяких версіях)	За допомогою транзакцій (у деяких версіях)	Так	За допомогою транзакцій (у деяких версіях)
Геореплікація	Так	Так	Так	Так
Підтримка геоданих	Так	Так	Ні	Так
Горизонтальне масштабування	Так	Так	Ні	Так
Кешування даних	Так	Так	Так	Так
Підтримка розширень	Так	Так	Так	Так
Використання	Веб-застосунки, аналітика даних	Інтернет-розробка, аналітика даних	Кешування, сесії, реального часу	Веб-застосунки, аналітика даних, соціальні мережі, аналітика даних

Отже виходячи з вищенаведених дани найкращим варіантом буде MySQL базаданих для основних даних системи і Cosmos DB NoSQL базу даних для транзакцій і даних для аналітики.

3.2 Розробка серверної частини та базових модулів

Для пришвидшення розробки було вирішено використати методи автоматичної генерації коду на основі розроблених моделей та діаграм класів. Для генерації коду на мові C# у Visual Paradigm використовується інструмент Instant Generator, що знаходить на панелі інструментів у секції коду (рисунок 3.1).

Найголовнішим етапом це вірно налаштувати генерацію коду. В цьому випадку потрібно вибрати наступні дії: вибрати модель з якої буде генеруватися код, вибрати мову програмування, розташування на комп'ютері для створюваних файлів та додаткові параметри, що залежать від обраної мови програмування. Вікно налаштування генерації з введеними параметрами зображено на рисунку 3.2.

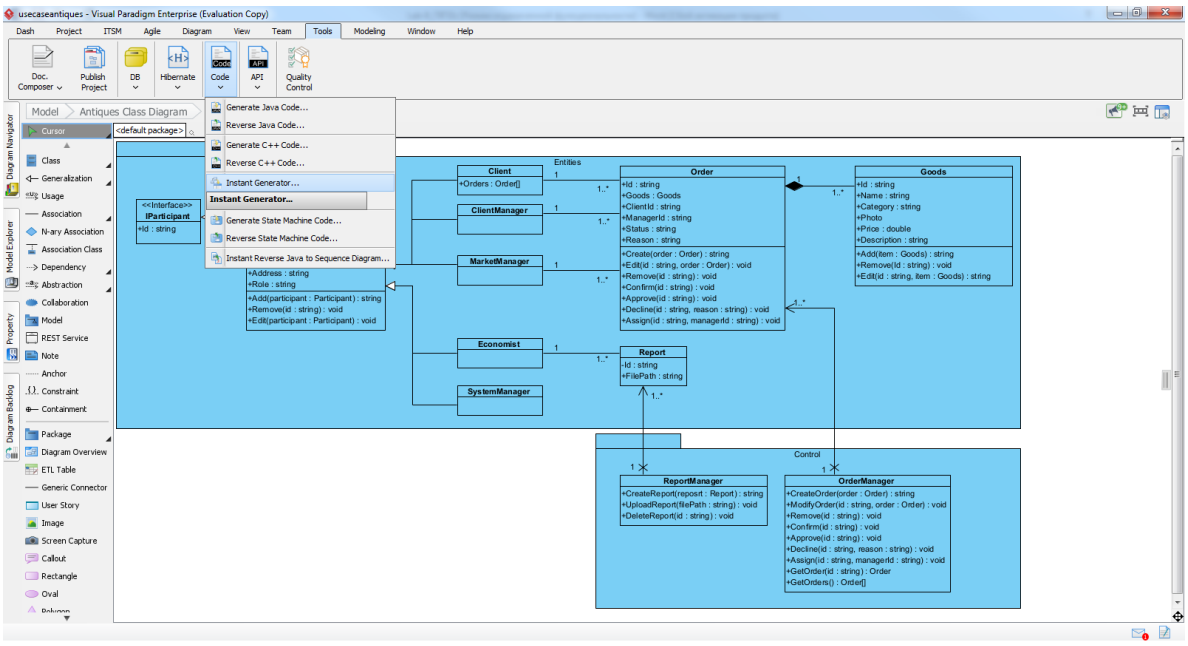


Рисунок 3.1 – Інструмент Instant Generator

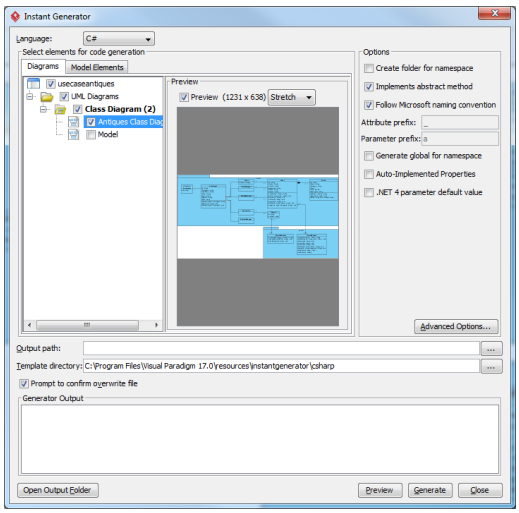


Рисунок 3.2 – Вікно налаштувань генерації коду

Також можна визначити елементи для яких буде згенеровано код і переглянути як саме буде згенеровано код. Відповідні дії зображені на рисунку 3.3 і рисунку 3.4 відповідно.

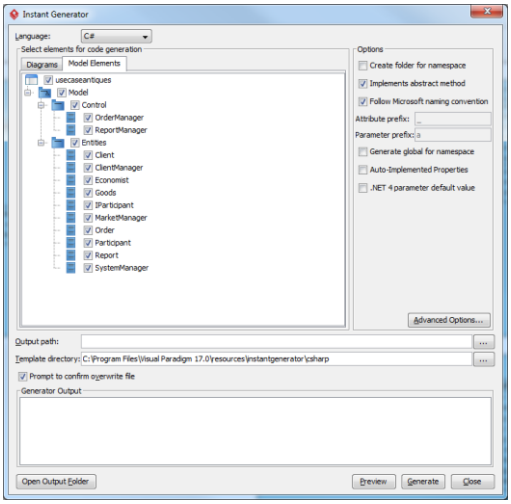


Рисунок 3.3 – Вікно налаштувань генерації коду

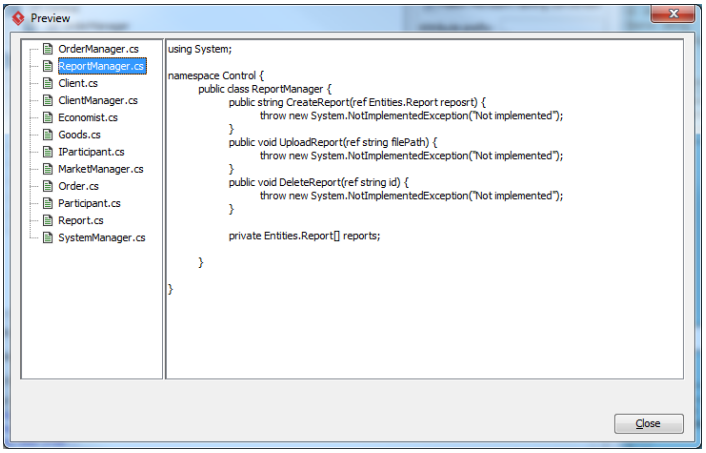


Рисунок 3.3 – Вікно перегляду генерації коду

Після всіх налаштувань можна перейти до генерації доку. Завершення генерації коду зображено на рисунку 3.3.

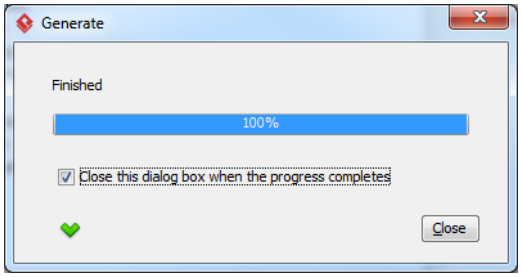


Рисунок 3.3 – Повідомлення про завершення генерацію коду

Результат генерації коду зображено на рисунку 3.4.

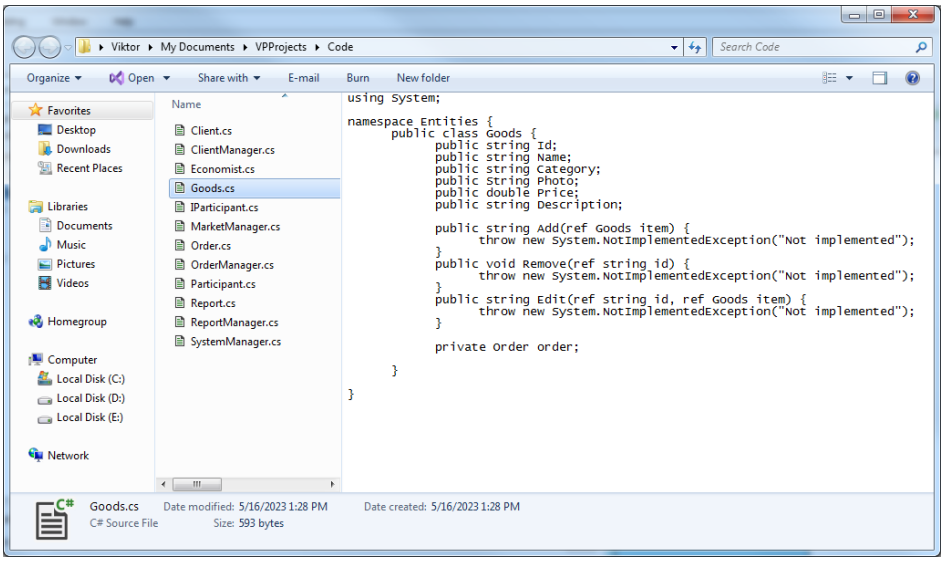


Рисунок 3.4 – Результат генерації коду

На основі створених діаграм класів за допомогою інструментів CASE-засобу Visual Paradigm було згенеровано програмний код. Отриманий програмний код буде використаний для подальшої розробки програмного забезпечення.

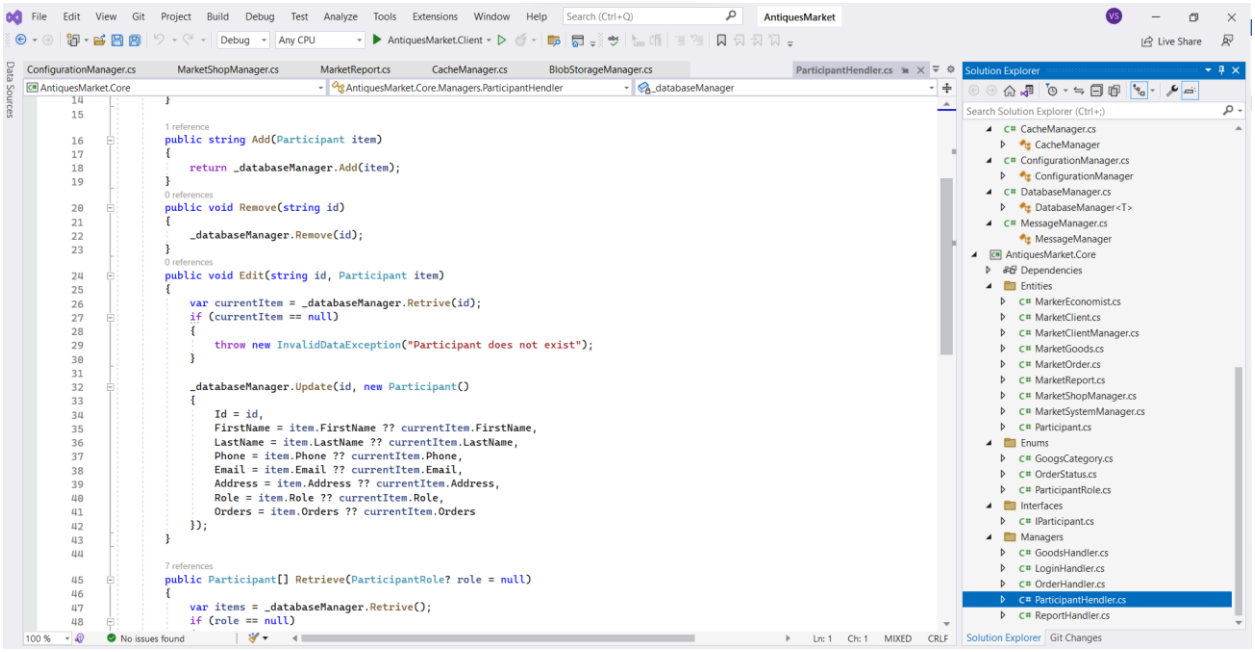


Рисунок 3.5 – Фрагмент реалізації

Програмне забезпечення було реалізоване на основі спроектованих класів і згенерованого коду. На рисунку 3.5 зображено фрагмент реалізації на основі згенерованого коду.

3.3 Розробка користувацького інтерфейсу

Розробка користувацького інтерфейсу (UI) є важливою складовою будь-якого програмного продукту, оскільки це та частина, з якою користувачі взаємодіють напряму. Ефективний інтерфейс робить програму більш зручною та доступною для користувачів. Нижче наведено кроки та рекомендації для розробки користувацького інтерфейсу:

Збір вимог користувача. Розпочніть із збору вимог до інтерфейсу від користувачів. Проведіть опитування, співбесіди та аналіз конкурентів, щоб зрозуміти, як користувачі очікують, що буде включено до інтерфейсу.

Проектування інтерфейсу. Створіть інтерфейс на основі зібраних вимог. Визначте структуру сторінок (для веб-додатків) або вікон (для настольних додатків). Розмістіть елементи керування, кнопки, поля введення та інші компоненти так, щоб інтерфейс був зручним для користувача.

Візуальний дизайн. Розробіть візуальний дизайн інтерфейсу,

включаючи вибір кольорової палітри, шрифтів, стилів і графіки. Забезпечте, щоб дизайн був відповідним бренду та цільовій аудиторії.

Розробка інтерфейсу. Використовуйте технології та інструменти розробки для створення інтерфейсу. Для веб-додатків це може бути HTML, CSS і JavaScript, а для настольних додатків - фреймворки та бібліотеки, такі як WPF або WinForms для платформи .NET.

Тестування. Перевірте інтерфейс на наявність помилок, таких як баги в інтерфейсі, незручності в користуванні, несумісності на різних пристроях і браузерах (для веб-додатків). Виправляйте знайдені проблеми.

Адаптація до різних пристроїв. Забезпечте, щоб інтерфейс був адаптований до різних типів пристроїв, включаючи комп'ютери, смартфони, планшети і інші. Використовуйте адаптивний дизайн або розроблюйте окремі версії для різних пристроїв.

Забезпечення доступності.Розгляньте питання доступності для користувачів з обмеженими можливостями. Додайте можливість збільшення шрифту, адекватну роботу з читальними програмами і т.д.

Оптимізація продуктивності. Зробіть інтерфейс максимально продуктивним та ефективним. Зменшуйте завантаження сторінок або вікон, використовуйте кешування для швидкості завантаження інформації.

Тестування з користувачами. Залучіть кількох користувачів до тестування інтерфейсу. Отримайте їхні відгуки та спостереження для подальших вдосконалень.

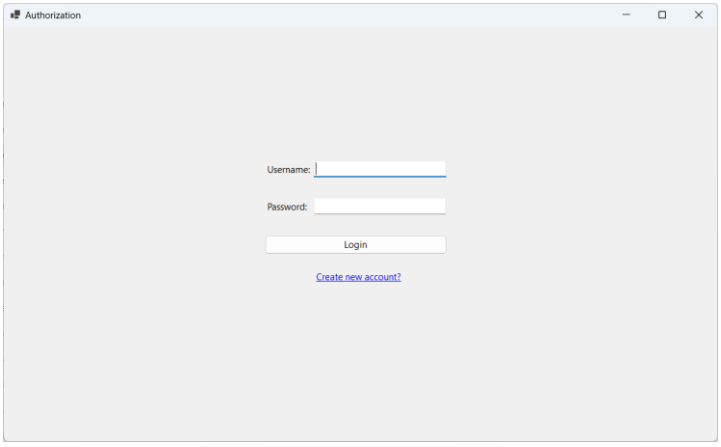


Рисунок 3.6 – Вікно авторизації

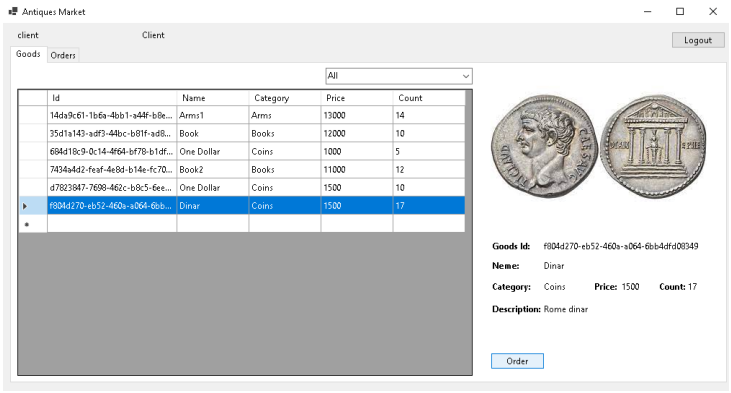


Рисунок 3.7 – Вікно перегляду товарів магазину антикваріату

Постійне вдосконалення. Інтерфейс не є статичним елементом, і його потрібно постійно вдосконалювати відповідно до змін вимог і потреб користувачів.

Розробка користувацького інтерфейсу - це ітеративний процес, і важливо слухати фідбек користувачів та вдосконалювати інтерфейс для забезпечення найкращого досвіду користувачів.

3.4 Висновки

В ході варіантного аналізу та обґрунтування вибору способів реалізації програмного засобу було виконано низку завдань щодо вибору платформи проектування і власне реалізації програмного продукту.

Був проведений аналіз і обґрунтування вибору інструментальних засобів для проектування і розробки програмного забезпечення. в результаті чого було обрано інструментальний засіб Visual Paradigm для проектування системи і Visual Studio для реалізації програмного забезпечення.

Вибір мови програмування C# був обґрунтований його потужним функціоналом, широкою підтримкою та доступністю для розробників.

Була здійснена реалізація програмного забезпечення на основі згенерованого коду, з використанням інтегрованого середовища розробки Visual Studio.

4 ТЕСТУВАННЯ ПРОГРАМНОГО ДОДАТКУ

4.1 Аналіз методів і засобів тестування

Аналіз методів і засобів тестування є важливим етапом в процесі розробки програмного продукту. Вибір правильних методів і інструментів допомагає забезпечити високу якість програми та позбутися потенційних помилок. Ось кілька методів і засобів тестування, які можна аналізувати:

Методи тестування:

- Модульне тестування (Unit Testing): Цей метод включає в себе тестування окремих компонентів або модулів програми. Зазвичай використовуються фреймворки, такі як JUnit для Java або NUnit для .NET, для написання і запуску модульних тестів.
- Інтеграційне тестування (Integration Testing): Під час інтеграційного тестування перевіряється взаємодія між різними модулями або компонентами системи. Може бути використано різні техніки, такі як чорний ящик або білий ящик.
- Системне тестування (System Testing): Тестування всієї системи як єдної одиниці. Може включати функціональне тестування, навантажувальне тестування, тестування продуктивності і т. д.
- Автоматизоване тестування (Automated Testing): Використання скриптів і спеціальних програмних інструментів для виконання тестів без прямого втручання користувача.

Інструменти тестування:

- Selenium: Використовується для автоматизованого тестування веб-додатків. Дозволяє відтворювати взаємодію зі сторінками веб-сайту.
- JUnit і TestNG: Фреймворки для модульного тестування в Java, що допомагають створювати та виконувати тестові сценарії.
- JUnit: Фреймворк для модульного тестування в Java.
- NUnit: Фреймворк для модульного тестування в .NET.

- Postman: Використовується для тестування API. Дозволяє створювати та виконувати запити до сервера.
- Jenkins та Travis CI: Ці інструменти використовуються для автоматизації процесу неперервної інтеграції (CI) і неперервної доставки (CD), включаючи автоматичне виконання тестів під час кожного злиття коду.

Техніки тестування:

- Функціональне тестування: Перевірка того, чи виконує програма свої функції відповідно до специфікацій.
- Навантажувальне тестування: Визначення та валідація відповідності програми певному навантаженню та показникам продуктивності.
- Тестування безпеки: Виявлення та усунення потенційних вразливостей, що можуть призвести до порушення безпеки програми.
- Тестування користувацького досвіду (Usability Testing): Оцінка зручності інтерфейсу для користувачів та виявлення незручностей в користуванні.
- Тестування з використанням реальних даних (Data-Driven Testing): Використання реальних даних для тестування програми в реальних умовах.

Аналіз методів і засобів тестування повинен враховувати конкретні потреби вашого проекту, рівень складності програми і доступні ресурси. Правильно підібрані методи та інструменти допоможуть виявити та виправити помилки, забезпечуючи високу якість програмного продукту.

4.2 Етапи тестування програмного додатку

Етапи тестування програмного додатку можуть варіюватися залежно від конкретного проекту та методології розробки, але основні кроки тестування включають в себе наступне.

Планування тестування:

- Визначення цілей та об'єктів тестування.

- Створення тестової стратегії та плану тестування.
- Вибір методів, інструментів і ресурсів для тестування.
- Розроблення розкладу тестування і виділення необхідних ресурсів.

Аналіз вимог:

- Ретельний огляд вимог до програмного додатку, включаючи функціональні та нефункціональні вимоги.
- Розроблення тестових сценаріїв та тест-кейсів на основі вимог.

Підготовка тестового середовища:

- Встановлення та налаштування необхідних середовищ (наприклад, тестових серверів або імітації зовнішніх систем).
- Підготовка тестових даних.

Виконання тестування:

- Виконання тестових сценаріїв та тест-кейсів на програмному додатку.
- Реєстрація результатів тестування, включаючи виявлені помилки.

Дебагінг і виправлення помилок:

- Аналіз та відлагодження виявлених помилок.
- Перевірка виправлених помилок для підтвердження їхнього виправлення.

Регресійне тестування:

- Перевірка програми на наявність нових помилок після внесення змін (зазвичай, після виправлення виявлених помилок).
- Виконання попередньої функціональності для підтвердження її цілісності.

Тестування навантаження:

- Проведення тестів на продуктивність для оцінки швидкості та завантажуваності програмного додатку.
- Виявлення та усунення проблем продуктивності.

Тестування безпеки:

- Перевірка програмного додатку на вразливості і можливість злому.
- Забезпечення відповідності стандартам безпеки.

Автоматизоване тестування:

- Розробка та виконання автоматизованих тестів для автоматичного виявлення помилок та перевірки функціональності.

Завершення тестування:

- Формування звіту про тестування, включаючи результати та обсяг виконаних тестів.
- Підготовка до випуску програмного додатку.

Повторний аналіз та оцінка:

- Аналіз результатів тестування для визначення якості програмного додатку.
- Перевірка відповідності програми вимогам і стандартам якості.

Завершення тестування:

- Формування звіту про тестування, включаючи результати та обсяг виконаних тестів.
- Підготовка до випуску програмного додатку.

Це загальні етапи тестування програмного додатку, але точні кроки та підходи можуть різнитися залежно від методології розробки (наприклад, Agile або Waterfall) та специфіки проекту. Важливо враховувати, що тестування - це постійний процес, і воно може включати в себе ітеративні аспекти, особливо у розробці програмного забезпечення методологією Agile.

Після реалізації програми, вона може бути протестована за допомогою різних методів тестування, щоб переконатися, що вона працює правильно і відповідає заданим вимогам.

Кожна окрема частина програми, така як класи, методи або функції, може бути протестована окремо, щоб перевірити їх правильність і працездатність. Було проведено модульне тестування окремих модулів

системи за допомогою юніт тестів.

Після модульного тестування окремі модулі програми можуть бути поєднані та протестовані як єдина система. За допомогою інтеграційних тестів було проведено тестування взаємодії між різними модулями, передачу даних, зв'язки та інші аспекти інтегрованої системи. На рисунку 3.6 і рисунку 3.7 зображено результати інтеграційного тестування.

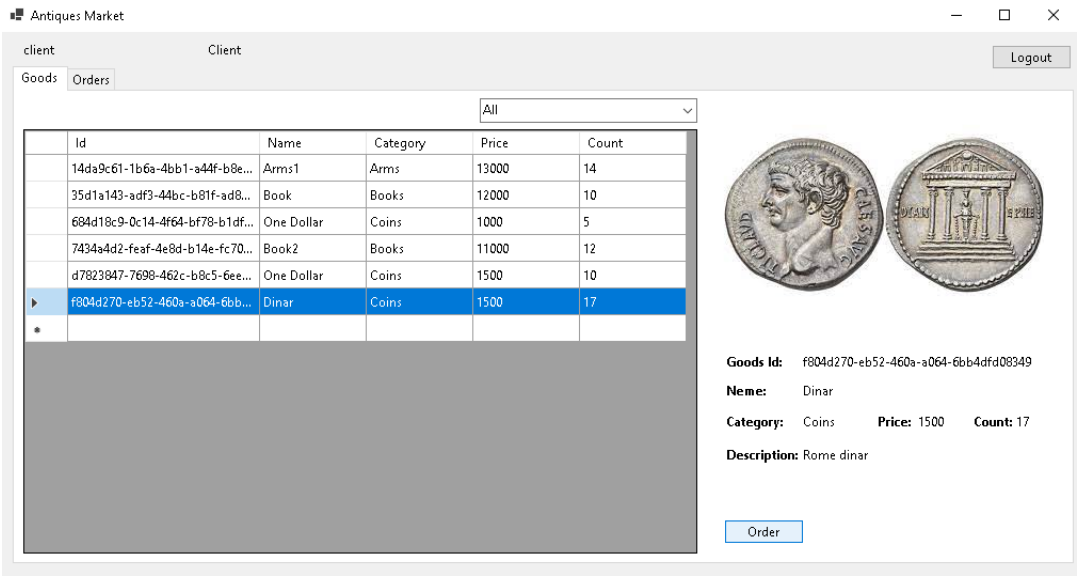


Рисунок 4.1 – Вікно перегляду товарів магазину антикваріату

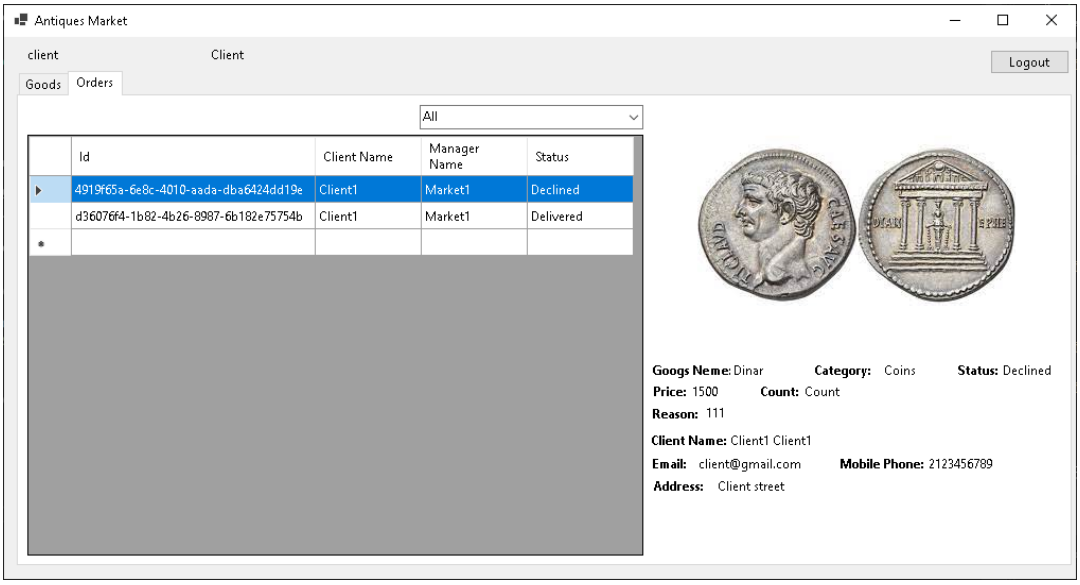


Рисунок 4.2 – Вікно перегляду історії замовлень

Звісно під час тестування було виявлено виявлено деякі помилки, які були виправлені. Після чого було проведене повторне тестування програмного продукту.

4.3 Висновки

Висновки щодо тестування системи управління обігом антикваріату мають на меті підсумувати результати тестування та оцінити якість системи. Отже після реалізації програмного забезпечення було проведене тестування, яке включало модульне тестування, інтеграційне тестування та системне тестування.

Під час тестування було перевірено відповідність програмного забезпечення вимогам, правильність його функціонування, а також здатність обробляти помилки.

Реалізація програми була успішно протестована і підтверджує її правильну роботу та функціональність.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення комерційного та технологічного аудиту є оцінювання комерційного потенціалу розробки методів і засобів системи управління обігом антикваріату.

Для проведення технологічного аудиту було залучено 3-х незалежних експертів Вінницького національного технічного університету к.т.н., доцента Майданюк В. П., к.т.н., доцента Ракитянську Г. Б., к.т.н., доцента Рейди О.М з кафедри програмного забезпечення.

Технологічний аудит проведемо з використанням таблиці 5.1 [22], де за п’ятибальною шкалою використовуючи 12 критеріїв оцінимо комерційний потенціал. В таблиці 5.1 наведені рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка:

Таблиця 5.1 – Рекомендовані критерії оцінювання комерційного потенціалу розробки та їх можлива бальна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
#	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовження табл. 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
#	0	1	2	3	4
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років

Продовження табл. 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
#	0	1	2	3	4
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Таблиця 5.2 – Рівні комерційного потенціалу розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0-10	Низький
11-20	Нижче середнього
21-30	Середній
31-40	Вище середнього
41-48	Високий

Таблиця 5.3 – Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1. Майданюк В. П.	2. Ракитянська Г. Б.	3. Рейда О. М.
	Бали, виставлені експертами:		
1	3	3	3
2	2	2	2
3	4	4	4
4	4	4	4
5	2	3	3
6	2	3	3
7	2	3	2
8	3	3	2
9	3	3	2
10	4	4	4
11	4	4	4
12	4	4	4
Сума балів	$СБ_1$=37	$СБ_2$=40	$СБ_3$=37
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{37+40+37}{3} = 38$		

Середньоарифметична сума балів, розрахована на основі висновків експертів склала 38 бали, що згідно таблиці 5.2 вважається, що рівень комерційного потенціалу проведених досліджень є вище середнього.

Основними клієнтами системи управління обігом антикваріату є як кінцеві продавці та покупці антикваріату, так і підприємства, онлайн аукціони, які можуть інтегрувати продукт з їх системою. Сервіс буде надаватися як SaaS – «Software as a service» у вигляді веб-додатку або веб-служби як форму хмарних обчислень. Відповідно це вимагає враховувати в вартість підтримку інфраструктури – бази даних, сервера обчислень, шини даних, тощо.

На українському ринку не існує повноцінного аналога, який би використовувався як система управління обігом антикваріату. Існує кілька аналогів програмного забезпечення, які можуть бути використані як магазин антикваріату.

В якості аналога для розробки було обрано онлайн-платформу Violity. Основними недоліками є те що ця система являється онлайн аукціоном і не зовсім відповідає системі для управління обігом антикваріату.

Таблиця 5.4 – Основні параметри нової розробки та товару-конкурента

Показник	Варіанти		Відносний показник якості	Коефіцієнт вагомості параметра
	Базовий (товар-конкурент)	Новий (інноваційне рішення)		
1	2	3	4	5
Точність обмежень для різних мов програмування, %	60	90	1.5	25%
Оформлення заказу, бали	7	10	1.6	30%
Середній час відповіді на запит (менше – краще), мс	600	250	1.79	20%
Відмовістійкість, %	90	90	1	15%
Використання ресурсів, %	30	20	1	10%

Проведемо оцінку якості нової розробки порівняно з аналогом. В

таблиці 5.4 наведені основні техніко-економічні показники аналога і нової розробки. Проведемо оцінку якості продукції, яка є найефективнішим засобом забезпечення вимог споживачів та порівняємо її з аналогом.

Визначимо відносні одиничні показники якості по кожному параметру за формулами (5.1) та (5.2) і занесемо їх у відповідну колонку табл. 5.5.

$$q_i = \frac{P_{Hi}}{P_{Бi}} \tag{5.1}$$

або

$$q_i = \frac{P_{Бi}}{P_{Hi}} \tag{5.2}$$

де P_{Hi}, $P_{Бi}$ – числові значення i-го параметру відповідно нового і базового виробів.

$$q_1 = \frac{90}{60} = 1.5;$$

$$q_2 = \frac{10}{7} = 1.4;$$

$$q_3 = \frac{600}{250} = 2.4;$$

$$q_4 = \frac{90}{90} = 1;$$

$$q_5 = \frac{30}{20} = 1.5;$$

Відносний рівень якості нової розробки визначаємо за формулою:

$$К_{я.в.} = \sum_{i=1}^{n} q_i \cdot \alpha_i, \tag{5.3}$$

$$К_{я.в.} = 1.5 * 0.25 + 1.4 * 0.30 + 2.4 * 0.2 + 1 * 0.15 + 1.5 * 0.1 = 1.58$$

Відносний коефіцієнт показника якості нової розробки більший одиниці, отже нова розробка незначно якісніше базового товару-конкурента.

Загальний показник конкурентоспроможності інноваційного рішення (К) з урахуванням вищезазначених груп показників можна визначити за формулою:

$$K = \frac{I_{m.n.}}{I_{e.n.}}, \tag{5.4}$$

де $I_{m.n.}$ – індекс технічних параметрів; $I_{e.n.}$ – індекс економічних параметрів.

Індекс технічних параметрів є відносним рівнем якості інноваційного рішення.

Індекс економічних параметрів визначається за формулою (6.6)

$$I_{e.n.} = \frac{\sum_{i=1}^{n} P_{Hei}}{\sum_{i=1}^{n} P_{Бei}}, \tag{5.5}$$

де P_{Hei}, $P_{Бei}$ – економічні параметри (ціна придбання та споживання товару) відповідно нового та базового товарів.

$$\mathrm{I_{e.п.}} = \frac{5}{7} = 0.7$$

$$\mathrm{К} = \frac{1.58}{0.7} = 2.3$$

Зважаючи на розрахунки, можна зробити висновок, що нова розробка буде більш конкурентоспроможне, ніж конкурентний товар.

5.2 Прогнозування витрат на виконання науково-дослідної роботи

1. Витрати пов’язані на проведення науково-дослідних робіт можливо скласти в наступні статті

- Витрати на оплату праці;
- Витрати на матеріальні засоби (канцтовари, обладнання);
- Витрати на інструментальне та інфраструктурне програмне забезпечення;

Основна заробітна плата кожного із залучених осіб визначається за формулою (5.6)

$$З_О = \sum_{i=1}^{K} \frac{M_{ni} * t_i}{T_p} \text{ (грн)} \quad (5.6)$$

де M – місячний посадовий оклад конкретного розробника (інженера, дослідника, науковця тощо), грн.;

$Т_р$– число робочих днів в місяці; приблизно $Т_р = 22$;

t - кількість робочих днів роботи розробника.

Для розробки системи для управління обігом антикваріатом необхідно залучити: Project Manager, Solution Architect, Business Analyst, Software Engineer (Backend), Software Engineer (Client), QA Engineer. Посадові оклади, число днів роботи та витрати на компенсацію додаємо в таблицю 5.5.

Таблиця 5.5 - Компенсація спеціаліста в дослідницькій установі

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату грн.
Project Manager	120000	5454	110	599940
Solution Architect	150000	6818	30	204540
Business Analyst	40000	1818	30	54540
Software Engineer (Backend)	150000	6818	110	749980
Software Engineer (Client)	70000	3181	60	190860
QA Engineer	50000	2272	50	113600
Всього				1913460

2. Розрахунок додаткової заробітної плати робітників

Додаткова заробітна плата $З_д$ всіх робітників, які приймали участь в розробці нового технічного рішення розраховується як 10 - 15 % від основної заробітної плати робітників як премія.

На даному підприємстві додаткова заробітна плата начисляється в розмірі 10% від основної заробітної плати.

$$З_д = (З_о + З_р) * \frac{Н_{доп}}{100\%} \quad (5.7)$$

$$З_д = 0{,}1 * 1913460 = 191346 \text{ грн}$$

3. Нарахування на заробітну плату $Н_{ЗП}$ робітників, які брали участь у виконанні роботи, розраховуються за формулою (6.9):

$$З_н = (З_о + З_р + З_д) * \frac{Н_{ЗП}}{100} \text{ (грн)} \quad (5.8)$$

де $З_о$ – основна заробітна плата розробників, грн.;

$З_д$ – додаткова заробітна плата всіх розробників та робітників, грн.;

$Н_{ЗП}$ – ставка єдиного внеску на загальнообов'язкове державне соціальне страхування, %. Основна ставка єдиного внеску на загальнообов'язкове державне соціальне страхування на 2023 рік – 22 %, тоді:

$$З_н = (1913460 + 191346) * 0.22 = 463057 \text{ (грн)}$$

4. Витрати на комплектуючі вироби, які використовують при виготовленні одиниці продукції, розраховуються, згідно їх номенклатури, за формулою:

$$К = \sum_{i=1}^{n} Н_i \cdot Ц_i \cdot К_i, \quad (5.8)$$

де $Н_i$ – кількість комплектуючих *i*-го виду, шт.;

$Ц_i$ – покупна ціна комплектуючих *i*-го найменування, грн.;

$К_i$ – коефіцієнт транспортних витрат (1,1…1,15).

Потрібно закладати витрати на доставку у вигляді коефіцієнту транспортних витрат – 1.1. Інформацію про використані матеріали та комплектуючі подаємо у вигляді табл. 5.6.

Таблиця 5.6 – Матеріали, що використані на розробку

Найменування матеріалу	Ціна за одиницю, грн.	Витрачено	Вартість витраченого матеріалу, грн.
Дошка і набір маркерів	1200	1	1200
Папір	700	1	700
Накопичувач	1000	1	1000
Всього			2900
З врахуванням коефіцієнта транспортування			3190

5. Використані засоби програмного забезпечення включає в себе витрати на інструментальне програмне забезпечення – середа розробки, клієнти взаємодії, тощо, так і інфраструктурні інструменти. Розгортання і підтримка повноцінної власної інфраструктури дуже коштовна справа. Для розробки буде використовуватись базові надані ресурси Microsoft Azure. Вже для подальшого хостінгу буде використовуватись розширені підпистки, так як буде потреба в більшій кількості ресурсів. Балансову вартість програмного забезпечення розраховують за формулою:

$$В_{прг} = \sum_{i=1}^{k} Ц_{іпрг} \cdot C_{пргі} \cdot К_і, \quad (5.9)$$

де $Ц_{іпрг}$ – ціна придбання/використання одиниці програмного засобу цього виду, грн;

$С_{пргі}$ – кількість одиниць програмного забезпечення відповідного найменування, які придбані для проведення досліджень, шт.;

$К_і$ – коефіцієнт, що враховує інсталяцію, налагодження програмного засобу тощо ($К_і$ = 1,10…1,12).

k – кількість найменувань програмних засобів.

Отримані результати занесемо в таблицю 5.7.

Таблиця 5.7 – Витрати на використання програмних засобів по кожному виду

Найменування устаткування	Час використання, місяців	Ціна за місяць, грн	Вартість
Visual Studio Profesional	4	600	2400
Visual Paradigm	4	400	1600
Azure Plan for DevTestc	4	3000	12000
Azure DevOps Server	4	2500	10000

Продовження табл. 5.7

Найменування устаткування	Час використання, місяців	Ціна за місяць, грн	Вартість
Power BI Premium	4	2600	10400
Всього			36400

6. Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання роботи

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

$$А_{обл} = \frac{Ц_б}{Т_в} * \frac{t_{вик}}{12} \quad (5.10)$$

де $Ц_б$ – балансова вартість даного виду обладнання (приміщень), грн.;

$t_{вик}$ – час користування;

$Т_в$ – термін використання обладнання (приміщень), цілі місяці.

Для розробки продукту використовувався три персональні комп'ютери вартістю 100000 грн

Таблиця 5.8 – Амортизаційні відрахування по кожному виду обладнання

Найменування обладнання	Балансова вартість, грн	Строк корисного використанн, років	Термін використання обладнання, місяців	Амортизаційні відрахування, грн
Ноутбук (3)	300000	4	5	5000
Офісне приміщення	1 500 000	25	5	25000
Всього				30000

7. До статті «Паливо та енергія для науково-виробничих цілей» відносяться витрати на всі види палива й енергії, що безпосередньо використовуються з технологічною метою на проведення досліджень.

$$В_е = \sum_{i=1}^{n} \frac{W_{yt} \cdot t_i \cdot Ц_е \cdot К_{впі}}{\eta_i} \quad (5.11)$$

де W_{yt} – встановлена потужність обладнання на певному етапі розробки, кВт;

t_i – тривалість роботи обладнання на етапі дослідження, год;

$Ц_е$ – вартість 1 кВт-години електроенергії, грн;

$К_{впі}$ – коефіцієнт, що враховує використання потужності, $К_{впі} < 1$;

η_i – коефіцієнт корисної дії обладнання, $\eta_i < 1$.

При використані 3 комп'ютерів з сумарною потужністю 1.5 кВт витрачалася електроенергія з ціною 7.6 грн за кВт в 2023 році.

$$В_е = \frac{1.5 \cdot 880 \cdot 7.6 \cdot 0.45}{0.76} = 5940$$

До статті «Службові відрядження» належать витрати на відрядження штатних працівників, працівників організацій, які працюють за договорами цивільно-правового характеру, аспірантів, зайнятих розробленням досліджень, відрядження, пов'язані з проведенням випробувань машин та приладів, а також витрати на відрядження на наукові з'їзди, конференції, наради, пов'язані з виконанням конкретних досліджень. Витрати за статтею «Службові відрядження» розраховуються як 20…25% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{св} = (З_о + З_р) \cdot \frac{Н_{св}}{100} \quad (5.12)$$

де $Н_{св}$ – норма нарахування за статтею «Інші витрати».

$$В_{св} = 1913460 * 0.23 = 440096 \text{ грн}$$

До статті «Витрати на роботи, які виконують сторонні підприємства, установи і організації» належать витрати на проведення досліджень, що не можуть бути виконані штатними працівниками або наявним обладнанням організації, а виконуються на договірній основі іншими підприємствами,

установами і організаціями незалежно від форм власності та позаштатними працівниками.

Витрати за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації» розраховуються як 30…45% від суми основної заробітної плати дослідників та робітників за формулою:

$$В_{сп} = (З_о + З_р) \cdot \frac{Н_{сп}}{100} \quad (5.13)$$

де $Н_{пв}$ – норма нарахування за статтею «Витрати на роботи, які виконують сторонні підприємства, установи і організації».

$$В_{сп} = 1913460 * 0.4 = 765384 \text{ грн}$$

До статті «Інші витрати» належать витрати, які не знайшли відображення у зазначених статтях витрат і можуть бути віднесені безпосередньо на собівартість досліджень за прямими ознаками.

Витрати за статтею «Інші витрати» розраховуються як 50…100% від суми основної заробітної плати дослідників та робітників за формулою:

$$І_в = (З_о + З_р) \cdot \frac{Н_{ів}}{100} \quad (5.14)$$

де $Н_{ів}$ – норма нарахування за статтею «Інші витрати».

$$І_в = 1913460 * 0.7 = 1339422 \text{ грн}$$

Накладні (загальновиробничі) витрати Внзв охоплюють: витрати на управління організацією, оплата службових відряджень, витрати на утримання, ремонт та експлуатацію основних засобів, витрати на опалення, освітлення, водопостачання, охорону праці тощо. Накладні (загальновиробничі) витрати Внзв можна прийняти як 110% від суми основної заробітної плати розробників та робітників, які виконували дану МКНР, тобто:

$$В_{нзв} = (З_о + З_р) \cdot \frac{Н_{нзв}}{100}, \quad (5.15)$$

де $Н_{нзв}$ – норма нарахування за статтею «Інші витрати».

$$В_{нзв} = 1913460 * 1.1 = 2104806 \text{ грн}$$

Сума всіх попередніх статей витрат дає витрати, які необхідні для повноцінної розробки системи:

$$В = З_о + З_р + З_{дод} + З_н + М + К_в + В_{спец} + В_{прг} + А_{обл} + В_е + В_{св} +$$

$$В_{сп} + І_в + В_{нзв} \quad (5.16)$$

$$В = 1913460 + 191346 + 463057 + 3190 + 36400 + 30000 + 5940 + 440096 + 765384 + 1339422 + 2104806 = 7293101 \text{ грн}$$

Прогнозування загальних втрат ЗВ на виконання та впровадження комерційної розробки системи багатокритеріальної оцінки житлової нерухомості здійснюється за формулою:

$$ЗВ = \frac{В}{\eta}, \quad (5.17)$$

де η – коефіцієнт, який характеризує стадію виконання даної НДР.

Оскільки, робота знаходиться на стадії дослідного зразка, то коефіцієнт $\beta = 0.5$.

Звідси:

$$ЗВ = \frac{7293101}{0.5} = 14586202 \text{ грн.}$$

5.3 Розрахунок економічної ефективності науково-технічної розробки

У даному підрозділі кількісно спрогнозуємо, яку вигоду, зиск можна отримати у майбутньому від впровадження результатів виконаної наукової роботи. Розрахуємо збільшення чистого прибутку підприємства $\Delta П_i$, для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, за формулою (5.17):

$$\Delta П_i = \sum_1^n \left(\Delta Ц_o \cdot N + Ц_o \cdot \Delta N\right)_i \cdot \lambda \cdot \rho \cdot \left(1 - \frac{\upsilon}{100}\right) \quad (5.18)$$

де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році.

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки:

Цо – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки:

л – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість дорівнює 20%, а коефіцієнт *л* = 0,8333.

р – коефіцієнт, який враховує рентабельність продукту. р = 0,25;

х – ставка податку на прибуток. У 2023 році – 18%.

Припустимо, очікувана комісія від реставрації і реалізації антикварного виробу– 30%, середня вартість – ~ 50000 грн. Тобто середня комісія за товар 15000 грн. Припустимо середня ціна зросте на 10000 грн., тобто на 3000 грн. комісії. І в перший рік 3000 товарів реалізується. Кількість реалізованих товарів в перший рік зросте на 2000, в другий на 4000, і в третій на 6000.

$$\Delta П_1 = [3000 \cdot 3000 + (15000 + 3000) \cdot 2000] \cdot 0{,}833 \cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) =$$

$$= 11058075 = 11 \text{ млн}$$

$$\Delta П_2 = [3000 \cdot 3000 + (15000 + 3000) \cdot (2000 + 4000)] \cdot 0{,}833 \cdot 0{,}25$$

$$\cdot \left(1 + \frac{18}{100}\right) = 28751000 = 28.8 \text{ млн}$$

$$\Delta П_3 = [3000 \cdot 3000 + (15000 + 3000) \cdot (2000 + 4000 + 6000)] \cdot 0{,}833$$

$$\cdot 0{,}25 \cdot \left(1 + \frac{18}{100}\right) = 55290380 = 55.3 \text{ млн}$$

5.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахуємо основні показники, які визначають доцільність

фінансування наукової розробки певним інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Розрахуємо величину початкових інвестицій PV, які потенційний інвестор має вкласти для впровадження і комерціалізації науково-технічної розробки.

$$PV = k_{\text{інв}} \cdot \text{ЗВ} \tag{5.19}$$

$k_{\text{інв}}$ – коефіцієнт, що враховує витрати інвестора на впровадження науково-технічної розробки та її комерціалізацію. Це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо ($k_{\text{інв}}$ = 2…5).

$$PV = 2 \cdot 14586202 = 29172404$$

Розрахуємо абсолютну ефективність вкладених інвестицій $E_{абс}$ згідно наступної формули:

$$E_{абс} = (ПП - PV) \tag{5.20}$$

де ПП – приведена вартість всіх чистих прибутків, що їх отримає підприємство від реалізації результатів наукової розробки, грн.;

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, \tag{5.21}$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДЦКР, грн.;

Т – період часу, протягом якою виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,2;

t – період часу (в роках).

$$ПП = \frac{11058075}{(1+0{,}2)^1} + \frac{28751000}{(1+0{,}2)^2} + \frac{55290380}{(1+0{,}2)^3}$$

$$= 9215063 + 19965972 + 31996748 = 61177783 \text{ грн.}$$

$$Е_{абс} = (61177783 - 29172404) = 32005379 \text{ грн.}$$

Оскільки $E_{абс} > 0$ то вкладання коштів на виконання та впровадження результатів НДДКР може бути доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[T_ж]{1 + \frac{E_{абс}}{PV}} - 1, \qquad (5.22)$$

$T_ж$ – життєвий цикл наукової розробки, роки.

$$Е_в = \sqrt[з]{1 + \frac{32005379}{29172404}} - 1 = \sqrt[3]{1 + 1.1} - 1 = 1.28 - 1$$

$$Е_в = 0.28 = 28\ \%$$

Визначимо мінімальну ставку дисконтування, яка у загальному вигляді визначається за формулою:

$$\tau = d + f, \qquad (5.23)$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2023 році в Україні d= (0,14…0,2);

f – показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,05...0,1).

$$\tau = 0.16 + 0.05 = 0.21$$

Так як $E_в > \tau_{\min}$ то інвестор може бути зацікавлений у фінансуванні

даної наукової розробки.

Розрахуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в} \tag{5.24}$$

$$\mathrm{T_{ок}} = \frac{1}{0.28} = 3.57 \text{ роки} = 43 \text{ міс} = 3 \text{ роки та } 7 \text{ міс}$$

Так як $T_{ок} \leq$ 3...5-ти років, то фінансування даної наукової розробки в принципі є доцільним.

5.5 Висновки

Було проведено оцінку комерційного потенціалу розробки програмного забезпечення системи для управління обігом антикваріату, яка є на вище середньому рівні. При порівнянні нової розробки з аналогом виявлено, що вона є якіснішою і конкурентоспроможною відносно аналога, а також краще по частці технічних і економічних показників.

Прогнозування витрат на виконання науково-дослідної роботи по кожній з статей витрат складе 7293101 грн. Загальна ж величина витрат на виконання та впровадження результатів даної НДР буде складати 14586202 грн.

Вкладені інвестиції в даний проект окупляться через 3 роки та 7 місяців при прогнозованому прибутку 32005379 грн. за три роки.

ВИСНОВКИ

У результаті виконання магістерської кваліфікаційної роботи було досліджено методи і моделі проектування і розробки систем для управління обігом антикваріату. В ході дослідження було проведено аналіз сучасного стану ринку антикварних товарів і виявлено проблеми та недоліки існуючих систем управління обігом.

Також було проаналізовано і удосконалено метод побудови мікросервісної архітектури за рахунок асинхронного спілкування мікросервісів за допомогою повідомлень та орхестрація мікросервісів для забезпечення контролю виконання дій. Розроблена мікросервісна архітектура системи дозволила розділити її на незалежні компоненти, що сприяє покращенню продуктивності, масштабованості та підтримки системи.

Було проаналізовано методи аналізу даних та було удосконалено метод побудови системи аналізу даних за рахунок агрегації і структуризації даних в одну таблицю, що дало змогу покращити продуктивність і ефективність побудови звітів.

Вдосконалено метод пошуку оптимального шляху з використанням нейронних мереж що дозволяє системі самостійно вдосконалювати пошук на базі зібраних даних. Також впроваджено пошук клієнтських потреб за допомогою нейронних мереж на основі даних зібраних системою аналізу даних що дозволяє системі надавати персоналізовані пропозиції клієнтам, що відповідають їхнім індивідуальним потребам.

Був розроблений прототип системи управління обігом антикваріату. Також було проведене тестування, яке включало модульне тестування, інтеграційне тестування та системне тестування. Під час тестування було перевірено відповідність програмного забезпечення вимогам, правильність його функціонування, а також здатність обробляти помилки. Реалізація прототипу було успішно протестована і підтверджено працездатність програмного продукту.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Сафо В.В. Публікація – Міжнародна науково-практична інтернет-конференція “ Молодь в науці: дослідження, проблеми, перспективи” URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/viewFile/18661/15476

2. Сафо В.В. Публікація – Всеукраїнської науково-практичної Інтернет-конференції “Електронні інформаційні ресурси: створення, використання, доступ”,URL:https://drive.google.com/file/d/1eAb_QXS6CvA-e1ikKxerTqvw6P1IPRhc/view

3. NET Microservices: Architecture for Containerized .NET Applications, URL: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/ (дата звернення: 10.09.2023).

4. Кріс Річардсон. Мікросервіси. Паттерни розробки і рефакторинга. URL: https://n-knigi.com.ua/ua/p1608672063-mikroservisy-patterny-razrabotki.html (дата звернення: 10.09.2023).

5. Explore the UML sequence diagram – IBM Developer: URL: https:// https://developer.ibm.com/articles/the-sequence-diagram/ (дата звернення: 20.09.2023).

6. Miro Samek. A crash course in UML state machines. California, 2015, 257 p.

7. Visual Studio: IDE and Code Editor for Software Developers and Teams: URL: https://visualstudio.microsoft.com/ (дата звернення: 28.09.2023).

8. C# 10 in a Nutshell: The Definitive Reference. 1st Ed. Joseph Albahari, 2014, 608 c.

9. M. Brown. MVVM Unleashed – Pearson Education (US), 2014. 53 c.

10. Jeffrey Richter. CLR via C#. Washington, 2012, 896p.

11. Mark J. Price. C# 9 and .NET 5 - Modern Cross-Platform Development. Birmingham, 2023, 1328p

12. Joseph Albahari, Albahari. C# 9.0 Pocket Reference". Sebastopol, 2021, 264p.

13. Christian Nagel, Professional C# and .NET, 2021 1st Edition, 863p.

14. Stephen Cleary, Concurrency in C# Cookbook: Asynchronous, Parallel, and Multithreaded Programming 2nd Edition, 2019, 351p

15. Alex Berson. Client/Server Architecture. New York : McGraw-Hill, 1996. 569 p.

16. Srinivasan Desikan. Software Testing. Principles and Practices. Vancuver : Pearson, 2009. 480 p.

17. George H. Fairbanks. Just Enough Software Architecture: A Risk-Driven Approach. Boulder : Marshall & Brainerd, 2010. 376 p.

18. Azure Cognitive Services. [Електронний ресурс]. URL: https:// azure.microsoft.com/en-us/services/cognitive-services/

19. General Architecture for Text Engineering. [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/General_Architecture_for_Text_Engineering.

20. Recurrent neural network. [Електронний ресурс]. URL: https://en.wikipedia.org/wiki/Recurrent_neural_network.

21. George H. Fairbanks. Just Enough Software Architecture: A Risk-Driven Approach. Boulder : Marshall & Brainerd, 2010. 376 p.

22. Alex Berson. Client/Server Architecture. New York : McGraw-Hill, 1996. 569 p.

23. Security Authentication vs. Authorization. A Quick Guide. URL: https://swoopnow.com/security-authentication-vs-authorization (дата звернення: 10.10.2023).

24. JWT Authentication Flow with Refresh Tokens in ASP.NET Core Web API. URL: https://fullstackmark.com/post/19/jwt-authentication-flow-with-refresh-tokens-in-aspnet-core-web-api (дата звернення: 12.10.2023).

25. How to enable HTTPS on your server. URL: https://www.godaddy.com/garage/enable-https-server (дата звернення: 10.10.2023).

26. MySQL – Introduction. URL: https://tutorialspoint.com/mysql/ mysql-introduction.htm (дата звернення: 10.09.2023).

27. MS SQL Server. URL: https://searchsqlserver.tech.target.com /definition/SQL-Server (дата звернення: 10.09.2023).

28. Software testing – Wikipedia. URL: https://en.wikipedia.org /wiki/Software_testing (дата звернення: 10.09.2023).

29. Srinivasan Desikan. Software Testing. Principles and Practices. Vancuver : Pearson, 2009. 480 p.

30. How to configure and use Live Unit Testing. URL: https://docs.microsoft.com/en-us/visualstudio/test/live-unit-testing (дата звернення: 10.09.2023).

31. Нейроні мережі. URL: https://uk.wikipedia.org/wiki/ Штучна_нейронна_мережа (дата звернення: 10.09.2023).

32. Violity – Інтернет аукціон. URL: https://ain.ua/2023/03/24/shho-take-violity-ta-yak-na-nomu-prodavaty-instrukcziya/ (дата звернення: 10.09.2023).

33. OLX. URL: https://uk.wikipedia.org/wiki/OLX (дата звернення: 10.09.2023).

34. Нейроні мережі, теорія і практика. URL: http://eir.zntu.edu.ua/bitstream/123456789/6800/1/Subbotin_Neural.pdf (дата звернення: 10.09.2023).

35. Рекуре́нтні нейро́нні мере́жі. URL: https://uk.wikipedia.org/wiki/ Рекурентна_нейронна_мережа (дата звернення: 10.09.2023).

36. Deep Learning vs Machine Learning. URL: https://learn.microsoft.com/en-us/dotnet/machine-learning/deep-learning-overview (дата звернення: 10.09.2023).

37. Згорткова нейронна мережа. URL: https://uk.wikipedia.org/wiki/ Згорткова_нейронна_мережа (дата звернення: 10.09.2023).

38. Experience AI-powered browsing with the new Bing built-in. URL: https://www.microsoft.com/en-us/research/blog/microsoft-unveils-project-brainwav (дата звернення: 10.09.2023).

39. Accord.NET Framework. URL: http://accord-framework.net/index.html (дата звернення: 10.09.2023).

40. The Microsoft Cognitive Toolkit. URL: https://learn.microsoft.com/en-us/cognitive-toolkit/ (дата звернення: 10.09.2023).

41. CNTK Evaluation Overview. URL: https://learn.microsoft.com/en-us/cognitive-toolkit/CNTK-Evaluation-Overview (дата звернення: 10.09.2023).

42. CNTK Library C# API. URL: https://learn.microsoft.com/en-us/cognitive-toolkit/CNTK-Library-Managed-API (дата звернення: 10.09.2023).

43. CNTK. URL: https://github.com/microsoft/cntk (дата звернення: 10.09.2023).

44. Pattern: Microservice Architecture. URL: https://microservices.io/patterns/microservices.html (дата звернення: 10.09.2023).

45.What Are Microservices? URL: https://www.spiceworks.com/tech/devops/articles/what-are-microservices/ (дата звернення: 10.09.2023).

46. Microservices vs monolith. URL: https://www.n-ix.com/microservices-vs-monolith-which-architecture-best-choice-your-business/ (дата звернення: 10.09.2023).

47. Domain-Driven Design Principles for Microservices. URL: https://semaphoreci.com/blog/domain-driven-design-microservices (дата звернення: 10.09.2023).

48. .NET Microservices: Architecture for Containerized .NET Applications. URL: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/ (дата звернення: 10.09.2023).

49. Building and Deploying Microservices Using .NET. URL: https://www.symphony-solutions.eu/building-and-deploying-microservices-using-net/ (дата звернення: 10.09.2023).

50. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. – Вінниця : ВНТУ, 2021. – 42 с.

Д О Д А Т К И

102

Додаток А

Технічне завдання

Міністерство освіти і науки України
Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"19" вересня 2023 р.

Технічне завдання
на магістерську кваліфікаційну роботу «Розробка методів і засобів системи управління обігом антикваріату» за спеціальністю
121 – Інженерія програмного забезпечення

Керівник магістерської кваліфікаційної роботи:
_____ д. т. н., проф.каф. ПЗ Ліщинська Л.Б.
"19" вересня 2023 р.

Виконав:
_____студент гр.2ПІ-22м Сафо В.В.
"19" вересня 2023 р.

Вінниця – 2023 року

1. Найменування та галузь застосування

Магістерська кваліфікаційна робота: «Розробка методів і засобів системи управління обігом антикваріату».

Галузь застосування – торгівельні або інформаційні підприємства.

2. Підстава для розробки.

Підставою для виконання магістерської кваліфікаційної роботи (МКР) є індивідуальне завдання на МКР та наказ № 247 від 18 вересня 2023р. ректора по ВНТУ про закріплення тем МКР.

3. Мета та призначення розробки.

Метою дослідження підвищення ефективності системи для управління обігом антикваріату.

Призначення роботи – підвищення ефективності, продуктивності і надійності системи для управління обігом антикваріату.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись МКР.

1. Сафо В.В. Публікація – Міжнародна науково-практична інтернет-конференція “ Молодь в науці: дослідження, проблеми, перспективи” URL: https://conferences.vntu.edu.ua/index.php/mn/mn2023/paper/viewFile/18661/15476

2. Сафо В.В. Публікація – Всеукраїнської науково-практичної Інтернет-конференції “ Електронні інформаційні ресурси: створення, використання, доступ”,URL:https://drive.google.com/file/d/1eAb_QXS6CvA-e1ikKxerTqvw6P1IPRhc/view

3. NET Microservices: Architecture for Containerized .NET Applications, URL: https://learn.microsoft.com/en-us/dotnet/architecture/microservices/.

4. Кріс Річардсон. Мікросервіси. Паттерни розробки і рефакторинга. URL: https://n-knigi.com.ua/ua/p1608672063-mikroservisy-patterny-razrabotki.html.

5. Технічні вимоги

Платформа: Microsoft Azure, .NET.

Мова програмування: C#.

Технології: Microservices, Azure Functions, REST API, GraphQL API, SQL, SQL, Cosmos DB, Azure Service Bus, Azure Blob Storage, Redis.

6. Конструктивні вимоги.

Конструкція додатку повинна відповідати естетичним та ергономічним вимогам, повинна бути зручною в обслуговуванні та керуванні.

Графічна та текстова документація повинна відповідати діючим стандартам України.

7. Перелік технічної документації, що пред'являється по закінченню робіт:

– пояснювальна записка до МКР;

– технічне завдання;

– лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи
1	Аналіз сучасного стану питання та обґрунтування задачі	20.09.2023-6.09.2023
2	Розробка архітектури програмного додатку	6.09.2023-30.09.2023
3	Проектування бази даних	1.10.2023 – 7.10.2023
4	Обґрунтування вибору мови програмування та середовища розробки додатку	7.10.2023 – 10.10.2023
5.	Розробка загального алгоритму роботи програмного додатку	10.10.2023 – 1.10.2023
6.	Розробка графічного інтерфейсу	1.11.2023 – 10.11.2023
7.	Економічна частина	10.11.2023- 13.11.2023
8.	Оформлення матеріалів магістерської кваліфікаційної роботи	13.11.2023 – 01.12.2023

10. Порядок контролю та прийняття.

Виконання етапів магістерської кваліфікаційної роботи контролюється керівником згідно з графіком виконання роботи.

Прийняття магістерської кваліфікаційної роботи здійснюється ДЕК, затвердженою зав. кафедрою згідно з графіком.

105

Додаток Б

Протокол перевірки роботи

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: **Розробка методів і засобів системи управління обігом антикваріату.**

Тип роботи: кваліфікаційна робота

Підрозділ : кафедра програмного забезпечення, ФІТКІ, 2ПІ – 22м

Науковий керівник: д. т. н., проф.каф. ПЗ Ліщинська Л.Б.

Unicheck	
Оригінальність	**97,6%**
Схожість	2,4%

Аналіз звіту подібності

■ **Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.**

□ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

□ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Особа, відповідальна за перевірку _____ Черноволик Г. О.

Опис прийнятого рішення: **допустити до захисту**

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck

Автор роботи _____ Сафо В. В.

Керівник роботи _____ Ліщинська Л. Б.

Додаток В

Лістинг програми

```
using AntiquesMarket.Core.Entities;
using AntiquesMarket.Core.Enums;
using AntiquesMarket.Core.Managers;
using System.Data;

namespace AntiquesMarket.Client
{
   public partial class AddGoodsForm : Form
   {
      private Participant _participant;
      private GoodsHandler _goodsHandler;

      public AddGoodsForm(Participant participant)
      {
         InitializeComponent();
         _participant = participant;
         _goodsHandler = new GoodsHandler();
         InitForm();
      }

      private void buttonAddGoods_Click(object sender, EventArgs e)
      {
         try
         {
            Validation();

            var goods = new MarketGoods()
            {
               Name = textBoxName.Text,
               Photo = textBoxFilePath.Text,
               Price = double.Parse(textBoxPrice.Text),
               Description = textBoxDescription.Text,
               Count = int.Parse(textBoxCount.Text),
               Category = (GoogsCategory?)Enum.Parse(typeof(GoogsCategory), comboBoxCategory.Text),
            };

            _goodsHandler.Add(goods);

            Redirect();
         }
         catch (Exception ex)
         {
            labelError.Text = ex.Message;
         }
      }


      private void Validation()
      {
         if (string.IsNullOrWhiteSpace(textBoxName.Text) ||
               string.IsNullOrWhiteSpace(textBoxFilePath.Text) ||
               string.IsNullOrWhiteSpace(textBoxPrice.Text) ||
               string.IsNullOrWhiteSpace(textBoxDescription.Text) ||
               string.IsNullOrWhiteSpace(textBoxCount.Text) ||
               string.IsNullOrWhiteSpace(comboBoxCategory.Text))
         {
            throw new InvalidDataException("Please fill required fields");
         }
```

```
            if (!double.TryParse(textBoxPrice.Text, out double price))
            {
                throw new InvalidDataException("Invalid Price");
            }

            if (!int.TryParse(textBoxCount.Text, out int count))
            {
                throw new InvalidDataException("Invalid Count");
            }
        }

        private void InitForm()
        {
            labelError.Text = string.Empty;


comboBoxCategory.Items.AddRange(Enum.GetValues(typeof(GoogsCategory)).Cast<GoogsCategory>().Select(x
=> x.ToString()).ToArray());
            comboBoxCategory.SelectedIndex = 0;

        }

        private void Redirect()
        {
            this.Hide();
            Form redirectedForm = new MainForm(_participant);
            redirectedForm.Show();
        }

        private void buttonCancel_Click(object sender, EventArgs e)
        {
            Redirect();
        }

        private void buttonBrowse_Click(object sender, EventArgs e)
        {
            try
            {
                using (OpenFileDialog dlg = new OpenFileDialog())
                {
                    dlg.Title = "Download Image";
                    dlg.Filter = "Image Files (*.bmp;*.jpg;*.jpeg,*.png)|*.BMP;*.JPG;*.JPEG;*.PNG";

                    if (dlg.ShowDialog() == DialogResult.OK)
                    {
                        pictureBoxGoodsImage.ImageLocation = dlg.FileName;
                        textBoxFilePath.Text = dlg.FileName;
                    }
                }
            }
            catch (Exception ex)
            {
                labelError.Text = ex.Message;
            }
        }
    }
}
using AntiquesMarket.Core.Entities;
using AntiquesMarket.Core.Managers;

namespace AntiquesMarket.Client
{
    public partial class LoginForm : Form
```

```
    {
        ParticipantHendler _participantHendler;

        public LoginForm()
        {
            InitializeComponent();
            LoadForm();
            _participantHendler = new ParticipantHendler();
        }

        private void LoadForm()
        {
            labelError.Text = string.Empty;
        }

        private void linkLabelRegistration_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)
        {
            this.Hide();
            Form redirectedForm = new RegistrationForm();
            redirectedForm.Show();
        }

        private void buttonLogin_Click(object sender, EventArgs e)
        {
            try
            {
                var participant = Validation();
                this.Hide();
                Form redirectedForm = new MainForm(participant);
                redirectedForm.Show();

            }
            catch (Exception ex)
            {
                labelError.Text = ex.Message;
            }
        }

        private Participant Validation()
        {
            if (string.IsNullOrWhiteSpace(textBoxUsername.Text) ||
                string.IsNullOrWhiteSpace(textBoxPassword.Text))
            {
                throw new InvalidDataException("Username or password is not provided");
            }

            var participants = _participantHendler.Retrieve();
            var participant = participants?.FirstOrDefault(x => x.Login == textBoxUsername.Text && x.Password ==
textBoxPassword.Text);

            if (participant == null)
            {
                throw new InvalidDataException("Invalid username or password");
            }

            return participant;
        }

    }
}

using AntiquesMarket.Client.Enums;
using AntiquesMarket.Core.Entities;
```

```
using AntiquesMarket.Core.Enums;
using AntiquesMarket.Core.Managers;

namespace AntiquesMarket.Client
{
   public partial class MainForm : Form
   {
      private readonly Participant _participant;

      private readonly ParticipantHendler _participantHendler;
      private readonly GoodsHandler _goodsHandler;
      private readonly OrderHandler _orderHandler;

      private MarketGoods _selectedGoods;
      private MarketOrder _selectedOrder;

      public MainForm(Participant participant)
      {
         InitializeComponent();
         _participant = participant;
         _participantHendler = new ParticipantHendler();
         _goodsHandler = new GoodsHandler();
         _orderHandler = new OrderHandler();
         InitForm();
      }

      private void InitForm()
      {

         labelUserName.Text = _participant.Login;
         labelUserRole.Text = _participant.Role.ToString();

         var goods = _goodsHandler.Retrieve();
         var participants = _participantHendler.Retrieve();
         var orders = _orderHandler.Retrieve();
         FillDataGridViewGoods(goods);

         comboBoxGoogsCategorySort.Items.Add("All");

comboBoxGoogsCategorySort.Items.AddRange(Enum.GetValues(typeof(GoogsCategory)).Cast<GoogsCategory>(
).Select(x => x.ToString()).ToArray());
         comboBoxGoogsCategorySort.SelectedIndex = 0;

         comboBoxOrderStatusSorting.Items.Add("All");

comboBoxOrderStatusSorting.Items.AddRange(Enum.GetValues(typeof(OrderStatus)).Cast<OrderStatus>().Select(
x => x.ToString()).ToArray());
         comboBoxOrderStatusSorting.SelectedIndex = 0;

         switch (_participant.Role)
         {
            case ParticipantRole.Client:
               buttonAddGoods.Visible = false;
               buttonOrderApprove.Visible = false;
               buttonDecline.Visible = false;
               tabControlMain.TabPages.Remove(tabPageParticipants);
               FillDataGridViewOrders(orders.Where(x => x.ClientId == _participant.Id).ToArray(), participants);
               break;
            case ParticipantRole.MarketManager:
               buttonGoodsOrder.Visible = false;
               tabControlMain.TabPages.Remove(tabPageParticipants);
               FillDataGridViewOrders(orders.Where(x => x.ManagerId == _participant.Id).ToArray(), participants);
               break;
```

```
            case ParticipantRole.ClientManager:
                buttonAddGoods.Visible = false;
                buttonGoodsOrder.Visible = false;
                tabControlMain.TabPages.Remove(tabPageParticipants);
                FillDataGridViewOrders(orders, participants);
                break;
            case ParticipantRole.SystemAdmin:
                FillDataGridViewParticipants(participants);
                FillDataGridViewOrders(orders, participants);

                comboBoxParticipantRoleSorting.Items.Add("All");

comboBoxParticipantRoleSorting.Items.AddRange(Enum.GetValues(typeof(ParticipantRole)).Cast<ParticipantRole
>().Select(x => x.ToString()).ToArray());
                comboBoxParticipantRoleSorting.SelectedIndex = 0;

                break;
        }


    }

    private void FillDataGridViewParticipants(Participant[] items)
    {
        dataGridViewParticipants.Rows.Clear();
        foreach (var item in items)
        {
            dataGridViewParticipants.Rows.Add(new object[]
            {
                item.Id,
                item.FirstName,
                item.LastName,
                item.Phone,
                item.Email,
                item.Address,
                item.Role.ToString(),
                item.Login
            });
        }
    }

    private void FillDataGridViewOrders(MarketOrder[] items, Participant[] participants)
    {
        dataGridViewOrders.Rows.Clear();
        foreach (var item in items)
        {

            dataGridViewOrders.Rows.Add(new object[]
            {
                item.Id,
                participants.FirstOrDefault(x => x.Id == item.ClientId)?.FirstName ?? string.Empty,
                participants.FirstOrDefault(x => x.Id == item.ManagerId)?.FirstName ?? string.Empty,
                item.Status.ToString()
            });
        }
        FillOrderDetails(items.FirstOrDefault()?.Id);
    }

    private void FillDataGridViewGoods(MarketGoods[] items)
    {
        dataGridViewGoods.Rows.Clear();
        foreach (var item in items)
        {
```

```
            dataGridViewGoods.Rows.Add(new object[]
            {
                item.Id,
                item.Name,
                item.Category.ToString(),
                item.Price,
                item.Count
            });
        }

        FillGoodsDetails(items.FirstOrDefault()?.Id);
    }

    private void buttonLogout_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form redirectedForm = new LoginForm();
        redirectedForm.Show();
    }

    private void buttonCreateParticipant_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form redirectedForm = new RegistrationForm(_participant);
        redirectedForm.Show();
    }

    private void buttonAddGoods_Click(object sender, EventArgs e)
    {
        this.Hide();
        Form redirectedForm = new AddGoodsForm(_participant);
        redirectedForm.Show();
    }

    private void dataGridViewGoods_RowStateChanged(object sender,
DataGridViewRowStateChangedEventArgs e)
    {
        if (e.StateChanged != DataGridViewElementStates.Selected) return;
        FillGoodsDetails(e?.Row?.Cells[0]?.Value?.ToString());
    }


    private void FillGoodsDetails(string id)
    {
        if (string.IsNullOrWhiteSpace(id)) return;

        _selectedGoods = _goodsHandler.Retrieve(id);
        if (_selectedGoods == null) return;

        labelGoodsId.Text = _selectedGoods.Id;
        labelGoodsName.Text = _selectedGoods.Name;
        labelGoodsPrice.Text = _selectedGoods.Price.ToString();
        labelGoodsCategory.Text = _selectedGoods.Category.ToString();
        labelGoodsDescription.Text = _selectedGoods.Description;
        labelGoodsCount.Text = _selectedGoods.Count.ToString();
        pictureBoxGoogds.ImageLocation = _goodsHandler.DownloadPhoto(_selectedGoods.Photo);
    }


    private void FillOrderDetails(string id)
    {
        if (string.IsNullOrWhiteSpace(id)) return;
```

```
            _selectedOrder = _orderHandler.Retrieve(id);
            if (_selectedOrder == null) return;

            var client = _participantHendler.Retrieve(_selectedOrder.ClientId);

            pictureBoxOrder.ImageLocation = _goodsHandler.DownloadPhoto(_selectedOrder.Goods.Photo);
            labelOrderGoodsName.Text = _selectedOrder.Goods.Name;
            labelOrderPrice.Text = _selectedOrder.Goods.Price.ToString();
            labelOrderCategory.Text = _selectedOrder.Goods.Category.ToString();
            labelClientName.Text = $"{client.FirstName} {client.LastName}";
            labelAddress.Text = client.Address;
            labelEmail.Text = client.Email;
            labelPhone.Text = client.Phone;
            labelOrderStatus.Text = _selectedOrder.Status.ToString();
            labelOrderReson.Text = _selectedOrder.Reason;
        }

        private void buttonGoodsOrder_Click(object sender, EventArgs e)
        {
            if (_selectedGoods == null) return;
            this.Hide();
            var order = new MarketOrder(_selectedGoods);
            order.ClientId = _participant.Id;
            Form redirectedForm = new OrderForm(_participant, order, OrderAction.Create);
            redirectedForm.Show();
        }

        private void dataGridViewOrders_RowStateChanged(object sender,
DataGridViewRowStateChangedEventArgs e)
        {
            if (e.StateChanged != DataGridViewElementStates.Selected) return;
            FillOrderDetails(e?.Row?.Cells[0]?.Value?.ToString());
        }

        private void buttonOrderApprove_Click(object sender, EventArgs e)
        {
            if (_selectedOrder == null) return;
            this.Hide();
            Form redirectedForm = new OrderForm(
                _participant,
                _selectedOrder,
                _participant.Role == ParticipantRole.ClientManager ? OrderAction.Approve : OrderAction.Deliver);
            redirectedForm.Show();
        }

        private void buttonDecline_Click(object sender, EventArgs e)
        {
            if (_selectedOrder == null) return;
            this.Hide();
            Form redirectedForm = new OrderForm(_participant, _selectedOrder, OrderAction.Decline);
            redirectedForm.Show();
        }

        private void comboBoxGoogsCategorySort_SelectedValueChanged(object sender, EventArgs e)
        {

            GoogsCategory? category = null;
            if (comboBoxGoogsCategorySort.Text != "All")
            {
                category = (GoogsCategory?)Enum.Parse(typeof(GoogsCategory), comboBoxGoogsCategorySort.Text);
            }

            var goods = _goodsHandler.Retrieve(category);
```

```
            FillDataGridViewGoods(goods);
        }

        private void comboBoxOrderStatusSorting_SelectedValueChanged(object sender, EventArgs e)
        {
            OrderStatus? status = null;
            if (comboBoxOrderStatusSorting.Text != "All")
            {
                status = (OrderStatus?)Enum.Parse(typeof(OrderStatus), comboBoxOrderStatusSorting.Text);
            }

            var participants = _participantHendler.Retrieve();
            var orders = _orderHandler.Retrieve();

            switch (_participant.Role)
            {
                case ParticipantRole.Client:
                    FillDataGridViewOrders(orders.Where(x => x.ClientId == _participant.Id && (status == null || x.Status
== status)).ToArray(), participants);
                    break;
                case ParticipantRole.MarketManager:
                    FillDataGridViewOrders(orders.Where(x => x.ManagerId == _participant.Id && (status == null ||
x.Status == status)).ToArray(), participants);
                    break;
                case ParticipantRole.ClientManager:
                    FillDataGridViewOrders(orders.Where(x => status == null || x.Status == status).ToArray(),
participants);
                    break;
                case ParticipantRole.SystemAdmin:
                    FillDataGridViewOrders(orders.Where(x => status == null || x.Status == status).ToArray(),
participants);
                    break;

            }
        }

        private void comboBoxParticipantRoleSorting_SelectedIndexChanged(object sender, EventArgs e)
        {
            ParticipantRole? role = null;
            if (comboBoxParticipantRoleSorting.Text != "All")
            {
                role = (ParticipantRole?)Enum.Parse(typeof(ParticipantRole), comboBoxParticipantRoleSorting.Text);
            }

            var participants = _participantHendler.Retrieve(role);
            FillDataGridViewParticipants(participants);
        }
    }
}
using AntiquesMarket.Client.DataContrants;
using AntiquesMarket.Client.Enums;
using AntiquesMarket.Core.Entities;
using AntiquesMarket.Core.Enums;
using AntiquesMarket.Core.Managers;
using System.Data;

namespace AntiquesMarket.Client
{

    public partial class OrderForm : Form
    {
        private readonly Participant _participant;
        private MarketOrder _marketOrder;
```

```
private readonly ParticipantHendler _participantHandler;
private readonly GoodsHandler _goodsHandler;
private readonly OrderHandler _orderHandler;
private readonly OrderAction _orderAction;

public OrderForm(Participant participant, MarketOrder marketOrder, OrderAction orderAction)
{
   InitializeComponent();
   _participant = participant;
   _marketOrder = marketOrder;
   _participantHandler = new ParticipantHendler();
   _goodsHandler = new GoodsHandler();
   _orderHandler = new OrderHandler();
   InitForm(orderAction);
   _orderAction = orderAction;
}

private void InitForm(OrderAction orderAction)
{
   var participants = _participantHandler.Retrieve();

   comboBoxManagers.Items.AddRange(participants.Where(x=> x.Role == ParticipantRole.MarketManager)
                                    .Select(x => new ComboboxItem()
                                    {
                                       Text = $"{x.FirstName} {x.LastName}",
                                       Value = x.Id
                                    }).ToArray());
   comboBoxManagers.SelectedIndex = 0;

   var client = participants.First(x => x.Id == _marketOrder.ClientId);

   pictureBoxOrder.ImageLocation = _goodsHandler.DownloadPhoto(_marketOrder.Goods.Photo);
   textBoxDescription.Text = _marketOrder.Goods.Description;
   labelOrderId.Text = _marketOrder.Id;
   labelOrderGoodsName.Text = _marketOrder.Goods.Name;
   labelOrderPrice.Text = _marketOrder.Goods.Price.ToString();
   labelOrderCategory.Text = _marketOrder.Goods.Category.ToString();
   labelClientName.Text = $"{client.FirstName} {client.LastName}";
   labelAddress.Text = client.Address;
   labelEmail.Text = client.Email;
   labelPhone.Text = client.Phone;

   labelError.Text = string.Empty;


   switch (orderAction)
   {
      case OrderAction.Create:
      case OrderAction.Deliver:
         labelManager.Visible = false;
         comboBoxManagers.Visible = false;
         labelReason.Visible = false;
         textBoxReason.Visible = false;
         break;
      case OrderAction.Approve:
         labelReason.Visible = false;
         textBoxReason.Visible = false;
         break;

      case OrderAction.Decline:
         labelManager.Visible = false;
         comboBoxManagers.Visible = false;
```

```
            break;
        }

    }

    private void buttonOk_Click(object sender, EventArgs e)
    {
        try
        {
            HandleOk();
            Redirect();
        }
        catch (Exception ex)
        {
            labelError.Text = ex.Message;
        }

    }

    private void Redirect()
    {
        this.Hide();
        Form redirectedForm = new MainForm(_participant);
        redirectedForm.Show();
    }

    private void HandleOk()
    {
        switch (_orderAction)
        {
            case OrderAction.Create:
                _orderHandler.Add(_marketOrder);
                break;
            case OrderAction.Approve:
                if (_marketOrder.Status != OrderStatus.New)
                {
                    throw new InvalidDataException($"Order in status {_marketOrder.Status}");
                }
                _marketOrder.Status = OrderStatus.Approved;
                _marketOrder.ManagerId = (comboBoxManagers.SelectedItem as ComboboxItem).Value.ToString();
                _orderHandler.Edit(_marketOrder.Id, _marketOrder);
                break;
            case OrderAction.Deliver:
                if (_marketOrder.Status != OrderStatus.Approved)
                {
                    throw new InvalidDataException($"Order in status {_marketOrder.Status}");
                }
                _marketOrder.Status = OrderStatus.Delivered;
                _orderHandler.Edit(_marketOrder.Id, _marketOrder);
                break;
            case OrderAction.Decline:
                if (string.IsNullOrWhiteSpace(textBoxReason.Text))
                {
                    throw new InvalidDataException("Reason is not provided");
                }

                if (_marketOrder.Status == OrderStatus.Delivered)
                {
                    throw new InvalidDataException($"Order in status {_marketOrder.Status}");
                }
                _marketOrder.Status = OrderStatus.Declined;
                _marketOrder.Reason = textBoxReason.Text;
                _orderHandler.Edit(_marketOrder.Id, _marketOrder);
```

```
                break;
            }
        }

        private void buttonCancel_Click(object sender, EventArgs e)
        {
            Redirect();
        }
    }
}
using AntiquesMarket.Core.Entities;
using AntiquesMarket.Core.Enums;
using AntiquesMarket.Core.Managers;
using System.Data;

namespace AntiquesMarket.Client
{
    public partial class RegistrationForm : Form
    {
        private ParticipantHendler _participantHendler;
        private Participant _admin;


        public RegistrationForm(Participant admin = null)
        {
            InitializeComponent();
            _participantHendler = new ParticipantHendler();
            _admin = admin;
            InitForm();
        }

        private void buttonRegistr_Click(object sender, EventArgs e)
        {
            try
            {
                Validation();

                var participant = new Participant()
                {
                    FirstName = textBoxFirstName.Text,
                    LastName = textBoxLastName.Text,
                    Phone = textBoxPhone.Text,
                    Email = textBoxEmail.Text,
                    Address = textBoxAddress.Text,
                    Login = textBoxUsername.Text,
                    Password = textBoxPassword.Text,
                    Role = _admin == null ? ParticipantRole.Client :
(ParticipantRole?)Enum.Parse(typeof(ParticipantRole), comboBoxRole.Text),
                };

                _participantHendler.Add(participant);

                Redirect();
            }
            catch (Exception ex)
            {
                labelError.Text = ex.Message;

            }
        }

        private void Redirect()
        {
```

```
            this.Hide();
            Form redirectedForm = _admin == null ? new LoginForm() : new MainForm(_admin);
            redirectedForm.Show();
        }

        private void Validation()
        {
            if (string.IsNullOrWhiteSpace(textBoxAddress.Text) ||
                string.IsNullOrWhiteSpace(textBoxConfirmPassword.Text) ||
                string.IsNullOrWhiteSpace(textBoxEmail.Text) ||
                string.IsNullOrWhiteSpace(textBoxFirstName.Text) ||
                string.IsNullOrWhiteSpace(textBoxLastName.Text) ||
                string.IsNullOrWhiteSpace(textBoxPassword.Text) ||
                string.IsNullOrWhiteSpace(textBoxPhone.Text) ||
                string.IsNullOrWhiteSpace(textBoxUsername.Text))
            {
                throw new InvalidDataException("Please fill required fields");
            }

            if (textBoxConfirmPassword.Text != textBoxPassword.Text)
            {
                throw new InvalidDataException("Invalid confirmed password");
            }

            var participants = _participantHendler.Retrieve();
            var participant = participants?.FirstOrDefault(x => x.Login == textBoxUsername.Text && x.Password ==
textBoxPassword.Text);

            if (participants?.Any(x => x.Login == textBoxUsername.Text ||
                                  x.Phone == textBoxPhone.Text ||
                                  x.Email == textBoxEmail.Text) == true)
            {
                throw new InvalidDataException("Dublicate account");
            }
        }

        private void InitForm()
        {
            labelError.Text = string.Empty;

            if (_admin == null)
            {
                comboBoxRole.Visible = false;
                labelParticipantRule.Visible = false;
            }
            else
            {

comboBoxRole.Items.AddRange(Enum.GetValues(typeof(ParticipantRole)).Cast<ParticipantRole>().Select(x =>
x.ToString()).ToArray());
                comboBoxRole.SelectedIndex = 0;
            }

        }

        private void buttonCancel_Click(object sender, EventArgs e)
        {
            Redirect();
        }
    }
}
namespace AntiquesMarket.Common.Interfaces
{
```

```
    public interface IDBEntity
    {
        public string Id { get; set; }
    }
}
namespace AntiquesMarket.Common
{
    public class BlobStorageManager
    {
        public BlobStorageManager()
        {

        }

        public string Add(string filePath)
        {
            string extension = filePath.Substring(filePath.IndexOf("."));
            string id = $"{Guid.NewGuid().ToString()}{extension}";
            File.Copy(filePath, Path.Combine(ConfigurationManager.CONNECTION_STRING, $"BlobStorage/{id}"));
            return id;
        }

        public void Remove(string id)
        {
            File.Delete(Path.Combine(ConfigurationManager.CONNECTION_STRING, $"{id}"));
        }

        public string Download(string id)
        {
            return Path.Combine(ConfigurationManager.CONNECTION_STRING, $"BlobStorage/{id}");
        }
    }
}
using Microsoft.Extensions.Caching.Memory;


namespace AntiquesMarket.Common
{
    public static class CacheManager
    {
                private static readonly MemoryCache _cache = new MemoryCache(new
MemoryCacheOptions());
                private static readonly int _expirationInterval = 360;

                public static T Get<T>(string key, object locker, Func<T> initializer) where T : class
                {
                        var cachedItem = _cache.Get(key) as T;
                        if (cachedItem != null)
                        {
                                return cachedItem;
                        }

                        lock (locker)
                        {
                                cachedItem = _cache.Get(key) as T;
                                if (cachedItem == null)
                                {
                                        cachedItem = initializer();
                                        _cache.Set(key, cachedItem, new
DateTimeOffset(DateTime.Now.AddMinutes(_expirationInterval)));
                                }
                        }
```

```
                    return cachedItem;
                }


            }
}
using AntiquesMarket.Common.Interfaces;
using Newtonsoft.Json;

namespace AntiquesMarket.Common
{
    public class DatabaseManager<T> where T : class, IDBEntity
    {
        private string _connectionString;

        public DatabaseManager(string connectionString)
        {
            _connectionString = connectionString;
        }

        public string Add(T item)
        {
            string id = Guid.NewGuid().ToString();
            item.Id = id;
            File.WriteAllText(Path.Combine(_connectionString, $"{id}.json"), JsonConvert.SerializeObject(item));
            return id;
        }

        public void Update(string id, T item)
        {
            File.WriteAllText(Path.Combine(_connectionString, $"{id}.json"), JsonConvert.SerializeObject(item));
        }

        public void Remove(string id)
        {
            File.Delete(Path.Combine(_connectionString, $"{id}.json"));
        }

        public T Retrive(string id)
        {
            string content = File.ReadAllText(Path.Combine(_connectionString, $"{id}.json"));
            return JsonConvert.DeserializeObject<T>(content);
        }

        public T[] Retrive()
        {
            string[] files = Directory.GetFiles(_connectionString);
            return files.Select(x =>
            {
                string content = File.ReadAllText(Path.Combine(_connectionString, x));
                return JsonConvert.DeserializeObject<T>(content);
            }).ToArray();
        }
    }
}
namespace AntiquesMarket.Core.Entities
{
    public class MarketClient: Participant
    {

    }
}
namespace AntiquesMarket.Core.Entities
```

```
{
   public class MarketClientManager : Participant
   {

   }
}
using AntiquesMarket.Common.Interfaces;
using AntiquesMarket.Core.Enums;

namespace AntiquesMarket.Core.Entities
{
   public class MarketGoods : IDBEntity
   {
                        public string Id { get; set; }
                        public string Name { get; set; }
                        public GoogsCategory? Category { get; set; }
                        public string Photo { get; set; }
                        public double? Price { get; set; }
                        public string Description { get; set; }
                        public int? Count { get; set; }
           }
}
using AntiquesMarket.Common.Interfaces;
using AntiquesMarket.Core.Enums;

namespace AntiquesMarket.Core.Entities
{
   public class MarketOrder : IDBEntity
   {
                        public string Id { get; set; }
                        public MarketGoods Goods { get; set; }
                        public string ClientId { get; set; }
                        public string ManagerId { get; set; }
                        public OrderStatus? Status { get; set; }
                        public string Reason { get; set; }

                        public MarketOrder(MarketGoods marketGoods)
                        {
                                Goods = marketGoods;
                                Status = OrderStatus.New;
                        }

                        public MarketOrder()
                        {

                        }

           }
}
namespace AntiquesMarket.Core.Entities
{
   public class MarketShopManager: Participant
   {
   }
}
namespace AntiquesMarket.Core.Entities
{
   public class MarketSystemManager : Participant
   {
   }
}
using AntiquesMarket.Core.Enums;
using AntiquesMarket.Core.Interfaces;
```

```
namespace AntiquesMarket.Core.Entities
{
   public class Participant : IParticipant
   {
                        public string Id { get; set; }
                        public string FirstName { get; set; }
                        public string LastName { get; set; }
                        public string Phone { get; set; }
                        public string Email { get; set; }
                        public string Address { get; set; }
                        public ParticipantRole? Role { get; set; }
                        public string Login { get; set; }
                        public string Password { get; set; }
                        public MarketOrder[] Orders { get; set; }
           }
}
namespace AntiquesMarket.Core.Enums
{
   public enum GoogsCategory
   {
      Coins,
      Books,
      Arms
   }
}
namespace AntiquesMarket.Core.Enums
{
   public enum OrderStatus
   {
      New,
      Approved,
      Declined,
      Delivered
   }
}
namespace AntiquesMarket.Core.Enums
{
   public enum ParticipantRole
   {
      SystemAdmin,
      Client,
      ClientManager,
      MarketManager
   }
}
using AntiquesMarket.Common.Interfaces;
using AntiquesMarket.Core.Enums;

namespace AntiquesMarket.Core.Interfaces
{
   public interface IParticipant : IDBEntity
           {
                        public string FirstName { get; set; }
                        public string LastName { get; set; }
                        public string Phone { get; set; }
                        public string Email { get; set; }
                        public string Address { get; set; }
                        public ParticipantRole? Role { get; set; }
                        public string Login { get; set; }
                        public string Password { get; set; }
           }
}
```

```
using AntiquesMarket.Common;
using AntiquesMarket.Core.Entities;
using AntiquesMarket.Core.Enums;

namespace AntiquesMarket.Core.Managers
{
   public class GoodsHandler
   {
                  private DatabaseManager<MarketGoods> _databaseManager;
                  private BlobStorageManager _blobStorage;


      public GoodsHandler()
      {
                        _databaseManager = new
DatabaseManager<MarketGoods>(Path.Combine(ConfigurationManager.CONNECTION_STRING, "Goods"));
                        _blobStorage = new BlobStorageManager();
                  }

                  public string Add(MarketGoods item)
                  {
                        item.Photo = _blobStorage.Add(item.Photo);
                        return _databaseManager.Add(item);
                  }
                  public void Remove(string id)
                  {
                        _databaseManager.Remove(id);
                  }
                  public void Edit(string id, MarketGoods item)
                  {
                        var currentItem = _databaseManager.Retrive(id);
                        if (currentItem == null)
                        {
                              throw new InvalidDataException("Goods does not exist");
                        }
                        if (item.Photo != null)
                        {
                              item.Photo = _blobStorage.Add(item.Photo);
                              _blobStorage.Remove(currentItem.Photo);
                        }

                        _databaseManager.Update(id, new MarketGoods()
                        {
                              Id = id,
                              Category = item.Category ?? currentItem.Category,
                              Photo = item.Photo ?? currentItem.Photo,
                              Price = item.Price ?? currentItem.Price,
                              Description = item.Description ?? currentItem.Description,
                              Count = item.Count ?? currentItem.Count
                        });
                  }

                  public MarketGoods[] Retrieve(GoogsCategory? category = null)
                  {
                        var items = _databaseManager.Retrive();
                        if (category == null)
                        {
                              return items;
                        }

                        return items.Where(x => x.Category == category).ToArray();
                  }
```

```
                public MarketGoods Retrieve(string id)
                {
                        return _databaseManager.Retrive(id);
                }

                public string DownloadPhoto(string fileName)
                {
                        return _blobStorage.Download(fileName);
                }
        }
}
using AntiquesMarket.Core.Entities;
namespace AntiquesMarket.Core.Managers
{
   public class LoginHandler
   {
      private ParticipantHendler _participantHendler;

      public LoginHandler()
      {
         _participantHendler = new ParticipantHendler();
      }
      public (bool isLoggedIn, Participant patricipant) Login(string username, string password)
      {
         var participants = _participantHendler.Retrieve();
         var patricipant = participants?.FirstOrDefault(x => x.Login == username && x.Password == password);
         if (patricipant != null)
         {
            return (true, patricipant);
         }

         return (false, null);
      }
   }
}
using AntiquesMarket.Common;
using AntiquesMarket.Core.Entities;
using AntiquesMarket.Core.Enums;

namespace AntiquesMarket.Core.Managers
{
   public class OrderHandler
   {
      private DatabaseManager<MarketOrder> _databaseManager;

      public OrderHandler()
      {
         _databaseManager = new
DatabaseManager<MarketOrder>(Path.Combine(ConfigurationManager.CONNECTION_STRING, "Orders"));
      }

                public string Add(MarketOrder item)
                {
                        return _databaseManager.Add(item);
                }
                public void Remove(string id)
                {
                        _databaseManager.Remove(id);
                }
                public void Edit(string id, MarketOrder item)
                {
                        var currentItem = _databaseManager.Retrive(id);
                        if (currentItem == null)
```

```
                {
                        throw new InvalidDataException("Order does not exist");
                }

                _databaseManager.Update(id, new MarketOrder()
                {
                        Id = id,
                        Goods = item.Goods ?? currentItem.Goods,
                        ClientId = item.ClientId ?? currentItem.ClientId,
                        ManagerId = item.ManagerId ?? currentItem.ManagerId,
                        Status = item.Status ?? currentItem.Status,
                        Reason = item.Reason ?? currentItem.Reason
                });
        }
        public MarketOrder[] Retrieve(OrderStatus? status = null, string clientId = null)
        {
                var items = _databaseManager.Retrive();
                if (status == null && clientId == null)
                {
                        return items;
                }
                return items.Where(x => (status != null && x.Status == status) || (clientId != null &&
x.ClientId == clientId)).ToArray();
        }

        public MarketOrder Retrieve(string id)
        {
                return _databaseManager.Retrive(id);
        }

    }
}
using AntiquesMarket.Common;
using AntiquesMarket.Core.Entities;
using AntiquesMarket.Core.Enums;

namespace AntiquesMarket.Core.Managers
{
   public class ParticipantHendler
   {
        private DatabaseManager<Participant> _databaseManager;

        public ParticipantHendler()
        {
                _databaseManager = new
DatabaseManager<Participant>(Path.Combine(ConfigurationManager.CONNECTION_STRING, "Participants"));
        }

        public string Add(Participant item)
        {
                return _databaseManager.Add(item);
        }
        public void Remove(string id)
        {
                _databaseManager.Remove(id);
        }
        public void Edit(string id, Participant item)
        {
                var currentItem = _databaseManager.Retrive(id);
                if (currentItem == null)
                {
                        throw new InvalidDataException("Participant does not exist");
                }
```

```
                    _databaseManager.Update(id, new Participant()
                    {
                        Id = id,
                        FirstName = item.FirstName ?? currentItem.FirstName,
                        LastName = item.LastName ?? currentItem.LastName,
                        Phone = item.Phone ?? currentItem.Phone,
                        Email = item.Email ?? currentItem.Email,
                        Address = item.Address ?? currentItem.Address,
                        Role = item.Role ?? currentItem.Role,
                        Orders = item.Orders ?? currentItem.Orders
                    });
                }

                public Participant[] Retrieve(ParticipantRole? role = null)
                {
                    var items = _databaseManager.Retrive();
                    if (role == null)
                    {
                        return items;
                    }

                    return items.Where(x => x.Role == role).ToArray();
                }

                public Participant Retrieve(string id)
                {
                    return _databaseManager.Retrive(id);
                }
            }
}
using AntiquesMarket.Core.Entities;

namespace AntiquesMarket.Core.Managers
{
   public class ReportHandler
   {
                public ReportHandler()
                {
                }
                public string Add(MarketReport item)
                {
                    throw new System.NotImplementedException("Not implemented");
                }
                public void Remove(string id)
                {
                    throw new System.NotImplementedException("Not implemented");
                }
                public void Edit(string id, MarketReport item)
                {
                    throw new System.NotImplementedException("Not implemented");
                }

                public void Upload(string id, MarketReport item)
                {
                    throw new System.NotImplementedException("Not implemented");
                }
                public MarketReport[] Retrieve()
                {
                    throw new System.NotImplementedException("Not implemented");
                }
            }
}
```

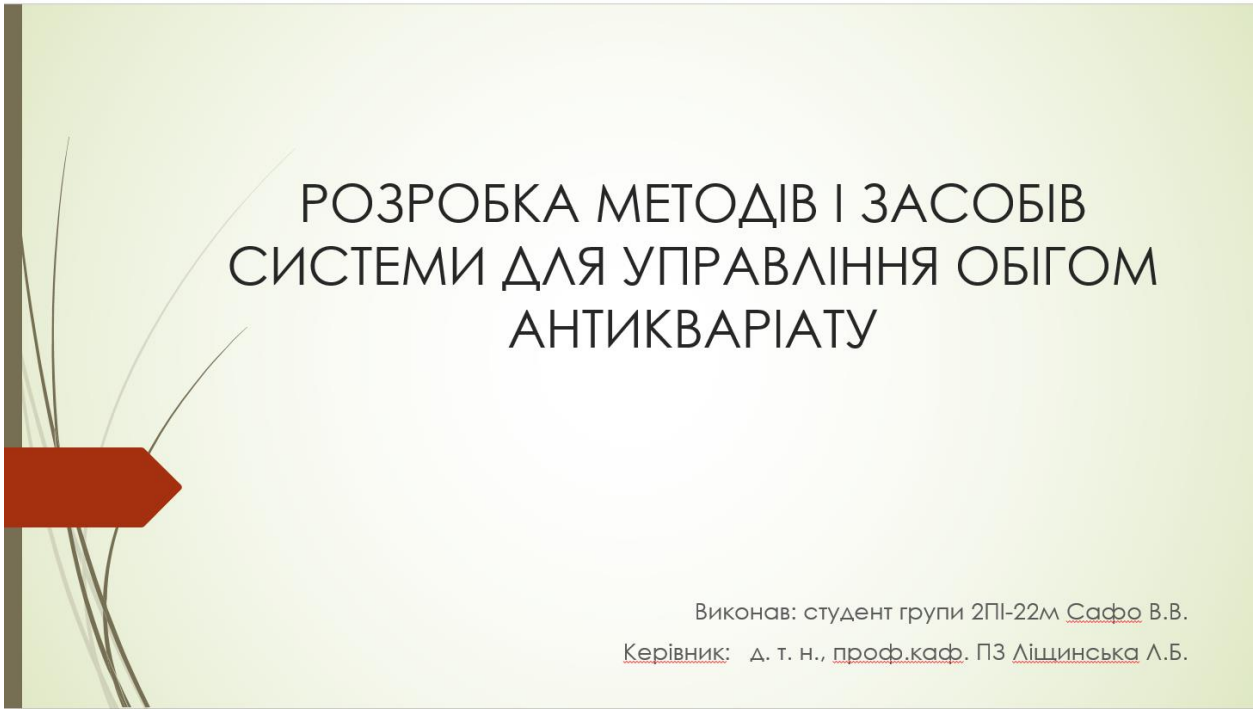
Додаток Г

Ілюстративна частина

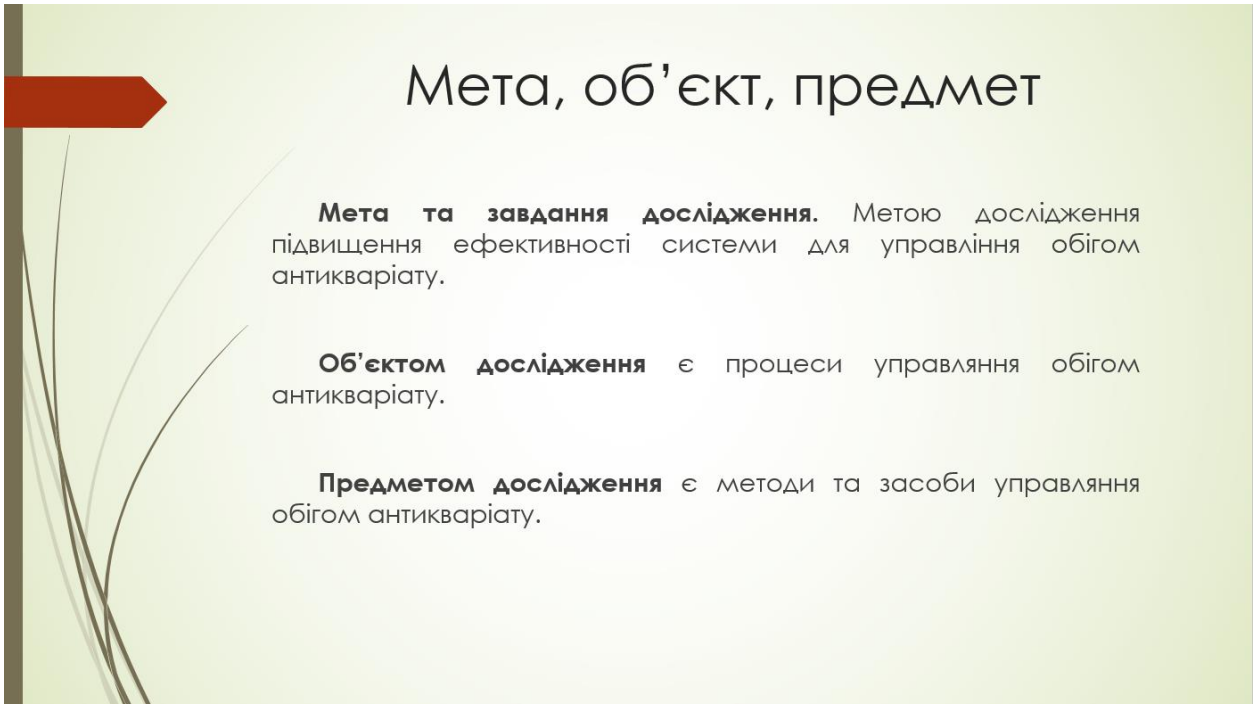
ІЛЮСТРАТИВНА ЧАСТИНА

РОЗРОБКА МЕТОДІВ І ЗАСОБІВ СИСТЕМИ УПРАВЛІННЯ ОБІГОМ АНТИКВАРІАТУ

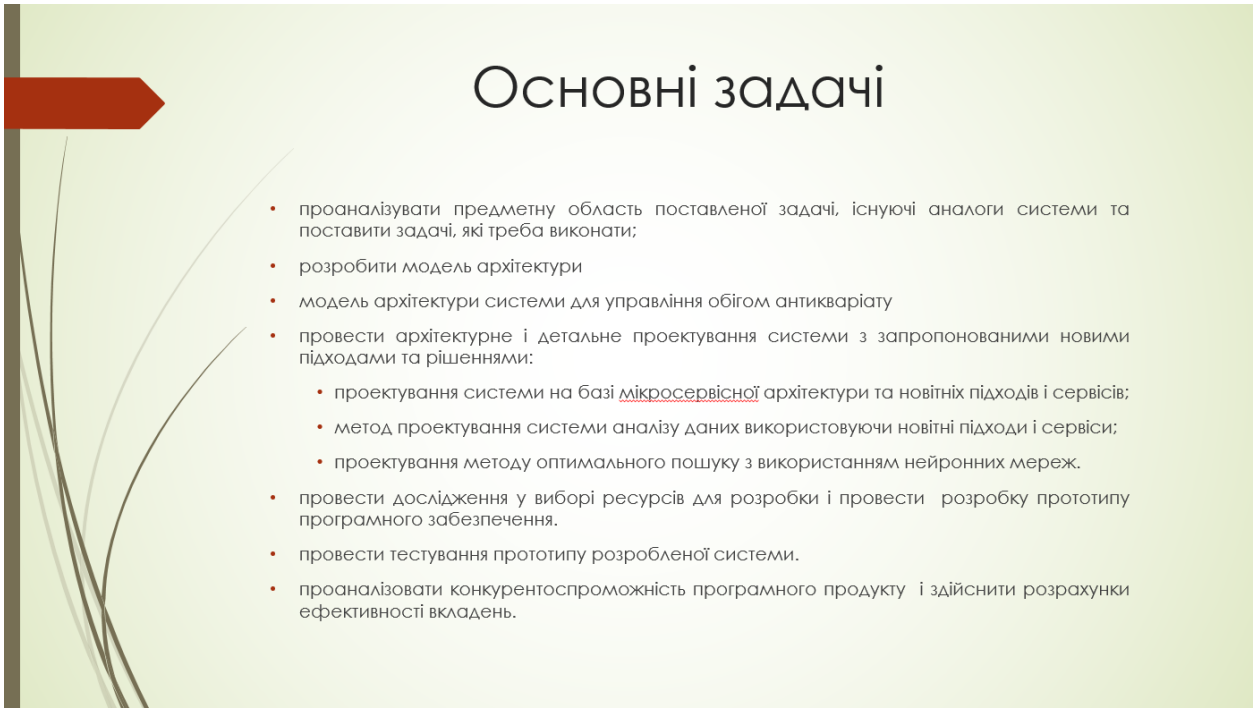
Плакат 1 - Тема, автор, науковий керівник магістерської кваліфікаційної роботи



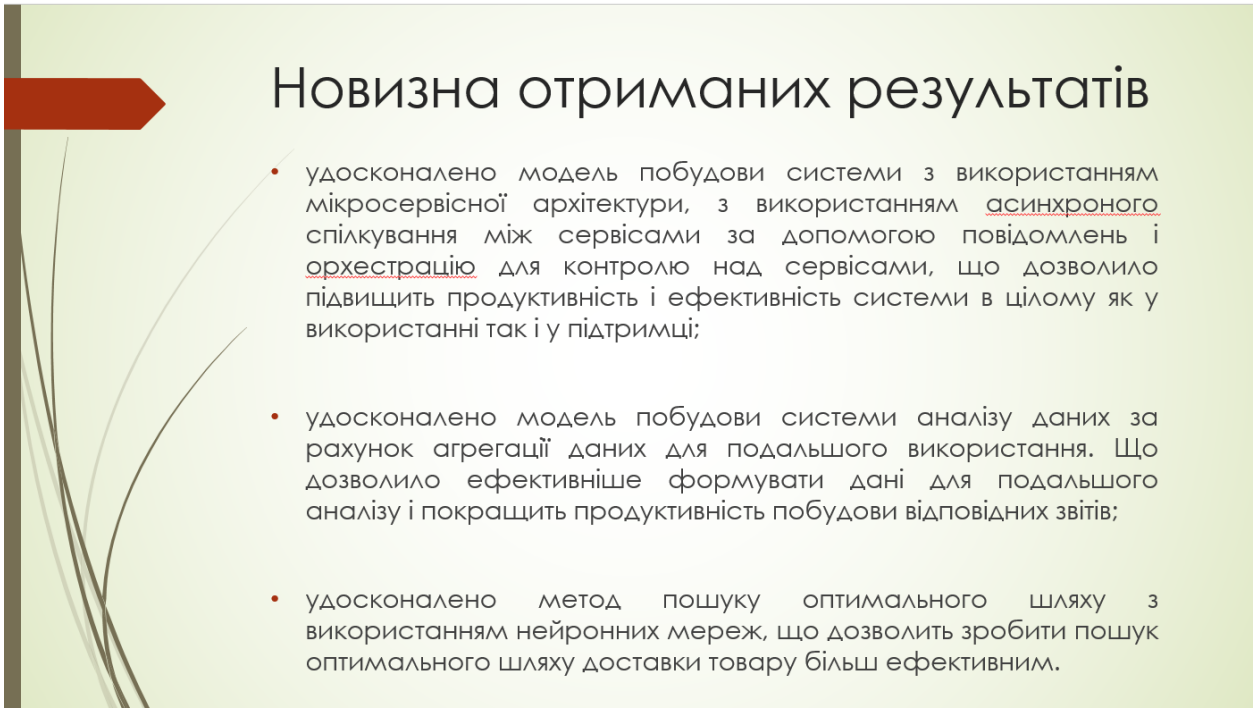
Плакат 2 - Мета, об’єкт, предмет



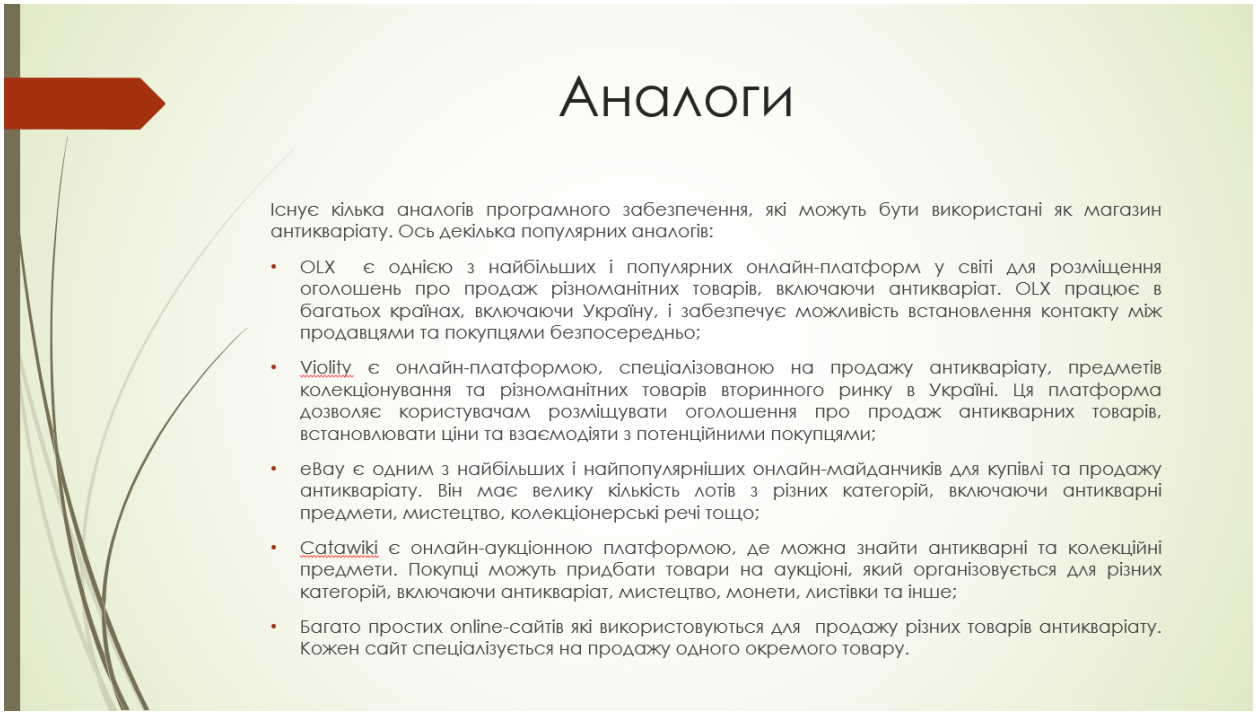
Плакат 3 - Основні задачі



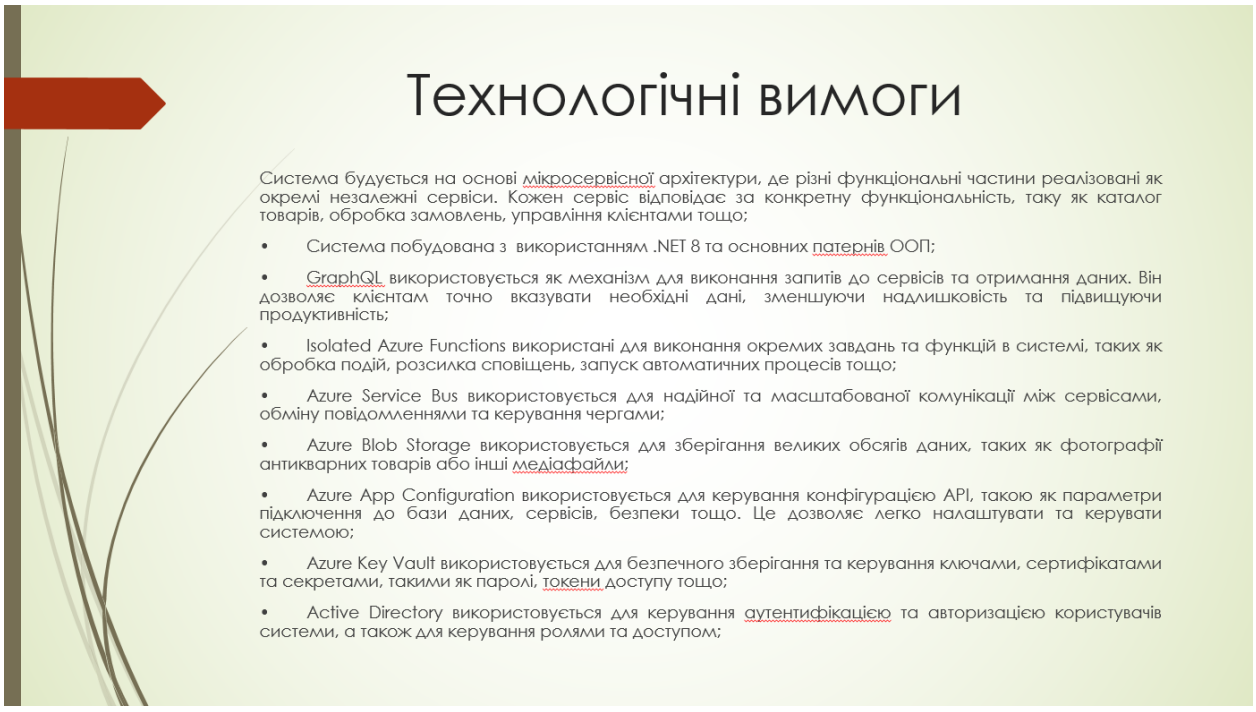
Плакат 4 - Новизна отриманих результатів



Плакат 5 – Аналоги

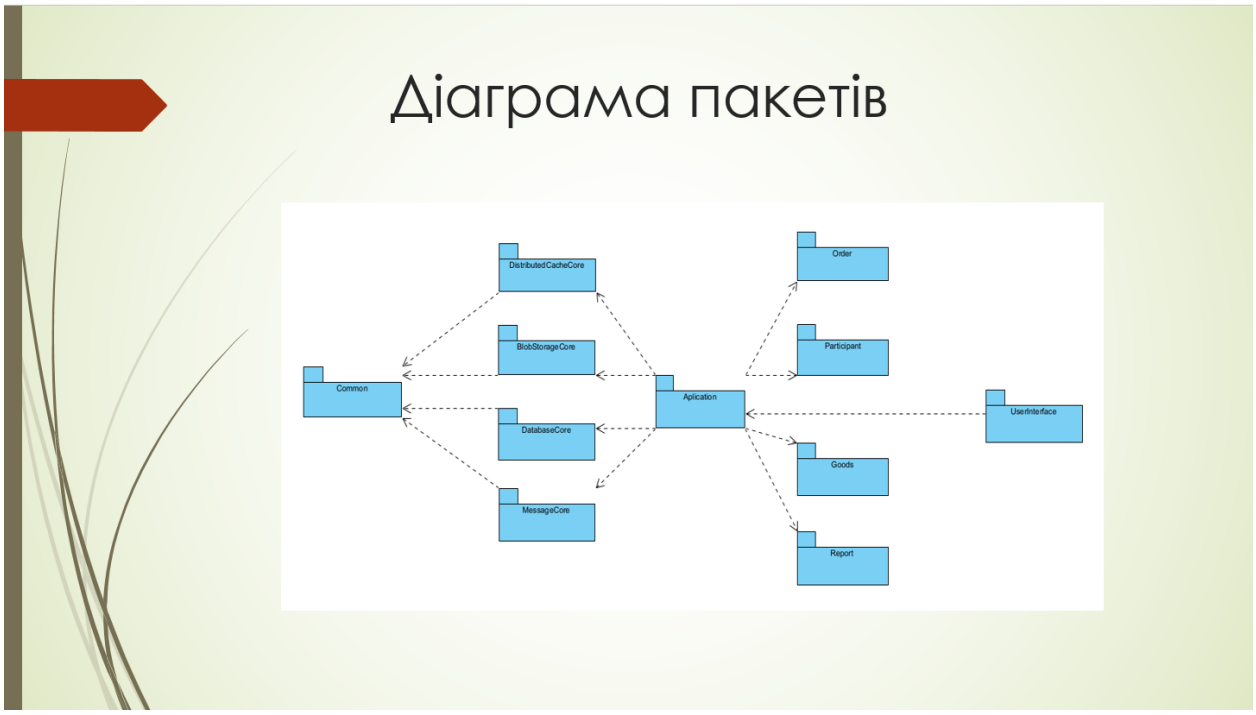


Плакат 6 – Технологічні вимоги



Плакат 7 – Архітектура системи

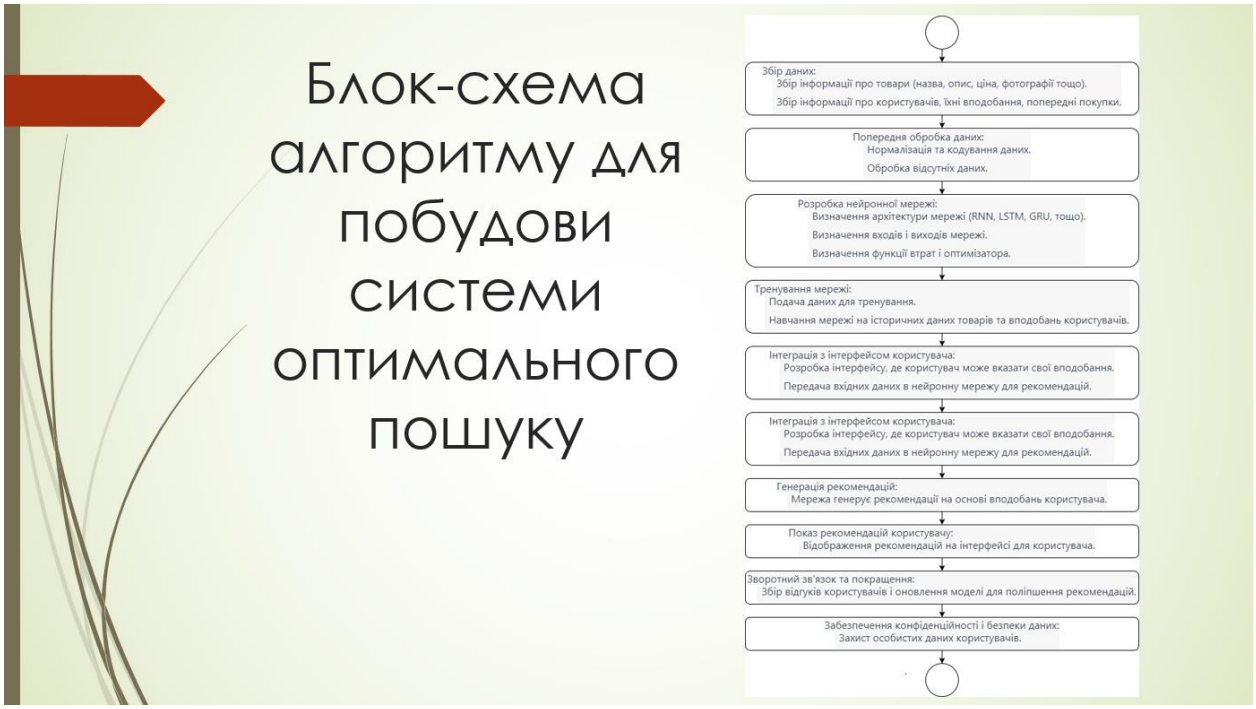
Плакат 8 – Діаграма пакетів



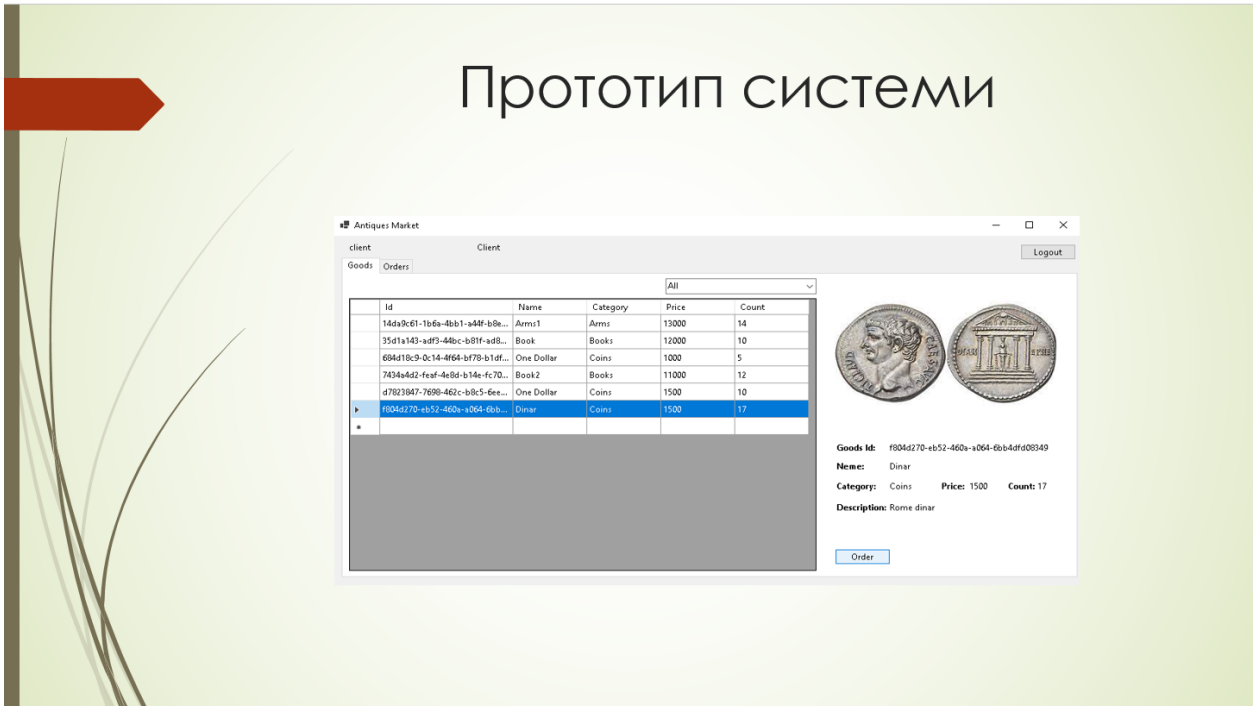
Плакат 9 – Діаграма розгортання

Плакат 10 – Архітектура системи аналізу даних

Плакат 11 – Блок-схема алгоритму для пошуку оптимальних товарів



Плакат 12 – Прототип системи



Плакат 13 – Результат роботи