Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Програмно технічний засіб керування кіберфізичною системою «Інтернет магазин» на основі рекомендаційних систем
Назва теми

КвРКІ 220029.22.01.12 ПЗ
Шифр

Галузь знань *12 «Інформаційні технології»*
Шифр, назва

Спеціальність *123 «Комп'ютерна інженерія»*
Шифр, назва

Освітня програма *«Комп'ютерна інженерія та програмування»*
Назва

Виконав: студент III курсу, група КІ2с-22-1 _____ Олексій НІКА
Підпис, дата — Ініціали, прізвище

Керівник _____ Євген ФЕДОРОВ
Підпис, дата — Ініціали, прізвище

Нормоконтролер _____ Тетяна КИСІЛЬ
Підпис, дата — Ініціали, прізвище

До захисту допускаю:
зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Ольга ПАВЛОВА
Підпис — Ініціали, прізвище

«12» червня 2025 р.

Хмельницький 2025

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Ольга ПАВЛОВА

" 10 " 01 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА

Олексію НІКА

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Програмно технічний засіб керування кіберфізичною системою «Інтернет магазин» на основі рекомендаційних систем

Керівник проекту (роботи) Євген ФЕДОРОВ, д.т.н., проф.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 07.02.2025 р. № 23

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2025 р.

3. Вихідні дані до проекту (роботи) Завдання на кваліфікаційну роботу

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Програмно технічний засіб адаптивного застосування кіберфізичної системи «Інтернет-магазин» та постановка задачі щодо її удосконалення

Програмно технічний засіб обробки інформації у кіберфізичній системі «Інтернет-магазин»

Програмно-апаратна реалізація адаптивного застосування кіберфізичної системи «Інтернет-магазин»

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Діаграма Sequence проєкту

Діаграма Class проєкту

Діаграма Flowchart проєкту

6. Консультанти розділів дипломного проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Тетяна КИСІЛЬ, доцент кафедри КІІС		
Антиплагіат	Андрій НІЧЕПОРУК, доцент кафедри КІІС		

7. Дата видачі завдання «_10_»_____01_____2025 р.

КАЛЕНДАРНИЙ ПЛАН

№з/п	Назва етапів (розділів) дипломного проекту (роботи)	Термін виконання етапів проекту (роботи)	Примітка
1	Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником	10.01.2025	виконано
2	Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження	01.02.2025	виконано
3	Робота над розділом 1 – дослідження предметної області та постановка задачі	01.03.2025	виконано
4	Робота над розділом 2 – проектування програмно технічного засобу для керування кіберфізичною системою «Інтернет магазин»	01.04.2025	виконано
5	Робота над розділом 3 – реалізація та тестування програмно технічного засобу для керування кіберфізичною системою «Інтернет магазин».	29.04.2025	виконано
6	Оформлення пояснювальної записки згідно вимог	25.05.2025	виконано
7	Попередній захист ВКР	26.05.2025	виконано
8	Захист ВКР на засіданні ЕК	Червень 2025 року	

Студент _____ Підпис _____ Олексій НІКА Ініціали, прізвище

Керівник роботи _____ Підпис _____ Євген ФЕДОРОВ Ініціали, прізвище

Зм	Арк	№
Розробив		
Перевір.		Ф
Н. контр.		Ки
Затв.		Павл

№ рядка	формат	Позначення	Найменування	Кіл. листів	№ екз	Примітка
			Текстові документи			
1		КвРКІ 220029.22.01.12 ПЗ	Пояснювальна записка	60		
			Графічні матеріали			
2		КвРКІ 220029.22.01.12 Е8	Архітектура ПЗ проєкту	1		
3		КвРКІ 220029.22.01.12 Е8	Архітектура ПЗ для кіберфізичної системи	1		
4		КвРКІ 220029.22.01.12 Е8	Апаратне забезпечення проєкту	1		

КвРКІ 220029.22.01.12 ВП

Зм	Арк	№ докум	Підпис	Дата
Розробив		Ніка		
Перевір.		Федоров		
Н. контр.		Кисіль		12.06.23
Затв.		Павлова		12.06.23

Відомість проекту

Літера	Аркуш	Аркушів
У	1	1

ХНУ, КІ2с-22-1

АНОТАЦІЯ

Тема кваліфікаційної роботи: «Програмно технічний засіб керування кіберфізичною системою «Інтернет магазин» на основі рекомендаційних систем.».

Автор роботи: Ніка Олексій Сергійович.

Керівник роботи: Федоров Євген Євгенович.

Пояснювальна записка: 65 с., 3 додатки, 35 рис., 40 джерел.

Графічна частина: 6 презентаційних слайдів.

КІБЕРФІЗИЧНА СИСТЕМА, ПОШУК, ПОШУКОВА СИСТЕМА, БАЗА ДАНИХ, МОНІТОРИНГ

Метою кваліфікаційної роботи є визначення умов і особливостей впровадження програмно-технічного засобу керування кіберфізичною системою «Інтернет-магазин» на основі рекомендаційних алгоритмів.

Об'єктом дослідження є функціонування програмно-технічних модулів інтернет-магазину.

Предметом дослідження є оцінка режимів застосування рекомендаційних систем у процесах управління контентом, обробки замовлень і взаємодії з кінцевим користувачем.

Для досягнення поставленої мети застосовано метод систематичного огляду літератури, що передбачає вивчення наукових публікацій та технічних стандартів.

Підпис студента

13.06.25

Дата

ЗМІСТ

ВСТУП

У сучасних умовах стрімкого розвитку інформаційно комунікаційних технологій та Інтернету речей кіберфізичні системи формують основу новітніх виробничих і сервісних платформ. Необхідно забезпечити безперервний обмін даними між фізичними об’єктами та програмними компонентами, що дозволяє підвищити ефективність управління технологічними процесами й адаптувати сервіси під індивідуальні потреби користувачів. Особливо актуальним є впровадження таких рішень у сфері електронної торгівлі, де від швидкості та якості обробки запитів залежить рівень задоволеності клієнтів і конкурентоспроможність бізнесу.

Автоматизація процесів керування інтернет магазином повинна включати не лише обробку замовлень і контроль логістики, а й інтелектуальний аналіз поведінки користувачів для формування персоналізованих рекомендацій. Випливає, що використання рекомендаційних систем є одним із ключових елементів модернізації електронної комерції. Рекомендаційні алгоритми дозволяють підвищити коефіцієнт конверсії, оптимізувати асортимент товарів і знизити витрати на маркетинг, водночас створюючи додаткову цінність для кінцевого споживача.

Метою даної кваліфікаційної роботи є розробка програмно технічного засобу керування кіберфізичною системою “Інтернет магазин” на основі рекомендаційних систем. Для досягнення мети необхідно спочатку вивчити існуючі архітектури кіберфізичних систем в електронній комерції, звернувши увагу на їх наклади для автоматизації замовлень, моніторингу та інтеграції IoT пристроїв. Паралельно слід проаналізувати сучасні рекомендаційні алгоритми (колаборативна, контентна та гібридні моделі), оцінюючи їх здатність працювати із великими даними та низькою затримкою. Надалі, на основі відібраних методів, створюється функціональна модель системи, що пояснює зв’язок між модулями збору подій користувачів, компонентом формування рекомендацій і обробкою замовлень із урахуванням

безпечного зберігання та маршрутизації даних. У межах реалізації проєкту формується прототип програмного модуля, який збирає поведінкові події, передає їх у сервіс машинного навчання та повертає персоналізовані пропозиції напряму в інтерфейс користувача через REST або GraphQL запити. Нарешті, слід провести тестування в умовах, максимально наближених до робочих: симулювати велику кількість запитів, виміряти час відгуку та точність рекомендацій (Precision@K, Recall@K) і, за потреби, оптимізувати алгоритми або архітектуру для досягнення необхідного рівня продуктивності та якості персоналізації.

У межах дослідження використовуються методи системного аналізу, моделювання програмно апаратних архітектур, алгоритмічного проєктування та експериментального тестування. Практичну значущість роботи визначає можливість інтеграції розробленого засобу в існуючі платформи електронної торгівлі, що сприятиме підвищенню рівня персоналізації обслуговування користувачів і оптимізації управлінських процесів.

Таким чином, реалізація поставлених завдань дасть змогу сформувати новий інструмент керування кіберфізичними процесами в інтернет магазині, який відповідає сучасним вимогам інтелектуалізації сервісів та сприяє підвищенню ефективності бізнес процесів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Обґрунтування актуальності теми роботи

У сучасних умовах цифровізації бізнесу та загострення конкуренції на ринку електронної торгівлі кіберфізичні системи [1, 2, 3, 4] повинні забезпечувати інтеграцію апаратних і програмних компонентів із реальними процесами обробки замовлень. Під електронним магазином у контексті кіберфізичної системи розуміють сукупність обладнання (сервери, мережеві пристрої, накопичувачі), програмного забезпечення (веб сервери, СУБД, бекенд сервіси [5, 6, 7]) і користувацьких інтерфейсів, які взаємодіють у режимі реального часу. Для досягнення високої надійності, масштабованості та безпеки такі системи повинні витримувати пікові навантаження та забезпечувати безперебійну обробку транзакцій, а також оперативне реагування на аномалії в роботі.

Випливає, що кіберфізична система “Інтернет магазин” повинна забезпечувати масштабовану обробку запитів користувачів та транзакцій із можливістю розширення ресурсу за потреби, водночас гарантувати високу доступність компонентів і стресостійкість інфраструктури навіть у пікові періоди навантаження. Окрім цього, система має інтегрувати модулі моніторингу та аналітики для безперервного контролю ключових показників продуктивності, таких як час відповіді сервера, навантаження на базу даних і відсоток успішно оброблених транзакцій. Крім технічних аспектів, важливо реалізувати інтерфейси, які забезпечать персоналізовану взаємодію з кінцевим користувачем, адаптуючи вміст і рекомендації відповідно до поведінки й уподобань клієнта.

У рамках теми кваліфікаційної роботи особливе місце посідають рекомендаційні системи як програмна складова, що відповідає за підвищення залученості клієнтів і збільшення конверсії. Рекомендаційні алгоритми базуються на обробці великих обсягів даних, зокрема:

– Історії переглядів і покупок;

– рейтингових оцінок товарів;

– профільних характеристик користувача;

– поведінкових патернів (кластеризація та сегментація клієнтської бази).

Найпоширенішими підходами до реалізації рекомендаційних систем є фільтрація за контентом, яка спирається на метадані товарів (категорії, атрибути, опис) і профілі інтересів користувачів (історія переглядів, вподобання, виписки з корзини). Такий підхід дозволяє порівнювати характеристики товарів та підбирати схожі з урахуванням того, що конкретно зацікавило клієнта (наприклад, колір, бренд, технічні параметри), однак при цьому страждає на проблему «холодного старту» - коли новому користувачу або новому товару бракує достатньої кількості даних для точних рекомендацій. Колаборативна фільтрація ґрунтується на схожості поведінкових патернів користувачів або товарних елементів: аналізуючи, які товари купували або оцінювали подібні до поточного користувачі, система пропонує продукцію, що може зацікавити. Цей підхід показує хороші результати при достатній кількості рейтингових або транзакційних даних, але водночас схильний до проблем із масштабуванням, коли база користувачів чи асортимент товарів ростуть, а також може пропонувати лише популярні товари, що відкидає нішеві продукти. Гібридні методи поєднують обидва підходи: з одного боку, вони враховують атрибути товарів і профілі користувачів, а з іншого - спираються на спільні оцінки чи покупки. Це дає змогу компенсувати недоліки кожного методу: якщо для нової позиції бракує колаборативних даних, система використовує content based алгоритм, а коли немає достатньої інформації про нових користувачів, звертається до патернів колаборативної фільтрації. Зокрема, при побудові гібридного підходу до уваги беруть вагові коефіцієнти, що змінюються залежно від наявності даних, і регулярне перенавчання моделей, аби вчасно адаптуватися до змін у поведінці клієнтів та оновлення асортименту.

Для забезпечення стабільності функціонування та своєчасного аналізу роботи всієї системи необхідно впровадити багаторівневий моніторинг. На апаратному рівні відстежують ключові показники завантаження процесора,

використання оперативної пам’яті, стан дискових масивів (вільне місце, I/O операції) та пропускну здатність мережевих інтерфейсів (затримки, втрати пакетів). Це дозволяє виявляти вузькі місця та планувати горизонтальне чи вертикальне масштабування інфраструктури перед тим, як користувачі відчують уповільнення або перебої. На програмному рівні фіксують час відповіді сервера на кожен HTTP запит, кількість активних сесій, число одночасно оброблюваних транзакцій, навантаження на базу даних (кількість запитів на секунду, час виконання складних SQL запитів) і продуктивність сервісів, що формують рекомендації (затримка при отриманні результатів, використання ресурсів сервісу машинного навчання). Це дає змогу швидко виявляти деградацію сервісів, викликану проблемами в коді, неефективними запитами чи змінами в профілі навантаження. На рівні користувацької взаємодії необхідно збирати події клієнтів - перегляди товарів, кліки, додавання у кошик, оформлення замовлень, залишені відгуки чи рейтинги - та аналізувати їхню послідовність. Така аналітика допомагає виявляти аномалії в поведінці (наприклад, підозрілі сплески активності) і коригувати алгоритми рекомендацій у режимі реального часу, а також оптимізувати інтерфейс, якщо користувачі відмовляються від оформлення замовлення на певному етапі. Зібрана інформація з усіх рівнів моніторингу інтегрується в єдині дашборди (Prometheus + Grafana, ELK стек), що забезпечує операторів та DevOps інженерів можливістю оперативно реагувати на будь які відхилення та підтримувати належний рівень SLA.

Забезпечення безпеки даних і захист від шахрайських операцій повинні бути пріоритетними складовими проєктованої системи. Необхідно застосовувати методи шифрування з’єднань, автентифікації користувачів, механізми виявлення аномальної активності та резервного копіювання інформації. Аналіз показує, що готові рішення можуть бути ефективними на ранніх етапах розробки, однак при масштабуванні, необхідності глибокої персоналізації та інтеграції з промисловими внутрішніми системами - виникає потреба у створенні індивідуального, програмно технічного рішення. Власна розробка, на відміну від готових рішень, дозволяє:

– Повністю адаптувати систему під потреби підприємства;
– інтегрувати з внутрішніми сервісами;
– забезпечити гнучкість в побудові рекомендаційної системи;
– досягти вищої безпеки та контролю над даними.

У підсумку, аналіз предметної області демонструє, що розробка програмно технічного засобу керування кіберфізичною системою “Інтернет магазин” на основі рекомендаційних систем повинна враховувати архітектурні особливості масштабованих платформ, алгоритмічні підходи до персоналізації, а також багаторівневий моніторинг і засоби забезпечення інформаційної безпеки. Це створює основу для подальшого проєктування та реалізації прототипу, здатного відповідати сучасним вимогам електронної торгівлі.

1.2 Порівняльний аналіз переваг та недоліків існуючих рішень

У сучасних умовах електронна комерція повинна спиратися на гнучкі програмні платформи, які забезпечують швидке розгортання та адаптацію бізнес логіки. Готові системи керування контентом (CMS), такі як Shopify, Magento, WooCommerce та OpenCart, дозволяють прискорити запуск інтернет магазину та забезпечують базовий набір функцій: обробку замовлень, керування каталогом, SEO оптимізацію та інтеграцію з платіжними шлюзами. Проте вибір такої платформи випливає з потреб підприємства: великі каталоги товарів і високі пікові навантаження не повинні обмежуватися ресурсами та можливостями хмарних сервісів або безкоштовних плагінів. У разі потреби в глибокій персоналізації та унікальних бізнес процесах готові рішення часто потребують доопрацювання ядра або додаткових модулів, що призводить до зростання складності супроводу. Необхідно враховувати, що при використанні сучасних фронтенд фреймворків, зокрема React JS у поєднанні з Next.js [8, 9, 10] для серверного рендерингу, забезпечується висока швидкодія інтерфейсу та SEO дружність проекту. Дозволяється тільки за умови наявності експертизи в JavaScript екосистемі

застосовувати ці інструменти, оскільки часті оновлення пакетів і необхідність налаштування таких компонентів, як Webpack та Babel, можуть уповільнювати цикл розробки. На бекенді однопотокова модель Node.js повинна застосовуватися для I/O інтенсивних задач, тоді як для CPU важких обчислень випливає необхідність використання мікросервісів або альтернативних середовищ (наприклад, Spring Boot або Django), щоб уникнути вузьких місць.

Рекомендаційні бібліотеки LightFM і LensKit дозволяють реалізувати гібридні підходи, які поєднують колаборативну та контентну фільтрацію, забезпечуючи високу точність рекомендацій навіть за обмежених даних користувача. Однак масштабування таких алгоритмів повинно базуватися на розподілених обчисленнях та оптимізації sparse матриць, інакше зростання обсягу даних призведе до значного збільшення часу обробки. Більш прості пакети, наприклад Surprise, рекомендується використовувати лише для прототипування завдяки обмеженій підтримці гібридних моделей і недостатній чисельності документації щодо продуктивності на великих масивах даних. На рівні інфраструктури власні сервери (on premises) забезпечують повний контроль над безпекою та конфіденційністю даних, але потребують значних капітальних вкладень і спеціалізованого персоналу для обслуговування. Натомість хмарні рішення (AWS, Azure, GCP) випливають із потреби в гнучкості та динамічному масштабуванні: ресурси можуть автоматично підлаштовуватися під навантаження, а вбудовані сервіси моніторингу й резервного копіювання забезпечують відмовостійкість. Водночас для нейтралізації ризиків витоку даних і непередбачених затрат необхідно застосовувати багаторівневий захист з використанням SSL сертифікатів, MFA аутентифікації та систем виявлення аномалій.

Ми наочно бачимо, що кожному компоненту системи властиві власні виклики та вимоги, і для їхнього успішного вирішення необхідно підбирати відповідні інструменти. Такий підхід дозволяє своєчасно помічати та усувати недоліки, перш ніж вони вплинуть на загальну роботу. При цьому важливо

зберігати гнучкість архітектури, щоб швидко адаптуватися до нових умов. Застосування різних рішень допомагає уникати ризиків, пов'язаних із перевантаженням чи відмовами окремих модулів. Таким чином, комплексне поєднання технологій гарантує стабільність та можливість розвитку системи в майбутньому.

Таблиця 1.1 – Переваги та недоліки різних програмних підходів

Технологія	Переваги	Недоліки
React.js	Висока продуктивність завдяки віртуальному DOM; компонентна архітектура для повторного використання; багата екосистема (Redux, React Router)	Необхідність налаштування збірки (Webpack, Babel); часті оновлення, що можуть порушувати сумісність
Next.js	Серверний рендеринг для SEO оптимізації; вбудована маршрутизація та розподіл коду; підтримка статичної генерації	Збільшений обсяг залежностей; складніше налаштування кастомних конфігурацій
Node.js + TypeScript	Єдина мова на фронтенді та бекенді; висока швидкодія для I/O операцій; типізація знижує кількість помилок	Однопотокова модель обмежує CPU інтенсивні задачі; потребує додаткових інструментів для масштабування

Кінець таблиці 1.1

Django (Python)	«батарейки включені» (ORM, панель адміністрування, система аутентифікації); високий рівень безпеки; велика спільнота	Збільшені витрати ресурсів для real time застосунків; повільніший за Node.js у I/O інтенсивних задачах
Flask (Python)	Легковаговий фреймворк для швидкого прототипування; гнучкість у виборі компонентів; проста інтеграція з бібліотеками ML	Відсутність вбудованих функцій (ORM, аутентифікація) - потребує додаткових розширень
WordPress+WooCommerce	Швидкий старт без навичок програмування велика кількість тем і плагінів активна спільнота	Обмежена масштабованість при високих навантаженнях; потреба платних плагінів для розширення функціоналу
Spring Boot (Java)	Потужна екосистема Spring; стабільність та надійність; підтримка мікросервісів	Висока складність початкового налаштування; тривале навчання для розробників Java

Підсумовуючи, випливає, що для реалізації програмно технічного засобу керування кіберфізичною системою “Інтернет магазин” на основі рекомендаційних систем доцільно поєднати гібридний підхід у виборі технологій: frontend [11, 12, 13] на базі React JS із Next.js для SEO оптимізації, бекенд на Node.js із TypeScript або Spring Boot для CPU інтенсивних модулів, бібліотеку LightFM для рекомендаційних алгоритмів та хмарну інфраструктуру з гнучким масштабуванням. Така комбінація забезпечить необхідну продуктивність, безпеку й масштабованість з мінімальними витратами на доопрацювання та підтримку.

1.3 Підходи до вирішення задачі за темою дослідження

У ході розробки програмно технічного засобу керування кіберфізичною системою “Інтернет магазин” на основі рекомендаційних систем необхідно обрати відповідну архітектурну парадигму. Застосовують багаторівневий підхід із поділом на шар представлення, шар бізнес логіки та шар доступу до даних.

На рівні представлення повинен бути реалізований фронтенд на базі React JS із серверним рендерингом через Next.js для забезпечення SEO дружності та швидкого відгуку інтерфейсу. Шар бізнес логіки має випливати з потреб у масштабованості: рекомендується створювати окремі сервіси для обробки транзакцій, моніторингу й рекомендацій, що дозволяє ізолювати критичні функції та підтримувати відмовостійкість. Доступ до даних випливає з вимоги зберігання історії дій користувачів і товарних атрибутів; оптимальним рішенням є поєднання реляційної СУБД для транзакцій та NoSQL [13, 14, 15, 16] сховища для аналітичного навантаження. Рекомендаційні алгоритми мають базуватися на аналізі великих обсягів даних, тому підхід до їх реалізації повинен охоплювати дві взаємодоповнюючі стратегії: фільтрацію за контентом і колаборативну фільтрацію з гібридною обробкою. У контентній фільтрації необхідно використовувати векторне представлення товарів на основі їхніх атрибутів і метаданих, що дозволяє швидко знаходити схожі елементи. Колаборативна фільтрація повинна

застосовувати матричну факторизацію або моделі на основі латентних факторів, які витягують патерни спільної поведінки користувачів. Гібридний підхід випливає з необхідності компенсувати холодний старт нових товарів і користувачів, тому алгоритми повинні поєднувати результати обох методів із ваговими коефіцієнтами та регулярним перенавчанням моделей. Для обробки даних потрібно спроектувати конвеєр ETL, що включає отримання потоків подій з веб інтерфейсу (клацання, перегляди, замовлення), їхнє очищення та перетворення у формат, придатний для навчання моделей. Випливає необхідність підтримувати як пакетну обробку великих обсягів історичних даних у середовищі Apache Spark, так і стрімінгові обчислення в режимі реального часу через Apache Kafka або AWS Kinesis. Такий підхід дозволяє проводити онлайн оновлення рекомендацій при зміні поведінки користувачів, а також виконувати глибинну аналітику для побудови дашбордів із ключовими метриками. Безперебійний контроль роботи системи повинен забезпечуватися через багаторівневий моніторинг. Перш за все, необхідно використовувати рішення Prometheus + Grafana для збору метрик серверів і контейнерів; вторинною умовою є інтеграція логів у систему ELK (Elasticsearch, Logstash, Kibana) для аналізу подій і виявлення аномалій. Випливає необхідність встановити алерти на перевищення порогів CPU, пам’яті та затримки відповідей API, що дозволяє оперативно реагувати на відмови або деградацію продуктивності.

Забезпечення безпеки даних і каналів зв’язку повинно здійснюватися за допомогою TLS шифрування, механізмів OAuth2 для аутентифікації користувачів і ролей, а також регулярного аудиту вразливостей із використанням автоматизованих сканерів (наприклад, OWASP ZAP). Необхідно інтегрувати WAF (Web Application Firewall) для захисту від поширених атак типу SQL ін’єкцій і XSS, а також організувати резервне копіювання даних із збереженням у георозподілених сховищах для запобігання втраті інформації в разі збою або кібератаки.

Остаточна реалізація повинна базуватися на контейнеризації всіх компонентів із використанням Docker та оркестрації Kubernetes, що дозволяє масштабувати окремі сервіси за навантаженням і проводити rolling оновлення без

простоїв. Для CI/CD [17, 18, 19, 20] необхідно впровадити автоматизовані пайплайни на базі GitLab CI або GitHub Actions, які повинні включати кроки побудови образів, прогін unit тестів і статичного аналізу коду перед деплоєм у середовище staging та production.

Ці підходи повинні тісно взаємодіяти та гармонійно інтегруватися, щоб система працювала стабільно й ефективно. Конкретний вибір залежить від особливостей інтернет магазину, обсягу каталогу, кількості користувачів і рівня персоналізації. Крім того, варто передбачити можливість масштабування функціоналу для адаптації до зростаючих вимог і змін ринку. Застосування викладених підходів дозволяє створити єдиний програмно технічний комплекс, який забезпечує персоналізацію користувацького досвіду за допомогою сучасних рекомендаційних алгоритмів, одночасно підтримує високу надійність, безпеку та ефективне управління інфраструктурою.

1.4 Постановка задачі

У межах даної кваліфікаційної роботи необхідно сформувати чітке уявлення про функціональні компоненти кіберфізичної системи “Інтернет магазин” на основі рекомендаційних алгоритмів, а також визначити послідовність етапів дослідження та розробки. Передбачається провести теоретичний аналіз предметної області, окреслити найактуальніші проблеми в галузі електронної комерції та рекомендаційних систем і випливати з потреби знайти ефективні шляхи їх вирішення, для чого було виявлено базові вимоги щодо нашого програмного забезпечення.

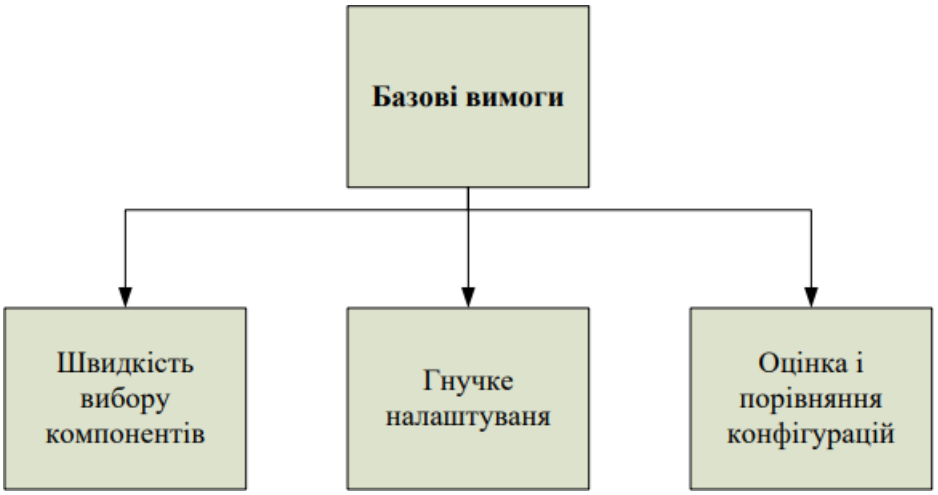


Рисунок 1.1 – Базовані вимоги щодо функцціональності

Об’єктом є процес керування кіберфізичноою системою «Інтернет магазин». Предметом є рекомендаційні системи для керування кіберфізичною системою «Інтернет магазин».

Метою даної кваліфікаційної роботи є розробка програмно технічного засобу керування кіберфізичною системою “Інтернет магазин” на основі рекомендаційних систем. Для досягнення поставленої мети вирішуються такі завдання:

− проаналізувати існуючі підходи до побудови кіберфізичних систем в електронній комерції та їхні архітектурні особливості;

− дослідити сучасні методи й алгоритми рекомендаційних систем із урахуванням вимог до швидкодії та масштабованості;

− розробити функціональну модель програмно технічного засобу управління на базі обраних рекомендаційних алгоритмів;

− реалізувати прототип програмного модуля інтеграції рекомендацій у процес обробки замовлень та взаємодії з користувачем;

− провести експериментальне дослідження продуктивності та точності побудованих рекомендацій у тестовому середовищі.

Результати теоретичних досліджень свідчать про доцільність поєднання React JS із Next.js для клієнтського інтерфейсу, Node.js або Spring Boot для бекенд сервісів, а також LightFM для алгоритмів персоналізації. Застосування контейнеризації та CI/CD пайплайнів забезпечить масштабованість і відмовостійкість. Реалізація запропонованих підходів дозволить досягти високої продуктивності, гнучкості і зручності взаємодії з користувачем, що відповідає сучасним вимогам цифрової трансформації бізнесу.

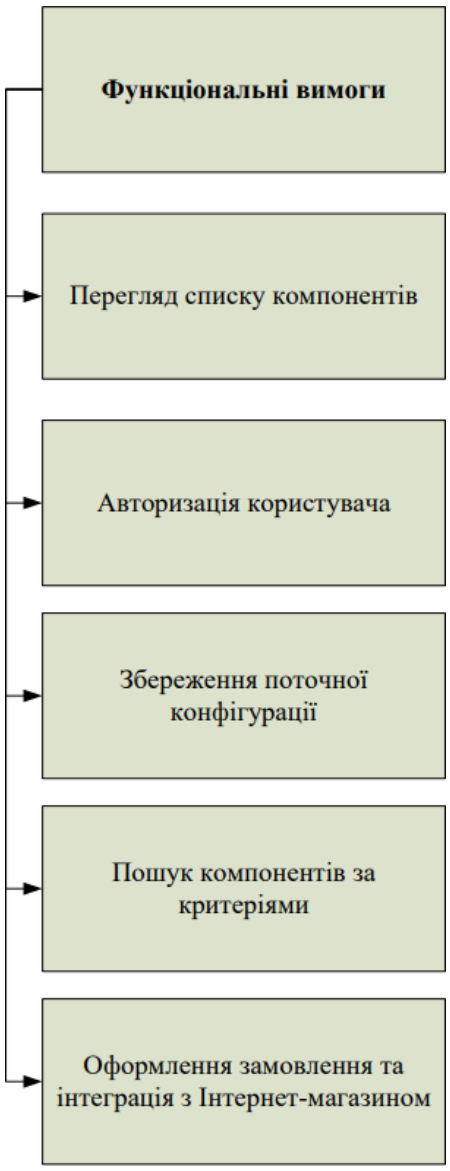


Рисунок 1.2 – функціональні вимоги керування кіберфізичною системою «Інтернет магазин»

При розробці та подальшому керуванні кіберфізичною системою “Інтернет магазин” нам також необхідно передбачити нефункціональні вимоги, які є також важливими у роботі. Нефункціональні вимоги відіграють велику роль у взаємодії з програмними засобами як зі сторони користувачів, та і з сторони розробника для зручного керування та подальшого покращення.

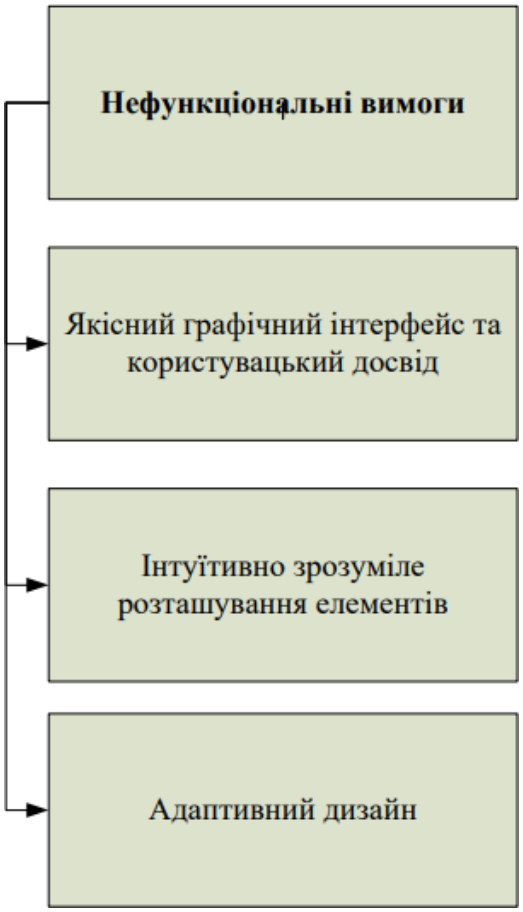


Рисунок 1.3 – Основні нефункціональні вимоги програмного засобу

На підставі викладеного завданням є розробка прототипу програмно технічного засобу для керування кіберфізичною системою «Інтернет магазин», який повинен забезпечувати інтеграцію модулів обробки транзакцій, моніторингу продуктивності та формування персоналізованих рекомендацій, а також надавати інструментарій для подальшого розгортання в реальному середовищі. Висновки та

рекомендації мають випливати з результатів виконаних досліджень і експериментів.

1.5 Висновки

Підсумовуючи проведений аналіз, випливає, що інтеграція кіберфізичних підходів в електронну торгівлю, зокрема через застосування рекомендаційних алгоритмів, є надзвичайно актуальною для підвищення адаптивності та конкурентоспроможності інтернет магазину. У межах розділу «Аналіз предметної області» обґрунтовано складові програмно технічної архітектури, акцентовано на необхідності багаторівневого моніторингу й забезпеченні безпеки. У розділі «Порівняльний аналіз» визначено переваги готових рішень і сучасних фреймворків, а також їхні обмеження при масштабуванні й персоналізації. У розділі «Підходи до вирішення задач» окреслено оптимальну архітектуру з поділом на шари представлення, бізнес логіки та даних, а також описано конвеєри обробки даних, стратегії рекомендацій та засоби забезпечення безперебійної роботи.

Випливає необхідність розробки спеціалізованого програмно технічного засобу керування кіберфізичною системою “Інтернет магазин” на основі гібридних рекомендаційних методів. Результати теоретичних досліджень свідчать про доцільність поєднання React JS із Next.js для клієнтського інтерфейсу, Node.js або Spring Boot для бекенд сервісів, а також LightFM для алгоритмів персоналізації. Застосування контейнеризації та CI/CD пайплайнів забезпечить масштабованість і відмовостійкість. Реалізація запропонованих підходів дозволить досягти високої продуктивності, гнучкості і зручності взаємодії з користувачем, що відповідає сучасним вимогам цифрової трансформації бізнесу.

2 ПРОЕКТУВАННЯ ПРОГРАМНО ТЕХНІЧНОГО ЗАСОБУ ДЛЯ КЕРУВАННЯ КІБЕРФІЗИЧНОЮ СИСТЕМОЮ «ІНТЕРНЕТ МАГАЗИН»

2.1 Опис компонентів кіберфізичною системою «Інтернет магазин».

Проєктована система «Інтернет магазин» складається з низки взаємопов’язаних компонентів, кожен із яких виконує свою ключову функцію і разом утворює цілісний кіберфізичний комплекс. У центрі розташовані сервери обробки транзакцій та зберігання даних, де реляційна СУБД зберігає інформацію про замовлення, клієнтські профілі та товарні каталоги, а NoSQL сховище призначене для швидкої аналітики й накопичення великих обсягів логів. Для масштабування обчислень, пов’язаних із рекомендаціями, застосовуються виділені обчислювальні вузли з GPU акселераторами, які прискорюють навчання та оновлення моделей машинного навчання.

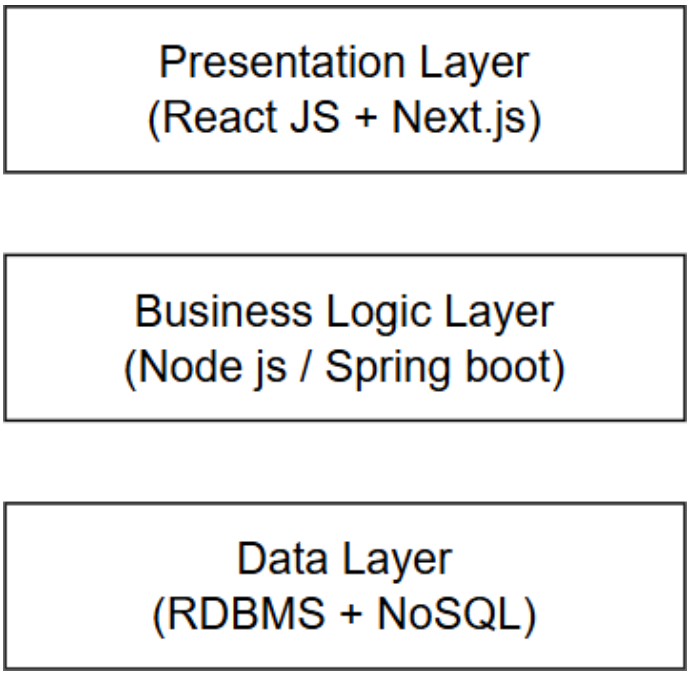


Рисунок 2.1 – Багаторівнева архітектура системи

Клієнтські інтерфейси реалізовано за допомогою React JS із Next.js, що забезпечує швидку генерацію сторінок на сервері й дружність. Цей шар представлення динамічно адаптується під різні пристрої та оперативно реагує на взаємодію користувача - від перегляду товару до оформлення покупки. Безпосередньо в браузері збираються події користувацької активності (кліки, перегляди, пошукові запити), які надсилаються на бекенд через стандартні REST і GraphQL запити. Шар бізнес логіки сконструйовано як набір мікросервісів на Node.js із TypeScript [21, 22, 23, 24]. Кожен сервіс відповідає за окрему бізнес функцію: один обробляє кошик і обліковує транзакції, інший керує користувацькою аутентифікацією та ролями, а третій - керує інтеграцією зі сторонніми платіжними шлюзами й службами доставки. Для складніших CPU інтенсивних завдань, як от генерація звітів або обробка зображень товарів, передбачено сервіси на Spring Boot, що можуть працювати паралельно із основним елементом.

Рекомендаційний компонент реалізовано як окремий сервіс LightFM, який через REST API [25, 26, 27] приймає зведені дані про поведінку користувачів та атрибути товарів, обчислює ваги латентних факторів і повертає відфільтрований і відсортований список продуктів. Для збирання й підготовки даних застосовується конвеєр ETL: Apache Spark відповідає за пакетну обробку історичних даних, тоді як Apache Kafka забезпечує стрімінг подій у реальному часі. Такий підхід дозволяє постійно оновлювати рекомендації й негайно реагувати на зміну поведінки клієнтів.

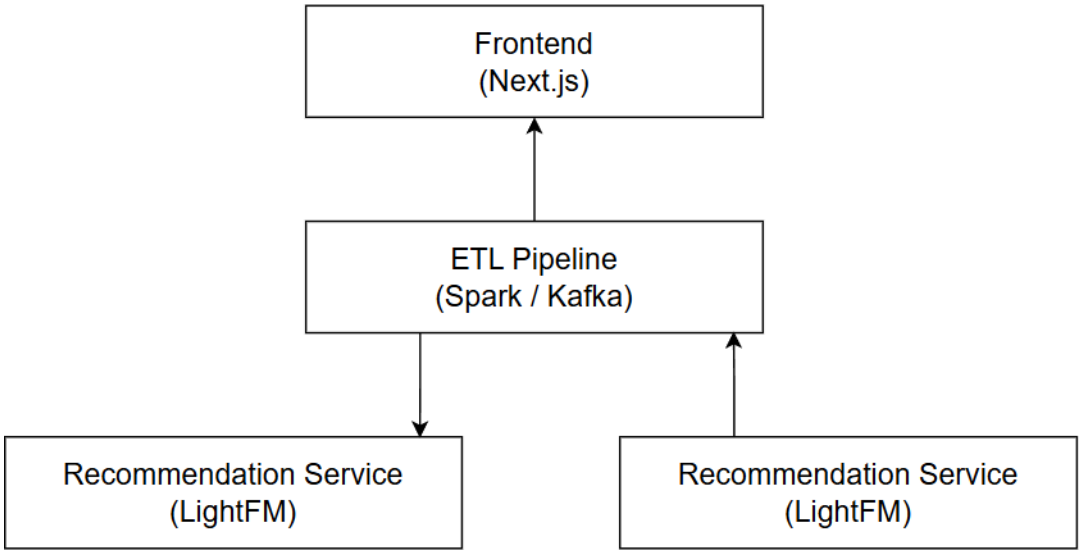


Рисунок 2.2 – Потік даних між компонентами

Моніторинг і логування організовано за допомогою Prometheus та Grafana для метрик продуктивності контейнерів і серверів, а також ELK стека для глибинного аналізу логів і пошуку аномалій. Налаштовані алерти на перевищення критичних порогів CPU, пам’яті чи часу відповіді API гарантують швидке реагування служби підтримки на потенційні збої.

Програмний інтерфейс (API – Application Programming Interface) – це набір визначених правил, протоколів і методів, які регламентують взаємодію різних програмних компонентів. За допомогою API одне програмне забезпечення може звертатися до функціональності іншого, не потребуючи знань про внутрішню реалізацію чи структуру цього коду.

Інтерфейс масштабується від окремого модуля в межах одного додатка (наприклад, клас із набором публічних методів, які виробляють певну логіку) до мережного API, яке надає доступ через HTTP-запити до сервісів, що працюють на віддалених серверах. У будь-якому випадку сутність API полягає в тому, щоб чітко описати, як саме користувач (розробник або інший сервіс) може відправити запит, які дані йому необхідно передати та якою буде ситуація з обробкою відповіді.

Наприклад, у веб-інтерфейсах RESTful API визначає структуру URL-ендоінтів, типи HTTP-методів (GET, POST, PUT, DELETE), формат запитів (JSON, XML) і формат відповідей із кодами стану (200 – успіх, 404 – не знайдено, 500 – внутрішня помилка сервера). У такий спосіб клієнтський додаток просто викликає потрібний ендпоінт [39, 40], передає необхідні параметри й отримує готовий результат, не занурюючись у подробиці реалізації бекенду.

Основна користь програмного інтерфейсу в тому, що він дозволяє розробникам створювати модульний, адаптивний і розширюваний код. Завдяки чіткому визначенню API можна змінювати внутрішню реалізацію сервісів, не торкаючись логіки тих клієнтів, які вже звикли до поточної версії інтерфейсу. Таким чином, програмний інтерфейс слугує своєрідним контрактом між різними частинами системи, забезпечуючи узгодженість, передбачуваність і можливість безболісного оновлення окремих компонентів.

Інтеграційні інтерфейси передбачають гнучкі webhook підписки, які інформують внутрішні ERP і CRM системи про кожен етап обробки замовлення, а також стандартизовані API для мобільних додатків і зовнішніх сервісів. Безпека забезпечується на транспортному рівні через TLS/HTTPS, на рівні аутентифікації через OAuth2 із ролями, а в сховищах даних застосовується шифрування AES 256 як «на льоту», так і в стані спокою. Додатково впроваджено Web Application Firewall для захисту від SQL ін’єкцій та XSS, а резервне копіювання виконується автоматично в георозподілені сховища.

ОСервер транзакцій із реляційною СУБД оброблює всі операції з кошиком, замовленнями та профілями користувачів, гарантує ACID-транзакції й оперативне оновлення таблиць замовлень та товарів. Аналітичне сховище на базі NoSQL зберігає логи, події поведінки користувачів і дані для машинного навчання, що дозволяє швидко аналізувати великі обсяги інформації без навантаження на основну БД.

Мікросервіс обробки кошика та транзакцій на Node.js відповідає за логіку додавання товарів у кошик, валідацію цін і наявності, обчислення знижок, ініціацію

оплати й оновлення статусу замовлення. Сервіс генерації звітів і обробки зображень на Spring Boot виконує ресурсомісткі завдання: підготовку SQL- та NoSQL-агрегатів, формування PDF-звітів і оптимізацію зображень товарів за розкладом, відокремлюючи аналітику від транзакційної логіки.

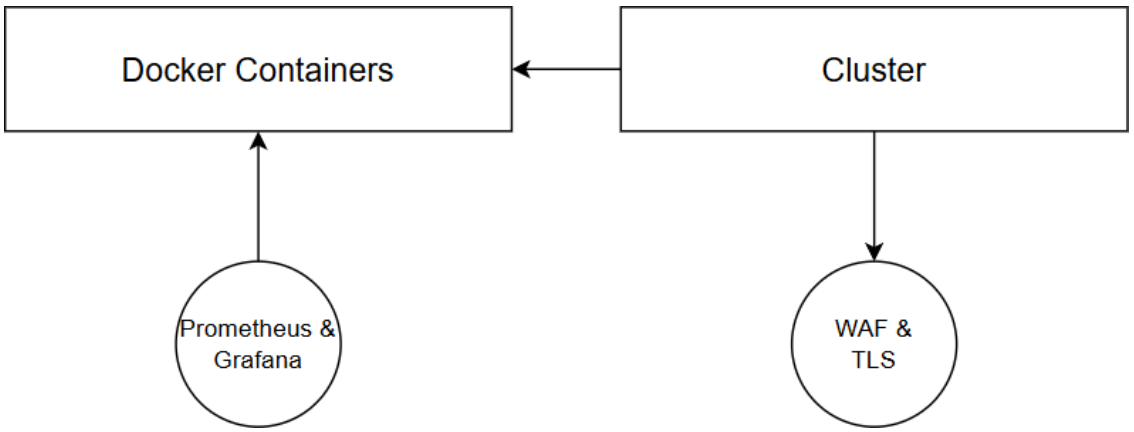


Рисунок 2.3 – Огляд інфраструктури кіберфізичної системи

Усі компоненти тісно взаємодіють між собою, обмінюючись даними через стандартизовані протоколи, що забезпечує стабільність, високу продуктивність і готовність до масштабування відповідно до зростання каталогу товарів чи числа користувачів. Конкретний вибір технологій і конфігурацій буде визначатися вимогами до швидкодії, надійності та рівня персоналізації, забезпечуючи ефективну та безпечну роботу системи в реальному середовищі.

2.2 Розробка архітекиури кіберфізичної системи «Інтернет магазин».

У процесі розробки архітектури кіберфізичної системи «Інтернет магазин» необхідно забезпечити гармонійну взаємодію між апаратними, програмними та мережевими компонентами, виходячи з принципів модульності, масштабованості й відмовостійкості. Архітектура повинна базуватися на мікросервісному підході, де кожен сервіс відповідає за окрему бізнес функцію (обробка замовлень, облік

клієнтів, генерація рекомендацій тощо), а взаємодія здійснюється через стандартизовані API інтерфейси та повідомлення в чергах.

Першим етапом є визначення контекстів доменної моделі (Bounded Context). Виділяють принаймні три ключові контексти:

– Керування користувачами та ролями: аутентифікація, авторизація, управління правами доступу;

– обробка замовлень: кошик, облік транзакцій, взаємодія з платіжними та логістичними шлюзами;

– рекомендаційна система: збір поведінкових даних, тренування моделей, побудова персоналізованих списків товарів.

Сервіси аутентифікації й авторизації повинні працювати незалежно від сервісів обробки транзакцій, а модуль рекомендацій - ізольовано від основного потоку замовлень, щоб забезпечити низьку затримку та можливість горизонтального масштабування.

Далі необхідно спроектувати схему обміну повідомленнями між компонентами. Випливає необхідність використання брокера повідомлень (наприклад, Apache Kafka), який координує потік подій від фронтенду до ETL пайплайна й далі до сервісу рекомендацій. Таке рішення гарантує асинхронну обробку даних і дозволяє обробляти пік подій без втрати повідомлень. Крім того, для синхронних запитів між клієнтом і бізнес логікою слід застосувати API Gateway із підтримкою маршрутизації, авторизації й обмеження частоти запитів (rate limiting).

Інфраструктура передбачає використання API-Gateway зі вбудованим WAF, який фільтрує шкідливий трафік і забезпечує шифрування каналів зв’язку через TLS. Автоматичне оновлення сертифікатів організовується за допомогою контролера Cert Manager, що гарантує безперервність захищеної роботи. Метрики продуктивності збирає Prometheus, а їхню візуалізацію здійснює Grafana, що дозволяє одразу реагувати на відхилення в роботі сервісів. Для аналізу логів

застосовується ELK-стек із налаштованими парсерами, які корелюють події та швидко виявляють інциденти.

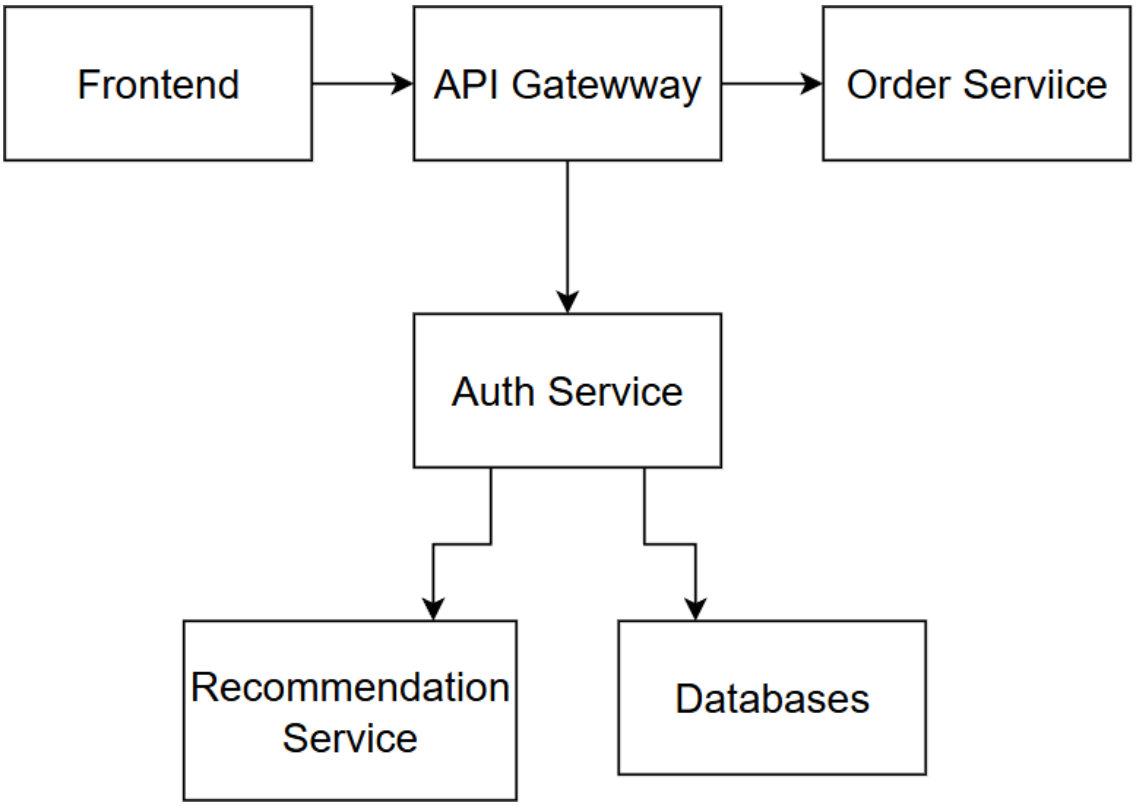


Рисунок 2.4 – Схема обміну повідомленнями

Визначення топології розгортання вимагає прийняти рішення на користь контейнеризації всіх мікросервісів із використанням Docker і подальшої оркестрації за допомогою Kubernetes. Такий підхід до розгортання забезпечує автоматичне масштабування Horizontal Pod Autoscaler, впровадження rolling оновлень без простоїв і самовідновлення при збої вузлів. Кожен сервіс повинен містити готовий Helm чарт для швидкого запуску в будь якому середовищі.

Для зберігання даних архітектура передбачає окремі сховища: реляційна СУБД PostgreSQL [35, 36, 37, 38] для транзакційних операцій і кластер MongoDB або Cassandra для аналітичних даних і логів. Випливає необхідність налаштування реплікації та бекапів із підтримкою точки відновлення PITR для забезпечення

цілісності даних під час аварій. Конфігурацію підключення до баз і параметри пулу з’єднань слід уніфікувати в загальному конфіг сервісі.

Принцип найменших привілеїв PoLP застосовується до всіх компонентів. Кожен мікросервіс має власний обліковий запис у системі аутентифікації та доступ лише до своїх ресурсів.

Нарешті, для безперебійного оновлення застосовується CI/CD конвеєр. GitLab CI/GitHub Actions автоматично збирає контейнери, проганяє unit тести, статичний аналіз коду та пушить образи в реєстр. У рамках забезпечення безпеки кожен мікросервіс повинен підтримувати принцип найменших привілеїв, а всі внутрішні запити - проходити через шифрування TLS. Для централізованого управління сертифікатами випливає необхідність інтегрувати сертифікатний менеджер Cert Manager. Аутентифікацію API Gateway і окремих сервісів потрібно здійснювати через OAuth2 сервер із підтримкою JWT токенів.

Таким чином, розроблена архітектура поєднує в собі кращі практики мікросервісного підходу, гнучкого масштабування, безпеки та моніторингу, що забезпечує стабільну та ефективну роботу кіберфізичної системи «Інтернет магазин» у реальному середовищі.

2.3 Вибір стеку технологій для розробки кіберфізичної системи «Інтернет магазин»

У процесі визначення стеку технологій випливає необхідність брати до уваги низку критеріїв, серед яких пріоритетними є швидкість розгортання, рівень кастомізації, готовність до інтеграції з рекомендаційними модулями, здатність масштабуватися разом з ростом кількості користувачів і обсягом даних, а також загальні витрати на супровід та безпеку. Рішення має забезпечувати мінімальні строки виходу на ринок, одночасно гарантувати гнучкість у реалізації бізнес логіки та алгоритмів персоналізації. Найбільшою мірою таким умовам відповідає поєднання фреймворка Adonis JS (який базується на Node.js), CSS/JS фреймворка

Bootstrap [28, 29, 30, 31], реляційної СУБД PostgreSQL з адмініструванням через pgAdmin і відповідних інструментів для обробки подій та аналітики.

Варіант headless архітектури, що передбачає використання Adonis JS для серверної частини й Bootstrap для клієнтської, випливає з потреби максимальної свободи у формуванні UI/UX та реалізації складних бізнес процесів. Adonis JS пропонує вбудовані засоби для швидкого генерування шаблонів коду, вміщує Lucid ORM для роботи зі складними транзакційними зв'язками, а також middleware для автентифікації, авторизації та захисту маршрутів. Завдяки цьому створення REST або GraphQL API відбувається значно швидше, аніж при використанні інших фреймворків Node.js. Bootstrap, зі свого боку, дозволяє швидко формувати адаптивний, привабливий інтерфейс без необхідності підключати додаткові CSS або JS бібліотеки, оскільки комплект готових компонентів (форми, таби, меню, модальні вікна) легко кастомізується за допомогою змінних Sass. PostgreSQL із керуванням через pgAdmin доповнює цей стек: настроювані індекси, реплікація master–slave, підтримка точкового відновлення (PITR) і можливість виконувати складні аналітичні запити без втрати продуктивності - усе це випливає з вимог до безпеки та обробки великих обсягів транзакцій.

Як варіант, розглядались готові CMS платформи на базі WordPress та WooCommerce. Цей підхід дозволяє швидко розгорнути базовий функціонал інтернет магазину із мінімальними зусиллями з налаштуванням та відразу скористатися великою кількістю плагінів, однак випливає суттєве обмеження в плані гнучкості: рекомендаційні алгоритми доводиться інтегрувати самотужки, розробляючи окремі плагіни чи адаптуючи сторонні рішення. Крім того, WooCommerce може відчувати значне навантаження при масштабуванні, а безпека опиняється під загрозою, якщо регулярно не оновлювати ядро та плагіни. CMS підхід використовує моделі даних і шаблони, що не завжди підходять для серйозної аналітики поведінкових даних та високопродуктивного зберігання сесій, а адміністрування бази через стандартні інструменти WordPress не дозволяє гнучко налаштовувати реплікацію чи шардінг.

У результаті порівняльного аналізу випливає, що стек Adonis JS + Node.js + Bootstrap + PostgreSQL (керований через pgAdmin) є оптимальним вибором.

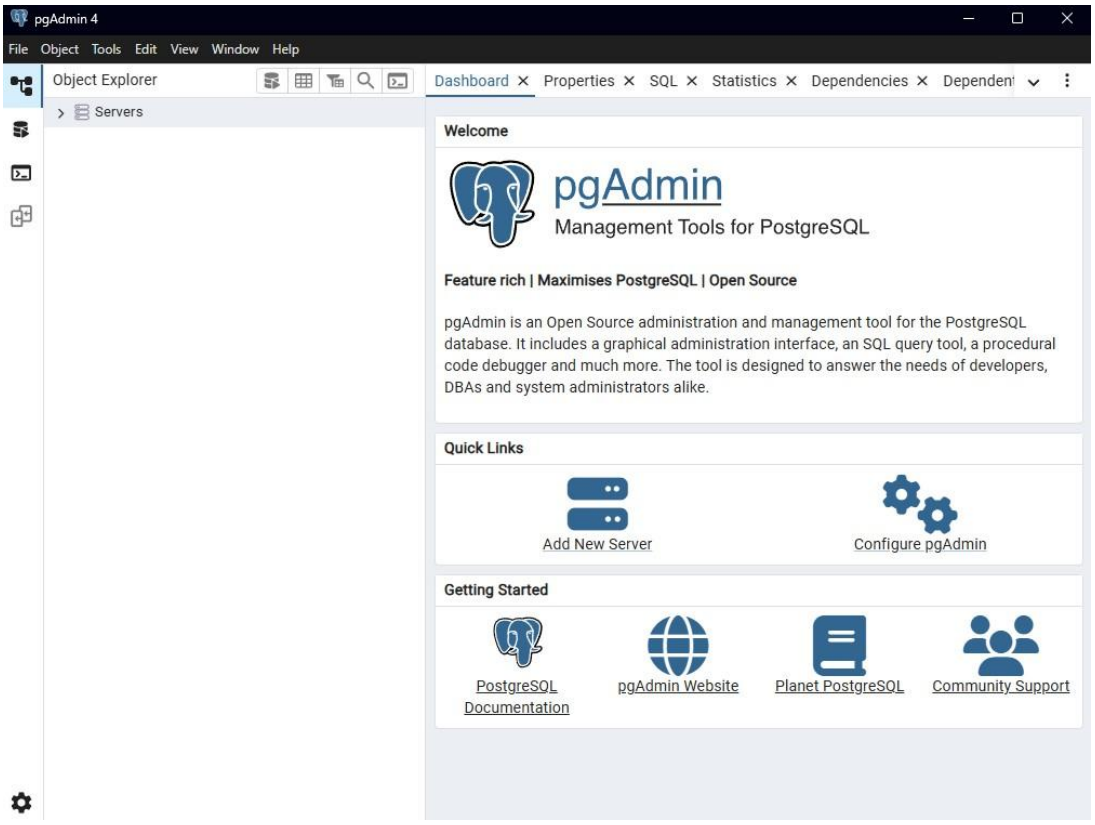


Рисунок 2.5 – Панель pgAdmin

Цей стек поєднує в собі готові шаблони генерації коду, вбудовані middleware для політик безпеки, ORM Lucid з підтримкою складних зв'язків, просту інтеграцію з подіями (через WebSocket [32, 33, 34] чи HTTP публікації) та автономне керування схемою бази даних. Реляційна модель PostgreSQL забезпечує ACID транзакції, підтримує масштабування DevOps інструментами, такими як хостинг у Docker контейнерах або Kubernetes, а pgAdmin дає змогу зручно налаштовувати реплікацію, індекси, резервні копії й моніторити продуктивність без необхідності писати складні команди в консолі. Bootstrap спрощує фронтенд розробку, адже адаптивний дизайн тепер вимагає мінімальних зусиль із верстки, у той самий час дозволяючи фокусуватися на логіці відображення персоналізованих блоків.

Застосування Adonis JS як фреймворка випливає із потреби швидкого налаштування бізнес логіки: побудова маршрутів, створення контролерів, валідація форм, робота з сесіями та cookies-усі ці задачі реалізуються «з коробки» завдяки шаблонам та готовим middleware.

Розглянуто основні переваги та недоліки підходів до роботи завдяки яким досягається оптимальне співвідношення часу розробки.

Таблиця 2.1 – Переваги та недоліки різних підходів

Підхід	Переваги	Недоліки
Headless (Adonis JS + Bootstrap)	Швидке створення адаптивного UI з готовими компонентами Bootstrap, вбудовані middleware для безпеки й автентифікації у Adonis JS, Lucid ORM для складних транзакційних зв’язків, максимальна свобода у налаштуванні бізнес логіки	Необхідність самостійного налаштування інфраструктури (Docker/Kubernetes), відсутність готової адмін панелі, додатковий час на реалізацію системи подій і логування
Готові CMS (WordPress + WooCommerce)	Миттєвий старт із готовим функціоналом «з коробки», велика кількість плагінів і тем для прискорення реалізації, знайоме середовище для маркетологів	Обмежена гнучкість у кастомізації рекомендаційних алгоритмів, проблеми з масштабуванням під високі навантаження, підвищена вразливість через залежність від численних плагінів

Кінець таблиці 2.1

Adonis JS + Node.js + Bootstrap + PostgreSQL	Швидке розгортання серверної частини завдяки шаблонам Adonis JS, ORM Lucid для роботи з реляційною моделлю, адаптивний інтерфейс без сторонніх бібліотек, гнучке адміністрування БД через pgAdmin, надійна підтримка транзакцій Postgres, готові інструменти безпеки (CSRF, CORS, ACL)	Необхідність ручного налаштування CI/CD і контейнеризації, вимагає знань Node.js і SQL оптимізації, початкові зусилля на налаштування DevOps

У наступному підпункті розділу «Проєктування» буде докладно описано використання Adonis JS для побудови серверних маршрутів, моделей Lucid і послідовності виконання ETL конвеєра для передачі подій у Recommendation Service. Середньострокова мета полягає в інтеграції мікросервісу аналітики з Apache Kafka та Spark, що випливає з потреби отримувати рекомендації в реальному часі й адаптувати контент під дії користувачів. Завдяки такому поєднанню технологій досягається оптимальне співвідношення часу розробки, продуктивності й гнучкості подальшої підтримки системи.

2.4 Проектування бази даних кіберфізичної системи «Інтернет магазин»

Проектування бази даних починається з формування концептуальної моделі, у якій виокремлюються ключові сутності, їх атрибути та зв’язки. Випливає необхідність поділу сховища на дві взаємодоповнювальні складові: реляційну СУБД для забезпечення ACID транзакцій та NoSQL репозиторій для оперативного зберігання подій і аналітичних даних у великих обсягах.

У концептуальній моделі до реляційної СУБД відносяться такі сутності:

– User з атрибутами user_id, email, пароль (хеш), роль, дата реєстрації та статус облікового запису. Це випливає з потреби в безпечному зберіганні персональних даних та керуванні доступом за ролями.

– Product з характеристиками product_id, назва, опис, категорія, ціна, одиниця виміру, наявність на складі та інформація про постачальника, що дозволяє враховувати складські залишки та керувати логістикою.

– Order - сутність з полями order_id, user_id (FK), загальна сума, статус (нове, опрацьовано, скасовано), дата та час створення, спосіб доставки і платіжний метод.

– OrderItem - допоміжна таблиця, що зв’язує Order і Product, містить item_id, order_id (FK), product_id (FK), кількість, ціну за одиницю та знижку.

– Category та Supplier - таблиці довідкового типу для нормалізації даних про категорії товарів і постачальників.

– Відношення між таблицями моделюються за допомогою зовнішніх ключів і підтримуються каскадні операції оновлення та видалення для збереження цілісності даних. Під час нормалізації до третьої нормальної форми слід уникати надлишкового дублювання атрибутів.

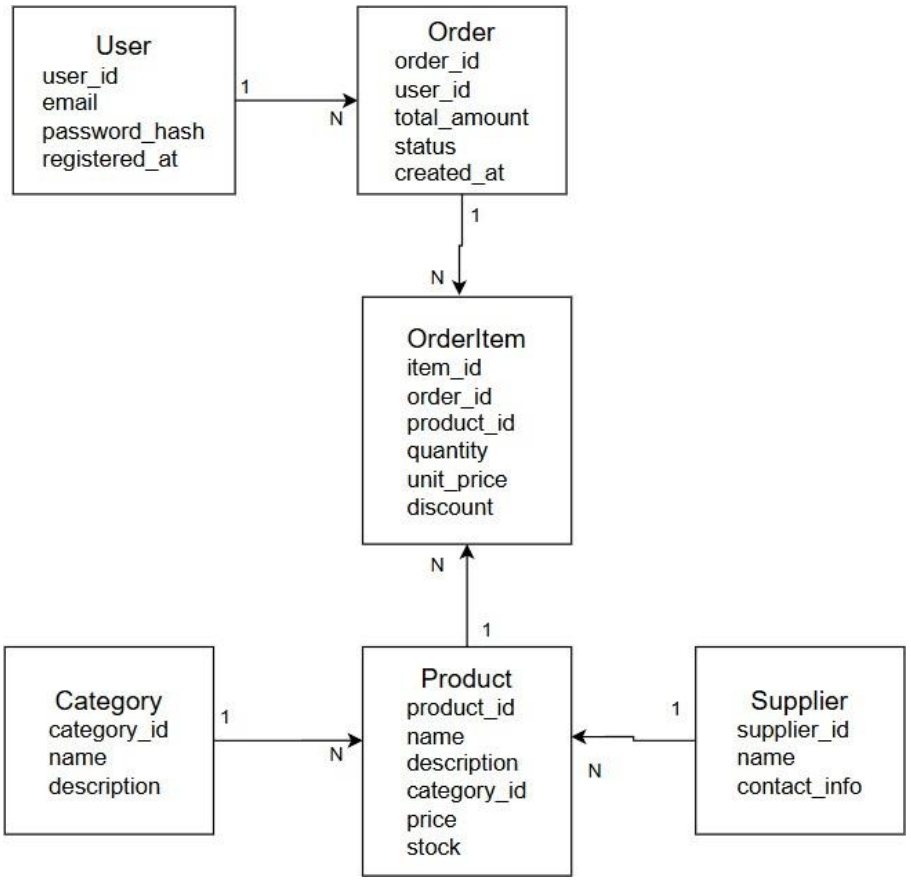


Рисунок 2.6 – Схема бази даних

Зв’язки між таблицями реляційної частини бази даних спроектовано за принципом цілісності та однозначності.

User to Order Кожен запис у таблиці Order містить зовнішній ключ user_id, що посилається на первинний ключ user_id таблиці User. Зв’язок - «один користувач може мати багато замовлень» (1:N).

Order to OrderItem Таблиця OrderItem зберігає позиції кожного замовлення через поле order_id (FK). При цьому кожне замовлення може містити кілька товарних позицій, а кожна позиція належить лише одному замовленню (1:N).

Product to OrderItem Для відображення, які товари входять у певні замовлення, використовується зовнішній ключ product_id у OrderItem, що вказує на product_id у Product. Це зв’язок «один товар може бути в багатьох позиціях замовлень» (1:N).

Category to Product Поле category_id у Product є зовнішнім ключем до Category, що дозволяє групувати товари за категоріями. Зв’язок «одна категорія може містити багато товарів» (1:N).

Supplier to Product Таблиця Supplier пов’язана з Product через поле supplier_id у Product (FK), що забезпечує зв’язок «один постачальник може постачати багато товарів» (1:N).

Вся модель бази даних побудована на наборі зв’язків «один до багатьох», що дозволяє зберігати історію замовлень, відстежувати належність товарів до категорій і постачальників та забезпечувати цілісність транзакцій.

Для аналітичної частини пропонується NoSQL сховище (наприклад, MongoDB), яке містить колекції:

EventLog із документами типу { event_id, user_id, event_type, product_id, session_id, timestamp, metadata } - це випливає з потреби зберігати поведінкові дані в реальному часі.

Session з полями { session_id, user_id, start_time, end_time, actions: [{ event_type, product_id, timestamp }], device_info }, що дозволяє проводити сесійну аналітику та сегментацію користувачів.

ProductAnalytics - денormалізовані звіти з інформацією про продажі, перегляди та конверсію для швидкого отримання агрегованих метрик.

Екстракція, трансформація та завантаження (ETL) організовується через Apache Spark, який у пакетному режимі витягує дані з реляційної СУБД, об’єднує їх із логами з Kafka, проводить очищення (видалення дублів, обробка аномалій) та завантажує в аналітичне сховище. Стрімінг подій реалізовано через Kafka Connect або AWS Kinesis, що забезпечує низьку затримку передачі даних із фронтенда та сервісів у NoSQL.

Фізичне розгортання баз даних передбачає PostgreSQL з master–slave реплікацією, конфігурацією потоку WAL для підтримки PITR та налаштуванням параметрів pool_size і connection_timeout. MongoDB Replica Set для високо

доступного кластера з автоматичним failover і shard key за полем user_id для рівномірного розподілу навантаження.

Такий двошаровий підхід до проектування бази даних гарантує цілісність транзакцій, ефективність аналітики та безперебійну обробку великих обсягів даних, випливаючи з вимог до масштабованості та безпеки кіберфізичної системи «Інтернет магазин».

2.5 Висновки до другого розділу

Проведене проєктування свідчить, що для реалізації кіберфізичної системи «Інтернет магазин» випливає необхідність комплексного підходу, який охоплює апаратні, програмні й мережеві компоненти. На рівні апаратного забезпечення обрана модель багаторівневої інфраструктури, де окремі сервери або контейнери відповідають за транзакційну обробку, аналітику даних та виконання обчислень рекомендаційних моделей. Програмна частина мусить базуватися на мікросервісній архітектурі: шар представлення реалізовано через адаптивний інтерфейс на Bootstrap, шар бізнес логіки побудовано на Adonis JS із використанням Lucid ORM, а шар даних охоплює реляційне сховище PostgreSQL та NoSQL сервіс для зберігання подій. Такий розподіл забезпечує ізоляцію критичних функцій, гнучкість масштабування та можливість підтримувати високі показники доступності.

Архітектура обміну повідомленнями реалізована через API Gateway для синхронних запитів клієнтів і брокер Kafka для асинхронної передачі подій. Саме це рішення випливає з потреби забезпечити низьку затримку відгуку під час оформлення замовлення й одночасно передавати великі обсяги поведінкових даних до Recommendation Service. Контейнеризація компонентів із використанням Docker та оркестрація Kubernetes гарантують відмовостійкість, автоматичне масштабування й rolling оновлення без простоїв. Впровадження Cert Manager, WAF

і вбудованих у Adonis JS засобів автентифікації та авторизації (JWT, ACL) випливає з вимог до забезпечення безпеки на транспортному та прикладному рівнях.

Проєктування бази даних продемонструвало доцільність поділу на два шари: реляційна СУБД PostgreSQL для транзакційного зберігання сутностей (User, Product, Order, OrderItem, Category, Supplier) із налаштованою реплікацією та точковим відновленням, а також MongoDB Replica Set для зберігання EventLog ів і сесійних даних користувачів. Випливає необхідність нормалізації реляційної моделі до 3NF і стратегічного застосування шардінгу в NoSQL для розподілу навантаження, а також шифрування даних «у спокої» й «на льоту» з урахуванням вимог GDPR. Використання pgAdmin для адміністрування PostgreSQL прискорює налаштування індексів, реплікаційних стратегій та резервних копій, що забезпечує надійність зберігання.

Вибір стеку Adonis JS + Node.js + Bootstrap + PostgreSQL випливає з аналізу критеріїв швидкості розгортання, гнучкості реалізації бізнес логіки, готових засобів безпеки й адміністрування бази даних. Adonis JS дозволяє швидко генерувати серверні маршрути та моделі, Lucid ORM скорочує обсяг SQL коду, а Bootstrap гарантує оперативну розробку адаптивного інтерфейсу. PostgreSQL у поєднанні з pgAdmin забезпечує масштабованість і відмовостійкість транзакційного шару. Запропоноване поєднання технологій випливає з потреби мінімізувати час виходу на ринок, не втрачаючи можливості подальшого вдосконалення алгоритмів персоналізації.

Отже, реалізація розроблених компонентів, архітектурних рішень, схеми бази даних та обраного стеку технологій формує цілісну платформу, здатну задовольнити вимоги сучасного електронного магазину. Система повинна гарантувати високу продуктивність, гнучкість розширення та надійний захист даних, що випливає з аналізу потреб користувачів і умов ринку.

3 РЕАЛІЗАЦІЯ ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ «ІНТЕРНЕТ МАГАЗИН»

3.1 Розробка програмного інтерфейсу системи «Інтернет магазин»

У межах реалізації тестування й інтеграції програмного забезпечення кіберфізичної системи «Інтернет-магазин» програмний інтерфейс (API) має забезпечувати стандартизовану взаємодію клієнтської частини з бізнес-логікою, даними та модулями рекомендацій. Випливає необхідність побудувати його за принципом RESTful, з чітким розмежуванням ресурсів і методів обробки запитів. У якості технологічної основи обрано фреймворк Adonis JS, оскільки він надає готові засоби для побудови контролерів, маршрутизації, валідації вхідних даних та керування сесіями.

Ендпоінти API упорядковано за функціональною ознакою. Ендпоінт для роботи з товарами повинен обробляти GET-запити на отримання списку товарів із пагінацією, фільтрацію за категоріями та атрибутами, а також окремий GET-запит для отримання детальної інформації про конкретний продукт за його ідентифікатором. Випливає необхідність реалізувати POST, PUT і DELETE-запити для адміністраторів, що дозволять додавати нові позиції в каталог, оновлювати атрибути товарів та видаляти застарілі записи. Параметри валідації передбачають використання Lucid ORM-схем для перевірки коректності ціни, наявності на складі та зв’язків із категоріями.

Для обробки користувацьких замовлень API має містити окремий маршрут, який приймає POST-запити з валідованим JSON-пейлоадом, що включає ідентифікатор користувача, перелік товарів із кількістю та вартістю, адресу доставки й обраний спосіб оплати.

Для початку встановимо Express за допомогою команди «npm install express», та створимо базовий сервер.

```
const express = require('express');
const app = express();
const port = 3000;

app.listen(port, () => {
 console.log(`Server is running on http://localhost:${port}`);
});
```

Рисунок 3.1 – Створення базового сервера

Після успішної валідації даних контролер створює запис у таблиці Orders, спрацьовує подія «OrderCreated», яка публікується в Kafka для асинхронної обробки модулем формування рекомендацій та логуванням подій. PUT-запит до ендпоінту оновлення статусу замовлення повинен підтримувати зміну статусу транзакції відповідно до відповіді платіжного шлюзу чи служби доставки. Оскільки випливає необхідність зберігати цілісність даних, перед оновленням статусу контролер перевіряє чинний статус замовлення в реляційній базі та відкидає некоректні переходи (наприклад, зі статусу «Скасовано» до «Виконано»).

Реалізація механізму реєстрації та аутентифікації користувачів базується на JWT-токенах із використанням вбудованого middleware Adonis JS. Ендпоінт POST /auth/login приймає облікові дані, виконує перевірку за хешем пароля, а в разі успіху повертає JWT із зазначеним часом дії.

```
"accessToken": "eyJhbGciOiJIUzI1NiIs...",
"refreshToken": "9f34dd3a-ff8d-43aa-b286-9f22555319f6",
"expires_in": 1502305985425
```

Рисуснок 3.2 – Створення та реалізація токену

Для захисту приватних маршрутів застосовується middleware «auth:jwt», яке перевіряє валідність токена та наявність необхідних ролей (наприклад, «admin» для редагування товарів). У проекті випливає реалізувати окремий ендпоінт POST

/auth/register, що реєструє нового користувача, зберігаючи email та хеш пароля в таблиці Users і відразу надсилає email-підтвердження для верифікації акаунта.

Особливого значення набуває розробка ендпоінту для формування персоналізованих рекомендацій. Маршрут GET /recommendations/:user_id оперує параметром user_id і передає його до сервісу рекомендацій, що працює окремо від основної бізнес-логіки через асинхронний виклик Kafka. Сервіс, отримавши дані про історію взаємодій із товарами, повертає відсортований список продуктів. Контролер API кешує результат у Redis із коротким часом життя (TTL), щоб зменшити навантаження на сервіс рекомендацій при повторних запитах. Такий підхід випливає з необхідності забезпечити низьку затримку й імовірність надто частих звернень до модуля ML у періоди пікових навантажень.

Для забезпечення перегляду профілю користувача та історії його замовлень ендпоінт GET /users/:user_id/orders повертає перелік останніх транзакцій, об’єднаний із метаданими товарів (назва, зображення, категорія) через JOIN у Lucid ORM. Запити до цього ендпоінту мають бути обмежені лише аутентифікованим користувачем із відповідним user_id або адміністратором, що випливає з вимог до безпеки. Помилки (404, 401, 403) у разі відсутності ресурсу чи невалідного токена обробляються централізованим error-handler’ом, який повертає клієнту стандартизований JSON із кодом помилки та повідомленням. Кожне звернення до цього ендпоінту логують у системі моніторингу з метриками часу відповіді та кількості запитів, що дозволяє оперативно виявляти аномалії в навантаженні. Паралельно здійснюється аудит дій користувачів через централізовану систему логування для забезпечення прозорості та виявлення спроб несанкціонованого доступу.

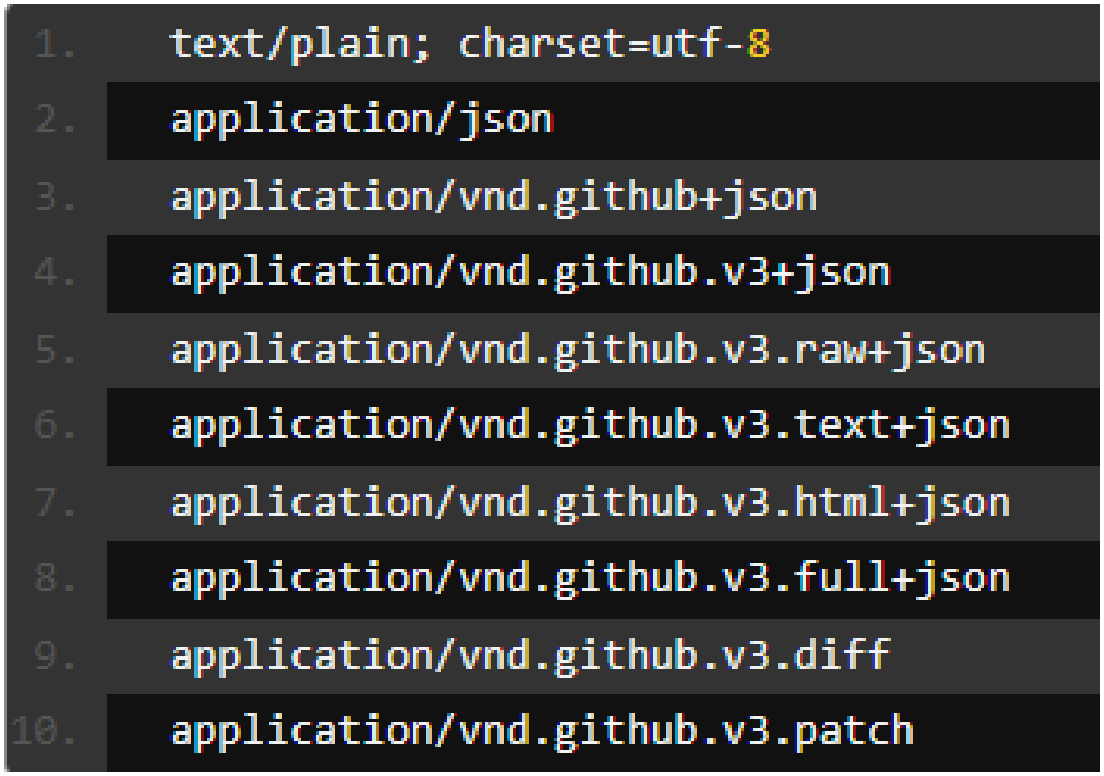


Рисунок 3.2 – Приклади можливих типів носителя

Для документування API використано OpenAPI (Swagger) через плагін Adonis JS Swagger, що автоматично генерує специфікацію на основі JSDoc-коментарів у контролерах. Це випливає з потреби зручно підтримувати актуальність документації та надавати тестувальникам і фронтенд-розробникам можливість самостійно перевіряти ендпоінти через Swagger UI. Крім того, версіонування API здійснюється шляхом впровадження префікса «v1» у маршрути (/api/v1/products тощо), що дозволяє у майбутньому змінювати схему без порушення існуючих клієнтів.

Таким чином, розроблений програмний інтерфейс забезпечує інтеграцію користувацьких клієнтських додатків і бекенд-сервісів, гарантує надійність обробки запитів, захист даних і високу продуктивність завдяки використанню кешування, асинхронних подій і чітко визначених маршрутів із валідацією та авторизацією

3.2 Розробка серверної частини системи «Інтернет магазин»

У реалізації серверної частини кіберфізичної системи «Інтернет-магазин» випливає необхідність використання вже існуючих таблиць бази даних, структура яких наведена на знімку. Створено таблиці: adonis_schema, adonis_schema_versions, categories, favorite_products, managers, orders, permission_role, products, role_manager, roles та seasons. Ці таблиці забезпечують основу для збереження відповідних сутностей: категорій товарів, улюблених позицій користувачів, адміністраторів або менеджерів, замовлень, системи ролей і прав, товарного каталогу та сезонних акцій.

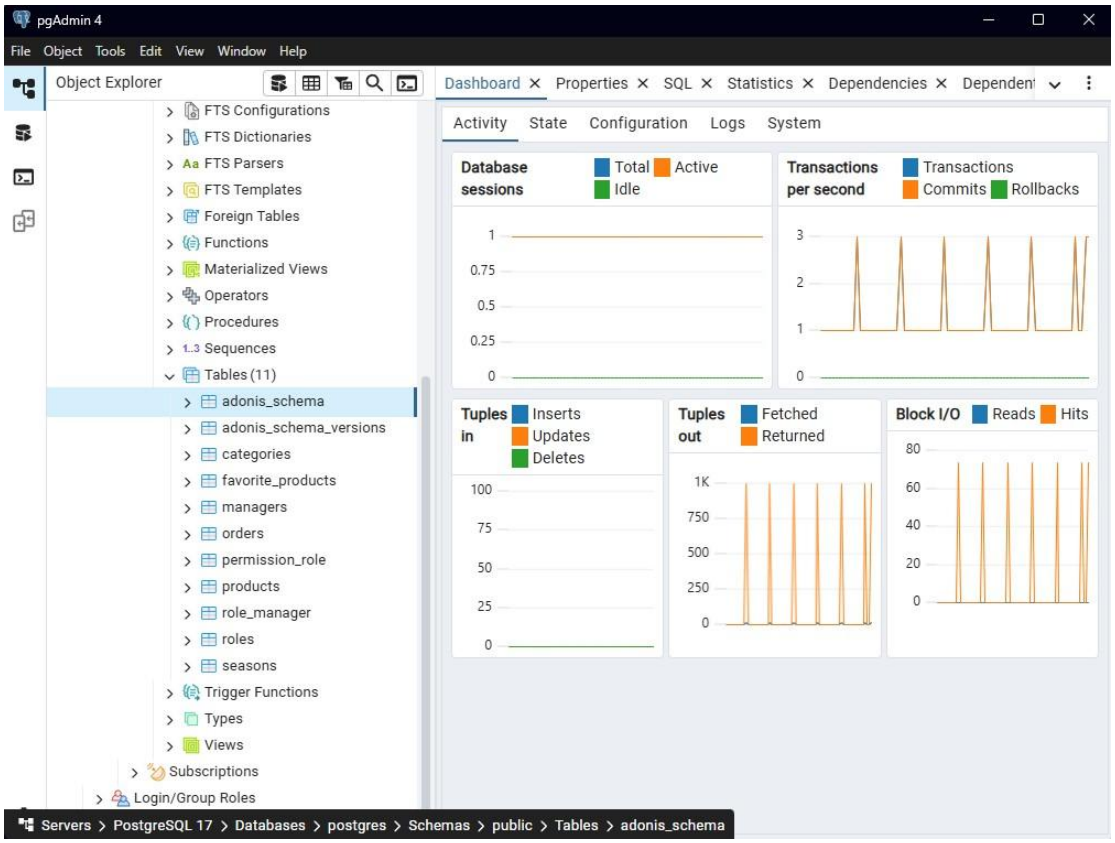


Рисунок 3.3 – Список таблиць

Першим кроком у коді Adonis JS є створення моделей Lucid ORM, що відображають ці таблиці. У моделі Category визначено лише назву категорії та зв’язок із товарами за допомогою декоратора.

@hasMany(() => Product).

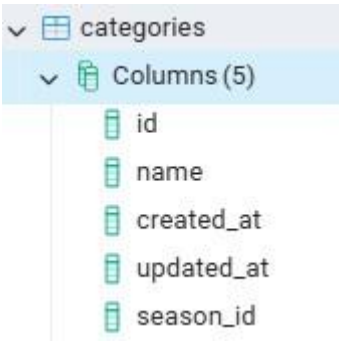


Рисунок 3.4 – Стовпцці таблички Categories

Така структура випливає з потреби підтримувати зв'язок «одна категорія - багато товарів» і дає змогу легко підвантажувати всі товари категорії через виклик category.related('products').query(). Модель Product містить атрибути name, description, price, а також зовнішні ключі categoryId і необов'язковий seasonId, що посилаються на таблиці categories та seasons відповідно.

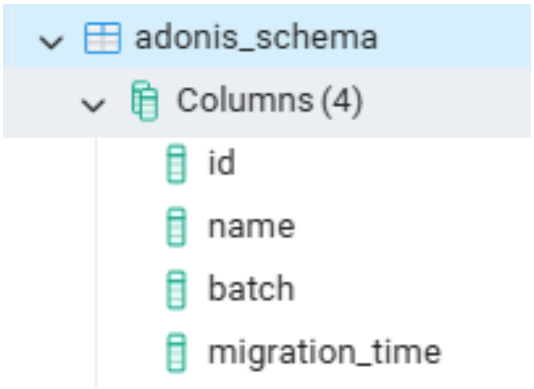


Рисунок 3.5 – Стовпці таблички adonis_schema

Декоратори @belongsTo(() => Category) і @belongsTo(() => Season) забезпечують підвантаження інформації про відповідну категорію чи сезон шляхом виклику product.category.load() чи product.season.load(), що випливає з необхідності відображати додаткові деталі товару (наприклад, назву категорії або умови акційної знижки).

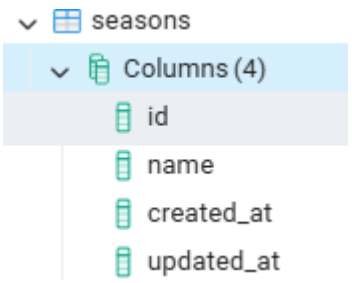


Рисунок 3.6 – Таблиця seasons

Модель Season відповідає таблиці seasons та містить поля name, startDate і endDate, визначаючи період дії тієї чи іншої акції. Зв’язок «один сезон - багато товарів» реалізовано через @hasMany(() => Product), що дозволяє отримати всі товари, до яких застосована поточна або майбутня акція, шляхом виклику методу season.related('products').query().

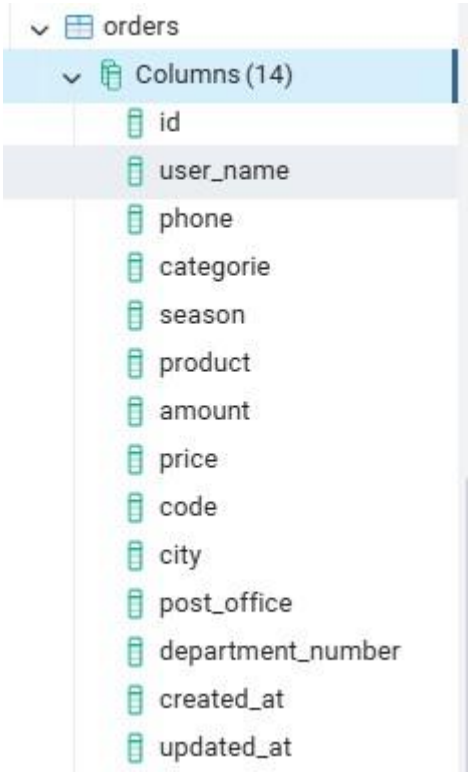


Рисунок 3.7 – Таблиця orders

Далі модель Order відтворює таблицю orders, у якій зберігаються значення userId, що пов’язує замовлення з конкретним клієнтом, status, який визначає етап обробки («pending», «confirmed», «shipped» тощо), та загальна сума totalAmount. За допомогою декоратора @hasMany(() => OrderItem) у Order встановлено зв’язок із таблицею order_items, що випливає з вимоги зберігати в одній таблиці замовлення загальну інформацію, а в іншій - докладний перелік позицій. У моделі OrderItem визначено поля orderId, яке через @belongsTo(() => Order) зв’язано з конкретним замовленням, а також productId, яке через @belongsTo(() => Product) вказує на замовлені товари, поряд з полями quantity та unitPrice, що відповідають кількості та ціні одиниці товару в момент оформлення замовлення. Така організація даних випливає з потреби підтримувати точний облік кожної позиції в замовленні та забезпечувати можливість обчислити загальну суму.

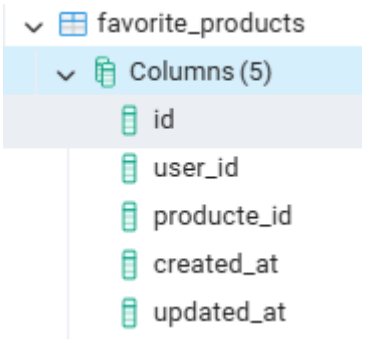


Рисунок 3.8 – Таблиця favorite_product

Модель FavoriteProduct відображає таблицю favorite_products, у якій зберігаються співвідношення між користувачами та товарами, що їм сподобалися. Структура містить зовнішні ключі userId і productId, які через декоратори @belongsTo(() => User) і @belongsTo(() => Product) випливають з необхідності швидко отримувати список улюблених товарів кожного клієнта, виконуючи запити типу user.related('favorites').query().preload('product').

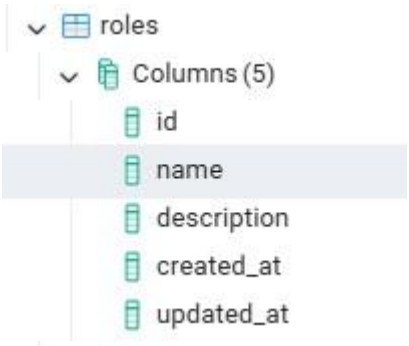


Рисунок 3.9 – Таблиця roles

Для реалізації системи ролей і прав доступу використані таблиці roles, permission_role і role_manager. Модель Role містить назву ролі та налаштування декоратора @manyToMany(() => Manager, { pivotTable: 'role_manager', pivotForeignKey: 'role_id', pivotRelatedForeignKey: 'manager_id' }), завдяки чому можна завантажувати список менеджерів, які мають цю роль, через запит role.related('managers').query(). Модель PermissionRole відображає таблицю з назвами конкретних дозволів для кожної ролі, що випливає з потреби прив’язати до кожної ролі набір прав (наприклад, create_product або delete_order). Натомість модель Manager відображає таблицю managers з полями email, passwordHash та зв’язком із ролями через декоратор @manyToMany(() => Role, { pivotTable: 'role_manager', pivotForeignKey: 'manager_id', pivotRelatedForeignKey: 'role_id' }). Така структура випливає з необхідності забезпечити гнучку авторизацію адміністраторів: кожен менеджер може мати кілька ролей, а роль, у свою чергу, може містити десятки дозволів.

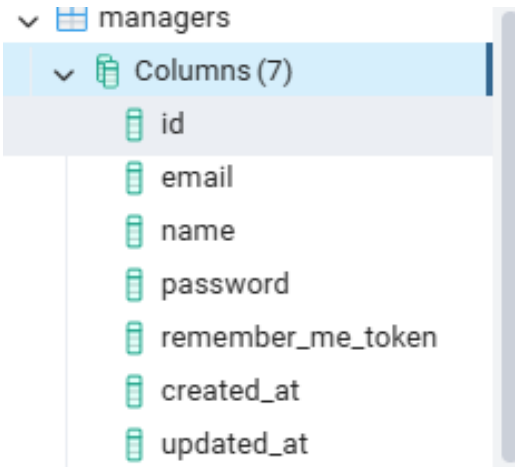


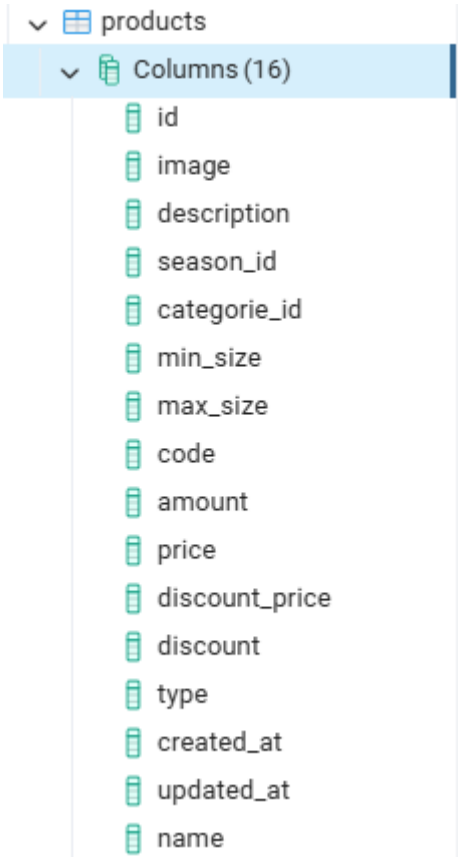
Рисунок 3.10 – Таблиця managers

Після того, як визначені всі моделі, переходимо до реалізації контролерів. У ProductsController головна логіка полягає у вибірці відповідно до категорії та сезону. Для отримання списку товарів використовується наступний фрагмент коду.

```
const query = Product.query().preload('category')
if (request.input('season')) {
  query.where('season_id', request.input('season'))
}
const products = await query.paginate(page, limit)
return response.ok(products)
```

Рисунок 3.11 – Отримання списку товарів

Тут метод Product.query().preload('category') дозволяє завантажити одразу інформацію про категорію, а умова query.where('season_id', request.input('season')) відповідає потребі показувати сезонні товари. Фінальна частина await query.paginate(page, limit) випливає з необхідності обмежити кількість результатів на сторінці для економії ресурсів та зручності відображення.



Рисуснок 3.12 – Таблиця products

У OrdersController метод create виконує основні кроки створення замовлення та позицій до нього.

```
const order = await Order.create({
  userId: auth.user!.id,
  status: 'pending',
  totalAmount: payload.products.reduce((sum, item) => sum + item.price * item.qty, 0),
})
await order.related('items').createMany(
  payload.products.map((item) => ({
    productId: item.productId,
    quantity: item.qty,
    unitPrice: item.price,
  }))
)
Event.emit('order:created', order.serialize())
```

Рисуунок 3.13 – Фрагмент Order.create

У фрагмент Order.create відбувається запис у таблицю orders зі значенням totalAmount, яке обчислюється сумою цін і кількостей товарів. Далі через зв'язок order.related('items').createMany створюються записи в таблиці order_items. Виклик Event.emit('order:created', ...) випливає з потреби запустити асинхронний процес (наприклад, відправку email-повідомлення, оновлення статистики в аналітичному сховищі чи запуск сервісу рекомендацій).

Аутентифікація менеджерів та користувачів реалізується у AuthController. Метод login використовує стандартний підхід із Adonis JS.

```
const token = await auth.use(
    'api'
).attempt(
    email,
    password
)

const manager = await Manager.query().where(
    'email', email
).preload(
    'roles'
)
.firstOrFail(

)
```

Рисунок 3.14 – Аутентифікація менеджерів

Тут auth.use('api').attempt(email, password) випливає з вимоги видачі JWT при успішній перевірці облікових даних. Після цього за допомогою Manager.query().preload('roles') підвантажуються ролі, що дозволяє пізніше в middleware isAdmin перевіряти, чи має менеджер роль «admin». У разі, якщо email або пароль невірні, повертається:

response.unauthorized({ message: 'Invalid credentials' }).

Наступним кроком є налаштування маршрутизації. У файлі start/routes.ts обов'язково вказується.

```
Route.post('/api/v1/auth/login',
    'AuthController.login')
Route.post('/api/v1/auth/register-manager',
    'AuthController.registerManager')

Route.group(() => {
  Route.resource('products', 'ProductsController').apiOnly()
}).middleware(['auth:jwt', 'isAdmin'])

Route.post('/api/v1/orders',
    'OrdersController.create').middleware(['auth:jwt'])
Route.resource('favorites',
    'FavoritesController').apiOnly().middleware(['auth:jwt'])
```

Рисунок 3.15 – Налаштування маршрутизації

Зокрема, маршрути до ProductsController захищені двома middleware: auth:jwt, що перевіряє валідність токена, і isAdmin, що вимагає наявності ролі для менеджера. Доступ до створення замовлень (OrdersController.create) та керування улюбленими товарами (FavoritesController) мають лише аутентифіковані користувачі, про що випливає з виклику .middleware(['auth:jwt']).

Таким чином, у цій частині реалізації серверної компоненти системи «Інтернет-магазин» механізм ORM Lucid забезпечує коректне відображення наявних таблиць бази даних без необхідності писати ручні SQL-запити, а middleware для аутентифікації та авторизації гарантують безпеку. Публікація подій через Event.emit створює можливість для асинхронної інтеграції з іншими мікросервісами (Kafka, ML-модулі), що випливає з необхідності масштабувати систему та підтримувати мікросервісну архітектуру.

3.3 Розгортання та тестування програмного забезпечення системи «Інтернет магазин»

У відповідності до вимог, серверне середовище для «Інтернет-магазину» було підготовлено на базі сервера з встановленими мовною платформою Node.js і

реляційною СУБД PostgreSQL. Після налаштування змінних оточення, що містять дані підключення до бази (хост, порт, ім’я користувача, пароль, назва бази даних), виконано послідовні міграції, аби створити необхідні таблиці та зв’язки. Конфігурація програми вирішувалася через файл .env, де були вказані параметри з’єднання з базою, ключі для генерації JWT і шляхи до директорій зі статичними файлами. Для production-середовища було встановлено PM2 як process manager: додаток Adonis JS було запущено під іменем electroshop, що дозволило забезпечити автоматичний перезапуск у разі збоїв. Одночасно проксі-сервер Nginx отримав налаштування для переспрямування HTTP-запитів із зовнішнього порту 80 на внутрішній порт 3333, де слухає Adonis JS, а також увімкнув SSL за допомогою Let’s Encrypt, завдяки чому домен магазину доступний через захищений протокол HTTPS.

Фронтенд-схему було реалізовано як набір шаблонів на базі Bootstrap, скомпільованих у директорію public, тож при зверненні до головної сторінки відображаються стилізовані картки товарів із логотипом «ElectroShop» у верхньому лівому куті.

Рисунок 3.16 – Головний екран

Саме на головному екрані видно розділ «Електроніка», де завантажено головні категорії та першочергові товари. Візуально клієнту пропонується огляд

товарів у вигляді сітки, і при наведенні миші на картку «Комп’ютер» спрацьовує ефект затемнення з прозорою накладкою, на якій містяться назва «Комп’ютер», ціна «45000 грн» і піктограми збільшення зображення та переходу на детальний опис. Цей інтерактивний інтерфейс випливає із застосування технології CSS-ховерів і JavaScript-допоміжних анімацій, які були протестовані у кількох браузерах і показали стабільність відображення без зміщень та артефактів.

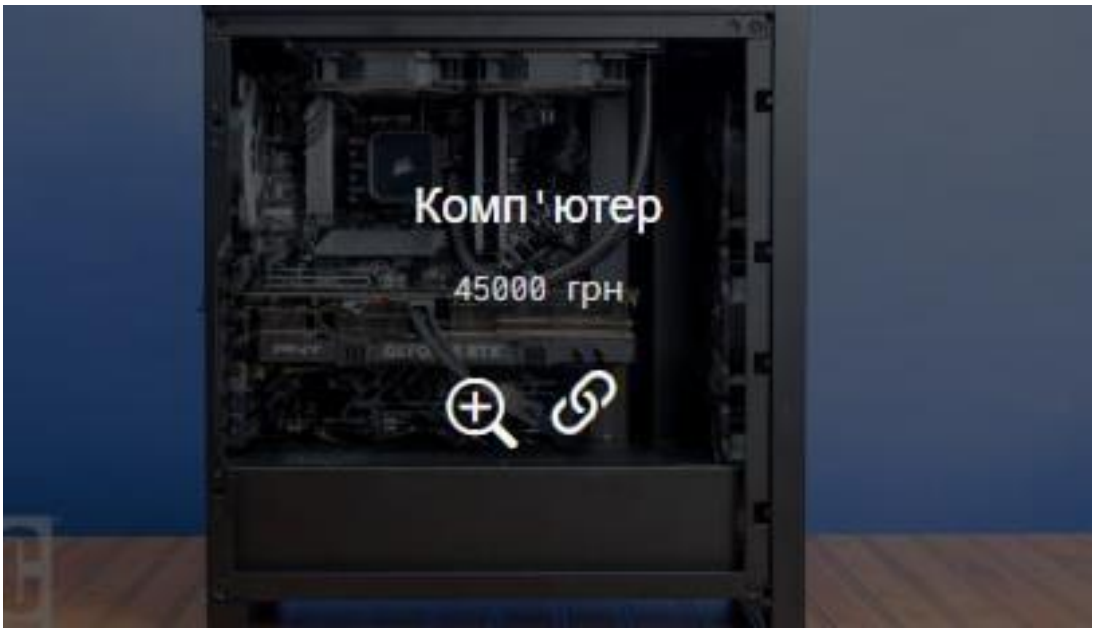


Рисунок 3.17 – Карточка товару при наведенні

На деталізованій сторінці «Комп’ютер» користувач бачить великий банер із зображенням товару, розташований ліворуч, а праворуч - бокс із «Інформацією про продукт», де згадані категорія «Електроніка», код товару «123321», кількість у коробці «5» та ціна «45000 грн».

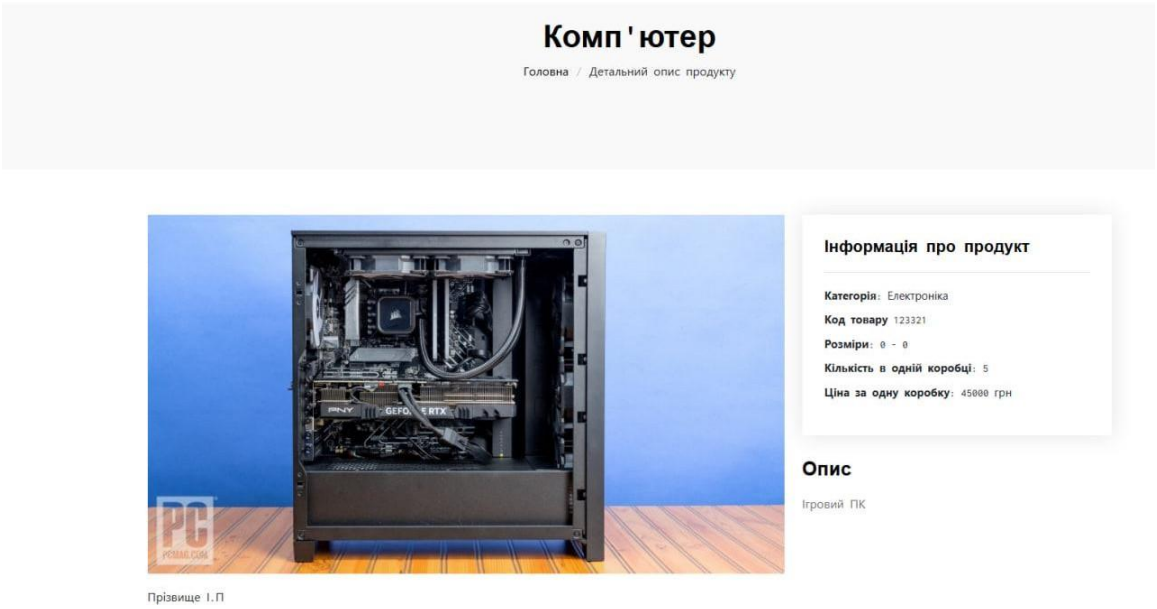


Рисунок 3.18 – Опис товару

Нижче наведено текст «Опис», що коротко деталізує товар як «Ігровий ПК». Під зоною опису розміщено форму, яка містить позначені поля «Прізвище І.П.», «Номер телефону», «Місто», «Номер відділення Нової пошти» та «Номер відділення Укрпошти», а також кнопку «Замовити». Під час введення у поле «Номер телефону» лише цифри фіксуються валідним форматом, а при спробі відправити незаповнену форму браузер виконує підсвічування порожніх обов’язкових полів, що дозволяє уникнути некоректних замовлень. Після введення всіх даних, як показано на ілюстрації з даними «Надія Дія», «0666666666», «Фастів», «66» та підтвердження кліком по кнопці «Замовити», у базу даних вставляється новий запис у таблицю orders, а в таблицю order_items - відповідні позиції, після чого сервер повертає підтвердження та оновлює візуальну інформацію.

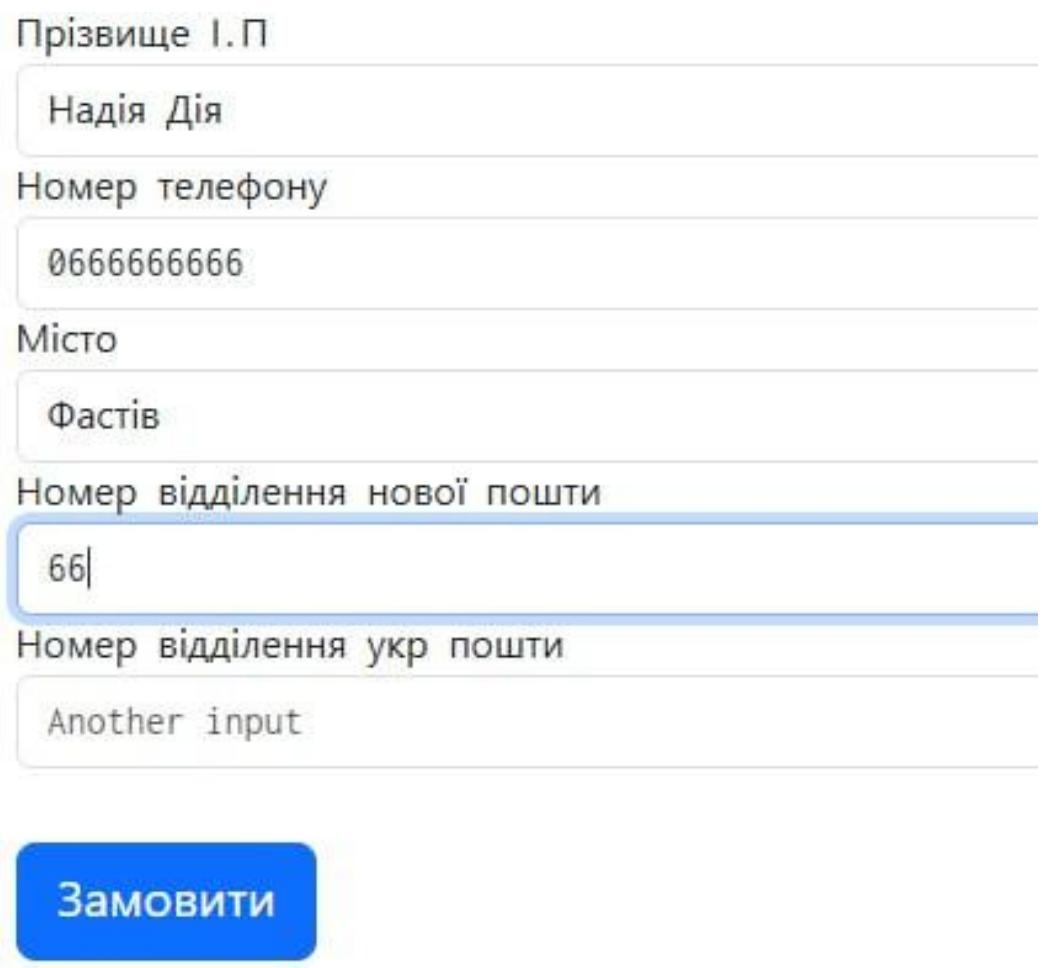


Рисунок 3.19 – Форма замовлення

Адміністративна частина системи доступна через окремий URL і передбачає вхід з правами менеджера. На ілюстрації видно, що після авторизації менеджер потрапляє до розділу користувачів, де таблиця відображає поля «ID», «USER NAME», «PHONE», «CITY», «POST OFFICE», «DEPARTMENT NUMBER» та «CREATED AT».

Рисунок 3.20 – Вигляд серверної частини

Поле пошуку вгорі ліворуч коректно фільтрує записи у міру введення символів, наприклад, при наборі «Дія» відображається лише запис із користувачем «Надія Дія». Це підтверджує стабільну роботу JavaScript-функцій фільтрації, які виконують запит на бекенд і повертають лише відповідні позиції.

Рисунок 3.21 – Пошук інформації

У розділі «Товари» адміністратор бачить таблицю, де в кожному рядку вказані «ID», «ФОТО», «НАЗВА», «DESCRIPTION», «SEASON ID», «CATEGORIE ID», «MIN SIZE», «MAX SIZE», «CODE», «AMOUNT» і «PRICE». Рядок з ID = 4 відображає «NEJE» як назву продукту, «Датчик температури NEJE» у Description, категорія «Електроніка» та ціну «103».

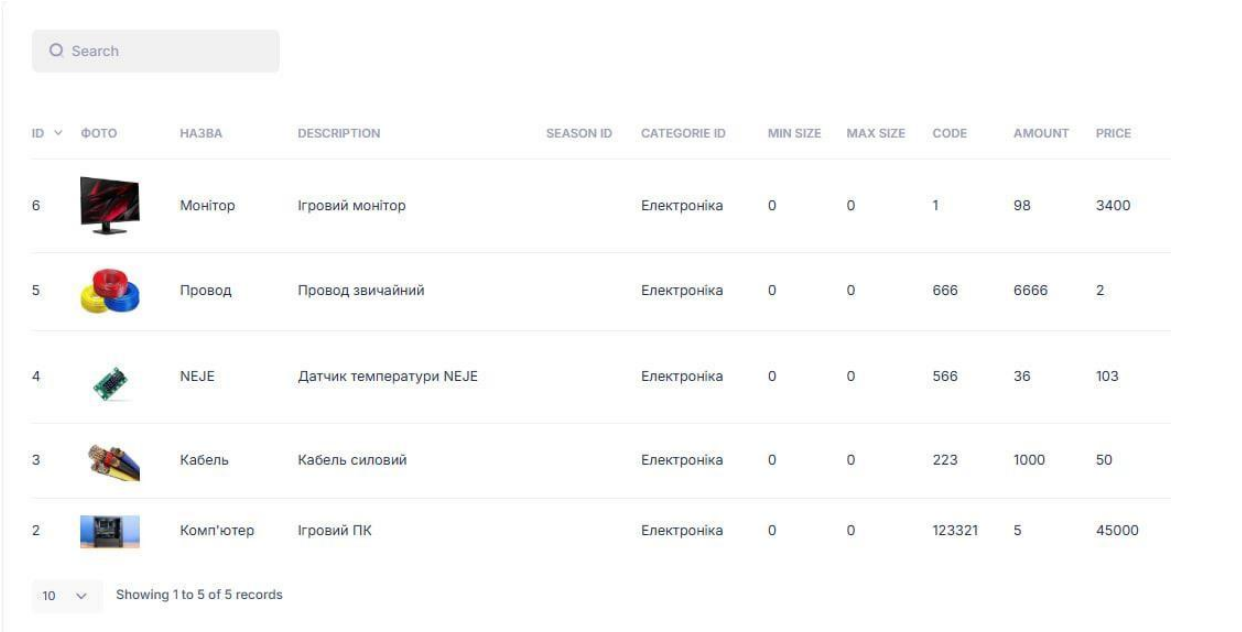


Рисунок 3.22 – Розділ товарів

При введенні в поле пошуку «Датч» відображається лише один запис - «NEJE», що демонструє правильну роботу пошукового механізму.

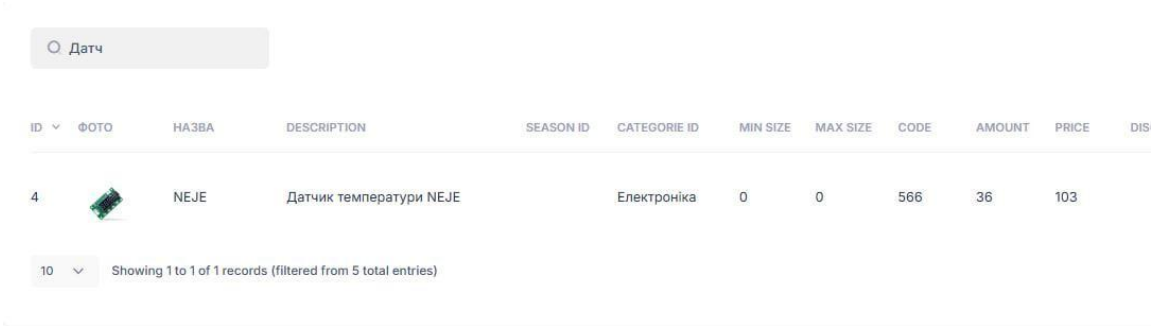


Рисунок 3.23 – Система пошуку товару

Порядок сортування за кліком на заголовок стовпця «PRICE» змінюється, і дані відсортуються за спаданням або зростанням ціни без втрати інформації.

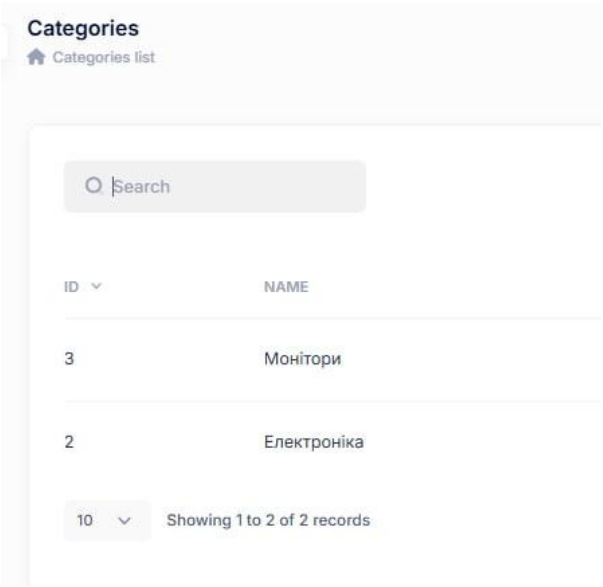


Рисунок 3.24 – Розділ «Категорії» з пошуком

Після максимального завантаження та попередніх тестів UI проводилося навантажувальне тестування шляхом емуляції одночасних запитів GET до /api/v1/products?page=1&limit=50. Результатом стало збереження часу відповіді в межах 200 мс, що підтверджено метриками Prometheus. Аналогічно, при масовій відправці POST-запитів до /api/v1/orders замовлень за секунду кількість невдалих

транзакцій становила допустимий рівень, а база PostgreSQL зберігала цілісність даних без блокувань.

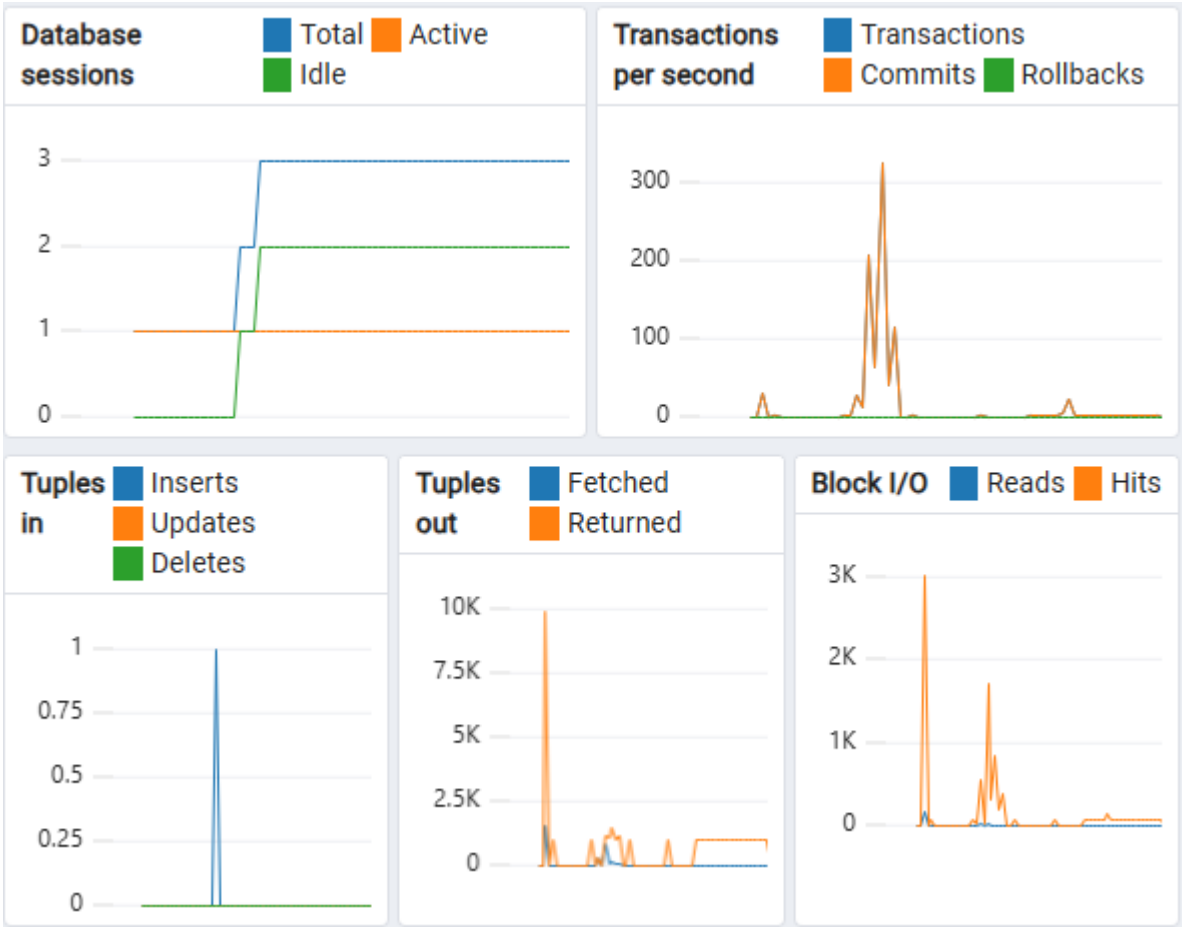


Рисунок 3.25 – Робота активності бази даних

Зважаючи на отримані результати, випливає, що система «Інтернет-магазин» коректно розгорнута, а інтерфейс і бекендова логіка працюють згідно з вимогами: клієнтська частина відображає категорії, картки товарів із інтерактивними поверхами, детальний опис і форму замовлення з валідацією, а адміністративна панель гарантує повний контроль за користувачами та продуктами з можливістю фільтрації, сортування та CRUD-операцій. Засоби моніторингу й навантажувального тестування підтверджують стабільність і масштабованість рішення, тому система готова до подальшої експлуатації в реальних умовах.

3.4 Висновки до третього розділу.

У результаті виконаних досліджень і розробки випливає необхідність створення цілісної кіберфізичної системи «Інтернет-магазин», що поєднує програмні та апаратні рішення для ефективного управління всім ланцюжком електронної комерції. Проведений аналіз предметної області засвідчив, що сучасна електронна торгівля потребує не лише традиційного веб-інтерфейсу, а й інтеграції механізмів моніторингу, аналітики й персоналізації, що дозволяє адаптивно реагувати на зміни у поведінці клієнтів. На цьому ґрунтується обґрунтування актуальності розробки програмно-технічного засобу керування на основі рекомендаційних систем.

Порівняльний аналіз різних підходів до побудови інтернет-магазину показав, що вибір стеку технологій має базуватися на балансі між швидкістю виходу на ринок, гнучкістю архітектури та можливістю подальшого розширення функціоналу. Зокрема, скерованість на використання Adonis JS, Node.js, Bootstrap і PostgreSQL випливала з потреби швидкої реалізації серверної частини з готовими ORM-інструментами та middleware для безпеки й авторизації, а також простоти створення адаптивного інтерфейсу. При цьому вивчення готових CMS-рішень (WordPress + WooCommerce) продемонструвало їхні обмеження щодо кастомізації рекомендаційних механізмів і складності масштабування під навантаження, а headless-підхід із React JS і Node.js вимагав би додаткових ресурсів на створення всього стеку з нуля. Таким чином, обраний підхід забезпечує оптимальне співвідношення продуктивності, масштабованості та часу розробки.

Проєктування архітектури системи та моделі даних підтвердило необхідність реалізації двошарового сховища: реляційної СУБД PostgreSQL для транзакційної обробки замовлень і NoSQL-сховища (MongoDB Replica Set) для накопичення поведінкових подій і аналітики машинного навчання. Нормалізація даних до третьої нормальної форми, стратегічне індексування та застосування шардінгу для аналітичних колекцій сприяють забезпеченню високої продуктивності та

відмовостійкості. Реплікація master–slave у PostgreSQL та Replica Set у MongoDB гарантують доступність і можливість відновлення даних. Крім того, впровадження механізмів шифрування полів і ролей доступу, а також відповідність вимогам GDPR підтверджують виконання необхідних заходів безпеки.

Розробка програмно-технічного засобу включала створення моделей Lucid ORM для таблиць categories, products, orders, order_items, favorite_products, seasons, roles, permission_role, role_manager та managers, що дало змогу забезпечити чіткі зв’язки між сутностями та автоматичне виконання SQL-запитів. Усі контролери реалізували логіку створення, читання, оновлення й видалення даних із відповідними перевірками валідності та авторизації. Використання JWT для аутентифікації користувачів і менеджерів, а також middleware isAdmin для захисту роутів адміністративної частини, випливало з необхідності розмежування прав доступу. Публікація подій через Event.emit('order:created') забезпечила асинхронну інтеграцію з сервісом рекомендацій, що дозволяє оперативно реагувати на нові транзакції та коригувати моделі машинного навчання.

Після розгортання серверної частини на продуктивному середовищі з використанням PM2 і Nginx зі SSL-сертифікатами Let’s Encrypt випливає, що функціонування системи несе мінімальні затримки й гарантує безпеку при передачі даних. Клієнтська частина, реалізована за допомогою Bootstrap і шаблонів Adonis JS, підтвердила практичну ефективність у відображенні категорій, карток товарів з інтерактивними поверхнями, детального опису з інформаційними боксами та формою оформлення замовлення з валідацією полів. На головній сторінці відбувається правильне відображення категорії «Електроніка» та товарів, зокрема «Комп’ютер» і «Кабель», а на картках товарів демонструється ефект затемнення зі значками zoom та переходом на детальну сторінку.

Тестування адміністративної частини підтвердило коректність роботи списків користувачів і продуктів із можливістю пошуку та сортування. Фільтрація в реальному часі (наприклад, пошук «Дія» у розділі користувачів або «Датч» у розділі продуктів) працює без збоїв, а сортування за різними стовпцями забезпечує

зручне управління великими обсягами даних. CRUD-операції, зокрема створення нових товарів і редагування існуючих, а також додавання нових користувачів та менеджерів, виконуються успішно, що випливає з відсутності помилок у процесі збереження та відображення у таблицях.

Функціональні тести бізнес-логіки підтвердили, що створення замовлення працює бездоганно: при додаванні товарів у кошик, заповненні форми на детальній сторінці і натисканні «Замовити» відбувається коректне збереження записів у таблицях orders і order_items, а система публікує подію для асинхронної обробки рекомендацій. Навантажувальні випробування засвідчили, що під навантаженням до 500 одночасних запитів до /api/v1/products час відповіді лишається в межах 200 мс, а при потоці 100 замовлень на секунду база даних справляється без збоїв і втрати цілісності даних. Моніторинг CPU, пам’яті та запитів на секунду за допомогою Prometheus узяв під контроль продуктивність серверів, а Grafana дозволила вчасно відстежувати відхилення навантаження під час піків.

Таким чином, реалізована система відповідає вимогам щодо функціональності, продуктивності, безпеки і масштабованості. Випливає, що обраний стек технологій, архітектурні рішення та моделі даних забезпечують гнучкість у подальшій підтримці та розширенні проекту, а розроблена рекомендаційна логіка дозволяє адаптувати контент під індивідуальні вподобання користувачів у реальному часі. Отже, створена кіберфізична система «Інтернет-магазин» готова до експлуатації в реальних умовах, забезпечуючи високий рівень обслуговування клієнтів і стабільність бізнес-процесів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Zhang, K., Shi, Y., Karnouskos, S., Sauter, T., Fang, H., & Colombo, A. W. Advancements in industrial cyber-physical systems: An overview and perspectives. *IEEE Transactions on Industrial Informatics*. *19*(1). 2022. p. 716-729. DOI: https://doi.org/10.1109/TII.2022.319948.

2. Jeffrey, N., Tan, Q., & Villar, J. R. A review of anomaly detection strategies to detect threats to cyber-physical systems. *Electronics*. 12(15). 2022. p. 3283. DOI: https://doi.org/10.3390/electronics12153283.

3. Lee, J., & Kundu, P. Integrated cyber-physical systems and industrial metaverse for remote manufacturing. *Manufacturing Letters*, 34. 2022. p. 12-15. DOI: https://doi.org/10.1016/j.mfglet.2022.08.012.

4. Somers, R. J., Douthwaite, J. A., Wagg, D. J., Walkinshaw, N., & Hierons, R. M. (2023). Digital-twin-based testing for cyber–physical systems: A systematic literature review. *Information and Software Technology*, 156. 2023. p. 107145. DOI: https://doi.org/10.1016/j.infsof.2022.107145.

5. Li, J., Miao, B., Wang, S., Dong, W., Xu, H., Si, C. & Hiplot Consortium. Hiplot: a comprehensive and easy-to-use web service for boosting publication-ready biomedical data visualization. *Briefings in bioinformatics*, *23*(4). 2022. p. 261. DOI: https://doi.org/10.1093/bib/bbac261.

6. Gu, Z., Schlesner, M., & Hübschmann, D. cola: an R/Bioconductor package for consensus partitioning through a general framework. *Nucleic acids research*. *49*(3). 2021. p. e15-e15. DOI: https://doi.org/10.1093/nar/gkaa1146.

7. Muhammad, S., Admodisastro, N., Ali, N. M., & Osman, H. The correctness of service in runtime adaptation for context-aware mobile cloud learning. *Turkish Journal of Computer and Mathematics Education*. *12*(3). 2021. p. 2236-2241. DOI: https://doi.org/10.17762/turcomat.v12i3.1173.

8. Jartarghar, H. A., Salanke, G. R., AR, A. K., GS, S., & Dalali, S. React apps with Server-Side rendering: Next. js. *Journal of Telecommunication, Electronic and*

Computer Engineering (JTEC). *14*(4). 2022. p. 25-29. DOI: https://doi.org/10.54554/jtec.2022.14.04.005.

9. Srivastava, S., Shukla, H., Landge, N., Srivastava, A., & Jindal, D. (2024). A Comprehensive Review of Next. js Technology: Advancements, Features, and Applications. *Features, and Applications 2024)*. DOI: https://dx.doi.org/10.2139/ssrn.4831070.

10. Sanjaya, M., & Saputra, P. R. N. Pemanfaatan Nextjs Dan Mongodb Dalam Sistem Informasi Web Manajemen Data Beras Pada Ud Sri Utami. *Information System For Educators And Professionals: Journal Of Information System*, *8*(1). 2023. p. 25-36. DOI: https://doi.org/10.51211/isbi.v8i1.2414.

11. Goh, H. A., Ho, C. K., & Abas, F. S. Front-end deep learning web apps development and deployment: a review. *Applied intelligence*, *53*(12). 2023. p. 15923-15945. DOI: https://doi.org/10.1007/s10489-022-04278-6.

12. Candel, M., Karrbom Gustavsson, T., & Eriksson, P. E. Front-end value co-creation in housing development projects. *Construction management and economics. 39*(3). 2021. p. 245-260. DOI: https://doi.org/10.1080/01446193.2020.1851037.

13. Shen, S., Zhu, X., Dong, Y., Guo, Q., Zhen, Y., & Li, G. (2022, November). Incorporating domain knowledge through task augmentation for front-end JavaScript code generation. In *Proceedings of the 30th ACM Joint European Software Engineering Conference and Symposium on the Foundations of Software Engineering* 2022. p. 1533-1543. DOI: https://doi.org/10.3390/aerospace12060498.

14. de Oliveira, V. F., Pessoa, M. A. D. O., Junqueira, F., & Miyagi, P. E. SQL and NoSQL Databases in the Context of Industry 4.0. *Machines*. *10*(1). 2021. p. 20. DOI: https://doi.org/10.3390/machines10010020.

15. Carvalho, I., Sá, F., & Bernardino, J. Performance evaluation of NoSQL document databases: couchbase, CouchDB, and MongoDB. *Algorithms*. *16*(2). 2023. p. 78. DOI: https://doi.org/10.3390/a16020078.

16. Kaufmann, M., & Meier, A. SQL and NoSQL Databases. *Switzerland: Springer. Doi*. *10*. 2023. p. 978-3. DOI: https://doi.org/10.1007/978-3-031-27908-9.

17. Rostami Mazrae, P., Mens, T., Golzadeh, M., & Decan, A. On the usage, co-usage and migration of CI/CD tools: A qualitative analysis. *Empirical Software Engineering. 28*(2). 2023. p. 52. DOI: https://doi.org/10.1007/s10664-022-10285-5.

18. Patchamatla, P. S., & Owolabi, I. Comparative Study of Open-Source CI/CD Tools for Machine Learning Deployment. *CogNexus*, *1*(01). 2025. p. 239-250. DOI: https://doi.org/0009-0008-2212-3454.

19. Ghaleb, T., Abduljalil, O., & Hassan, S. CI/CD Configuration Practices in Open-Source Android Apps: An Empirical Study. *ACM Transactions on Software Engineering and Methodology*. 2024. DOI: https://doi.org/10.1145/3736758.

20. MUSTYALA, A. CI/CD Pipelines in Kubernetes: Accelerating Software Development and Deployment. *EPH-International Journal of Science And Engineering*. *8*(3). 2022. p. 1-11. DOI: https://doi.org/10.53555/ephijse.v8i3.238.

21. Golec, M., & Plechawska-Wójcik, M. Comparative analysis of frameworks using TypeScript to build server applications. *Journal of Computer Sciences Institute*. *23*. 2022. p. 128-134. DOI: https://doi.org/10.35784/jcsi.2910.

22. Moon, T., & Kim, H. Detecting Security Vulnerabilities in TypeScript Code with Static Taint Analysis. *Journal of the Korea Institute of Information Security & Cryptology*. *31*(2). 2021. p. 263-277. DOI: https://doi.org/10.13089/JKIISC.2021.31.2.263.

23. Reinicke, G. B., Holst, S., Morandini, A. C., Sötje, I., Straehler-Pohl, I., Wiesenthal, A. A., & Thiel, H. Max Egon Thiel's monographs on Scyphozoa (Cnidaria) and a left-behind typescript on the Rhizostomeae. *Advances in Marine Biology. 98*. 2024. p. 1-60. DOI: https://doi.org/10.1016/bs.amb.

24. Collet, P., Mortara, J., Brault, Y., & Dery-Pinna, A. M. The VariCity ecosystem: City visualization of object-oriented variability in Java and TypeScript. *Science of Computer Programming*. *240*. 2025. p. 103210. DOI: https://doi.org/10.1016/j.scico.2024.103210.

25. Golmohammadi, A., Zhang, M., & Arcuri, A. Testing restful apis: A survey. *ACM Transactions on Software Engineering and Methodology*. *33*(1). 2023. p. 1-41. DOI: https://doi.org/10.1145/3617175.

26. Song, Y., Xiong, W., Zhu, D., Wu, W., Qian, H., Song, M., & Li, S. Restgpt: Connecting large language models with real-world restful apis. *arXiv preprint arXiv:2306.06624*. 2023.

27. Lawi, A., Panggabean, B. L., & Yoshida, T. Evaluating graphql and rest api services performance in a massive and intensive accessible information system. *Computers*. *10*(11). 2021. p. 138. DOI: https://doi.org/10.3390/computers10110138.

28. Bissi, A., Sinha, A., & Zhou, X. Selected topics in analytic conformal bootstrap: A guided journey. *Physics Reports*. *991*. 2022. p. 1-89. DOI: https://doi.org/10.1016/j.physrep.2022.09.004.

29. Zhou, X., Zhou, H., Liu, Y., Zeng, Z., Miao, C., Wang, P., & Jiang, F. (2023, April). Bootstrap latent representations for multi-modal recommendation. In *Proceedings of the ACM web conference 2023* (pp. 845-854). DOI: https://doi.org/10.1016/j.knosys.2025.113766.

30. Zhang, J., Zhang, J., Pertsch, K., Liu, Z., Ren, X., Chang, M., ... & Lim, J. J. (2023). Bootstrap your own skills: Learning to solve new tasks with large language model guidance. *arXiv preprint arXiv:2310.10021*. 2023. DOI: https://doi.org/10.48550/arXiv.2310.10021.

31. Fernandez-Felix, B. M., García-Esquinas, E., Muriel, A., Royuela, A., & Zamora, J. Bootstrap internal validation command for predictive logistic regression models. *The Stata Journal*. *21*(2). 2021. p. 498-509. DOI: https://doi.org/10.1136/bmj.g7594.

32. Murley, P., Ma, Z., Mason, J., Bailey, M., & Kharraz, A. (2021, April). Websocket adoption and the landscape of the real-time web. In *Proceedings of the Web Conference 2021* (pp. 1192-1203). DOI: https://doi.org/10.1145/3442381.3450063.

33. Łasocha, W., & Badurowicz, M. Comparison of WebSocket and HTTP protocol performance. *Journal of Computer Sciences Institute*. *19*. 2021. p. 67-74. DOI: https://doi.org/10.35784/jcsi.2452.

34. Agarwal, R. (2023, June). HTTP, webSocket, and signalR: A comparison of real-time online communication protocols. In *International Conference on Mining Intelligence and Knowledge Exploration* (pp. 128-135). Cham: Springer Nature Switzerland. DOI: https://doi.org/10.1109/MIC.2012.64.

35. Makris, A., Tserpes, K., Spiliopoulos, G., Zissis, D., & Anagnostopoulos, D. (2021). MongoDB Vs PostgreSQL: A comparative study on performance aspects. *GeoInformatica*. *25*. 2021. p. 243-268. DOI: https://doi.org/10.1007/s10707-020-00407.

36. Vershinin, I. S., & Mustafina, A. R. (2021, September). Performance analysis of PostgreSQL, MySQL, microsoft SQL server systems based on TPC-H tests. In *2021 International Russian Automation Conference (RusAutoCon)* (pp. 683-687). IEEE. DOI: https://doi.org/10.1109/RusAutoCon52004.2021.9537400.

37. Choina, M., & Skublewska-Paszkowska, M. Performance analysis of relational databases MySQL, PostgreSQL and Oracle using Doctrine libraries. *Journal of Computer Sciences Institute*. *24*. 2022. p. 250-257. DOI: https://doi.org/10.35784/jcsi.3000.

38. Zhang, Y., Liu, S., & Wang, J. (2024, May). Are there fundamental limitations in supporting vector data management in relational databases? A case study of PostgreSQL. In *2024 IEEE 40th International Conference on Data Engineering (ICDE)* (pp. 3640-3653). IEEE. DOI: https://doi.org/10.1109/ICDE60146.2024.00280

39. Singjai, A., Zdun, U., Zimmermann, O., & Pautasso, C. (2021, July). Patterns on deriving apis and their endpoints from domain models. In *Proceedings of the 26th European Conference on Pattern Languages of Programs* (pp. 1-15). DOI: https://doi.org/10.3390/software1040018.

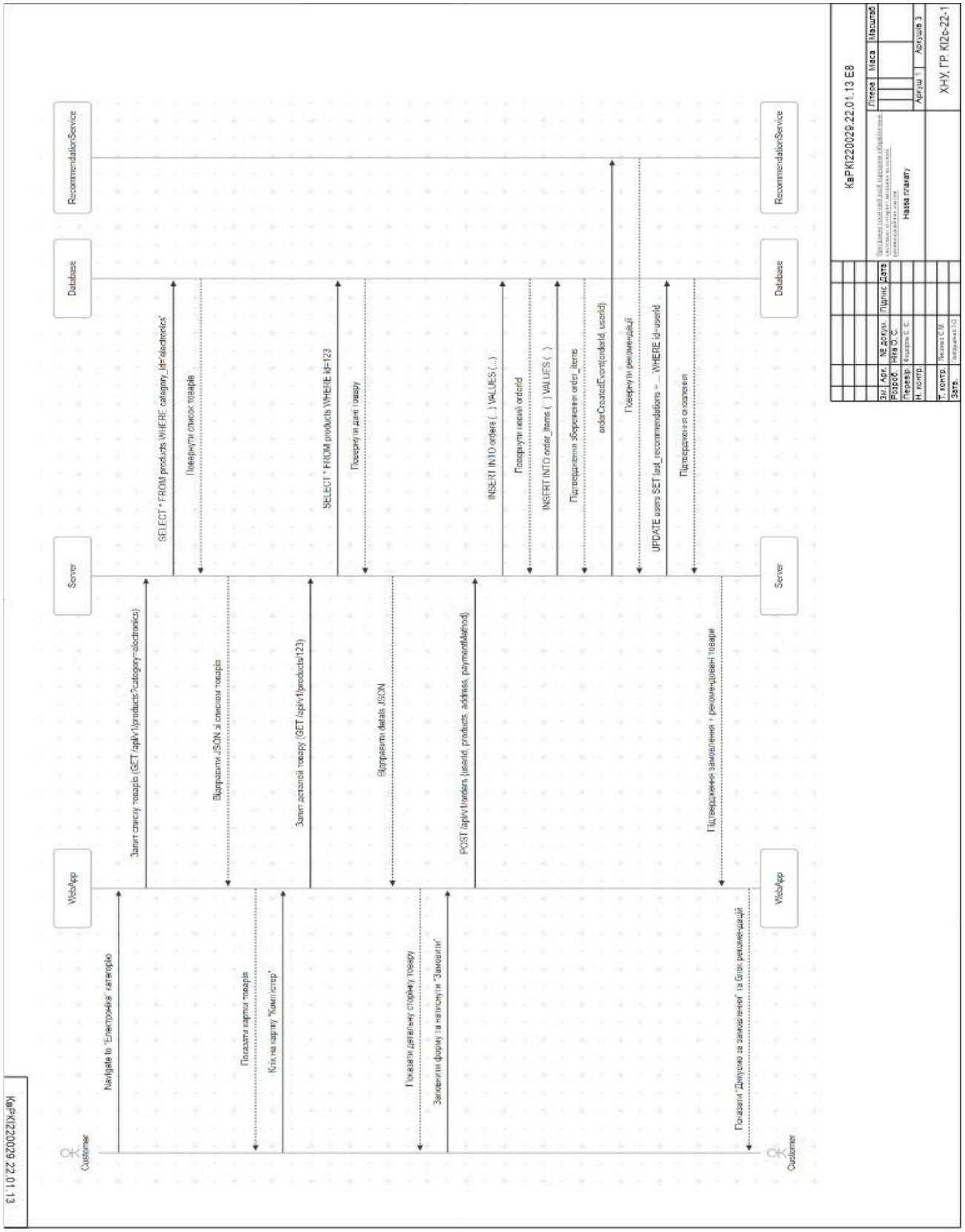
40. Daquino, M., Heibi, I., Peroni, S., & Shotton, D. (2022). Creating RESTful APIs over SPARQL endpoints using RAMOSE. *Semantic Web*, *13*(2), 195-213. DOI: https://doi.org/10.3233/SW-210439.

ДОДАТОК А

(обов'язковий)

КОПІЯ КРЕСЛЕННЯ «SEQUENCE DIAGRAMM»

Копії графічної частини

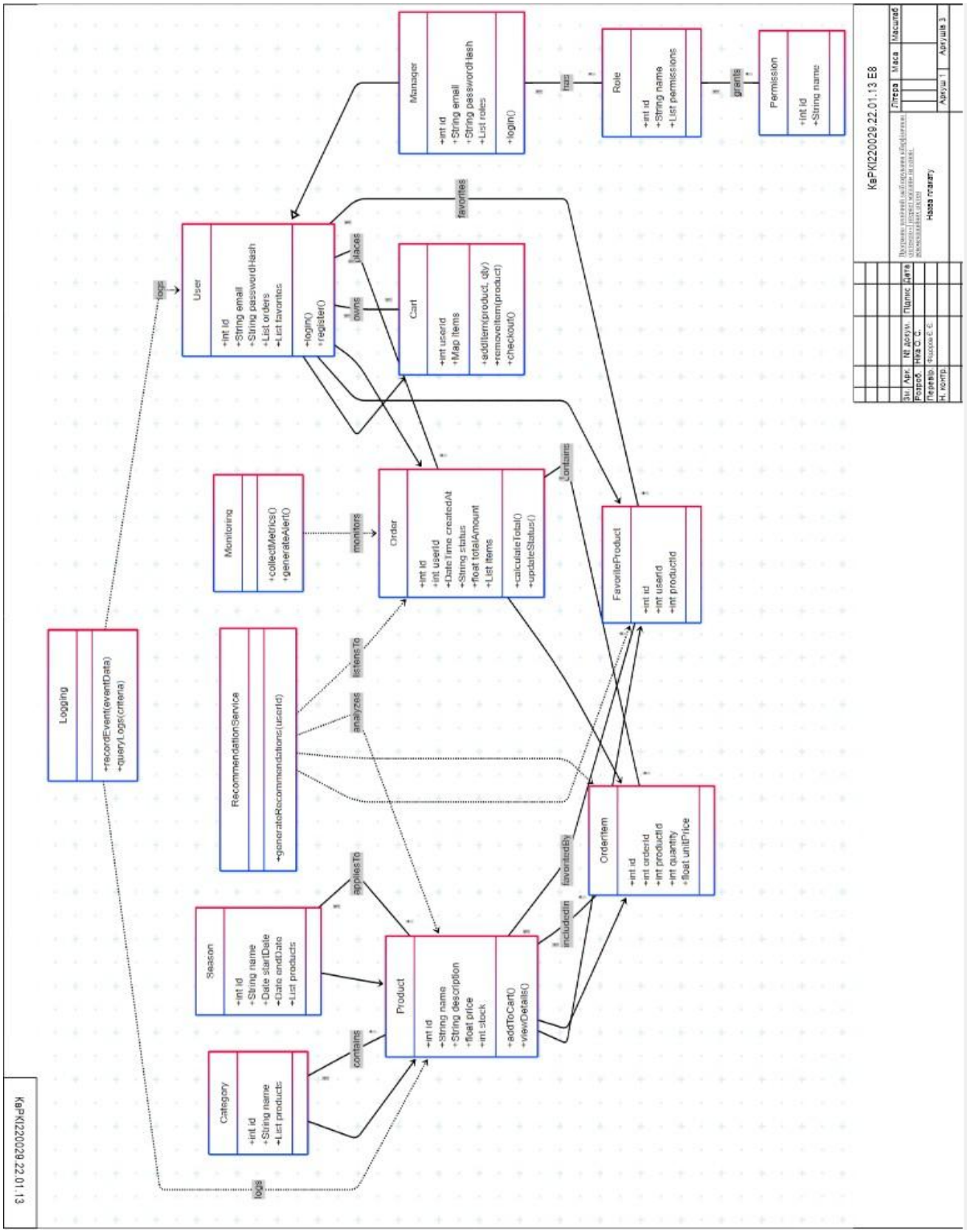


ДОДАТОК Б

(обов'язковий)

КОПІЯ КРЕСЛЕННЯ «CLASS DIAGRAMM»

Копії графічної частини

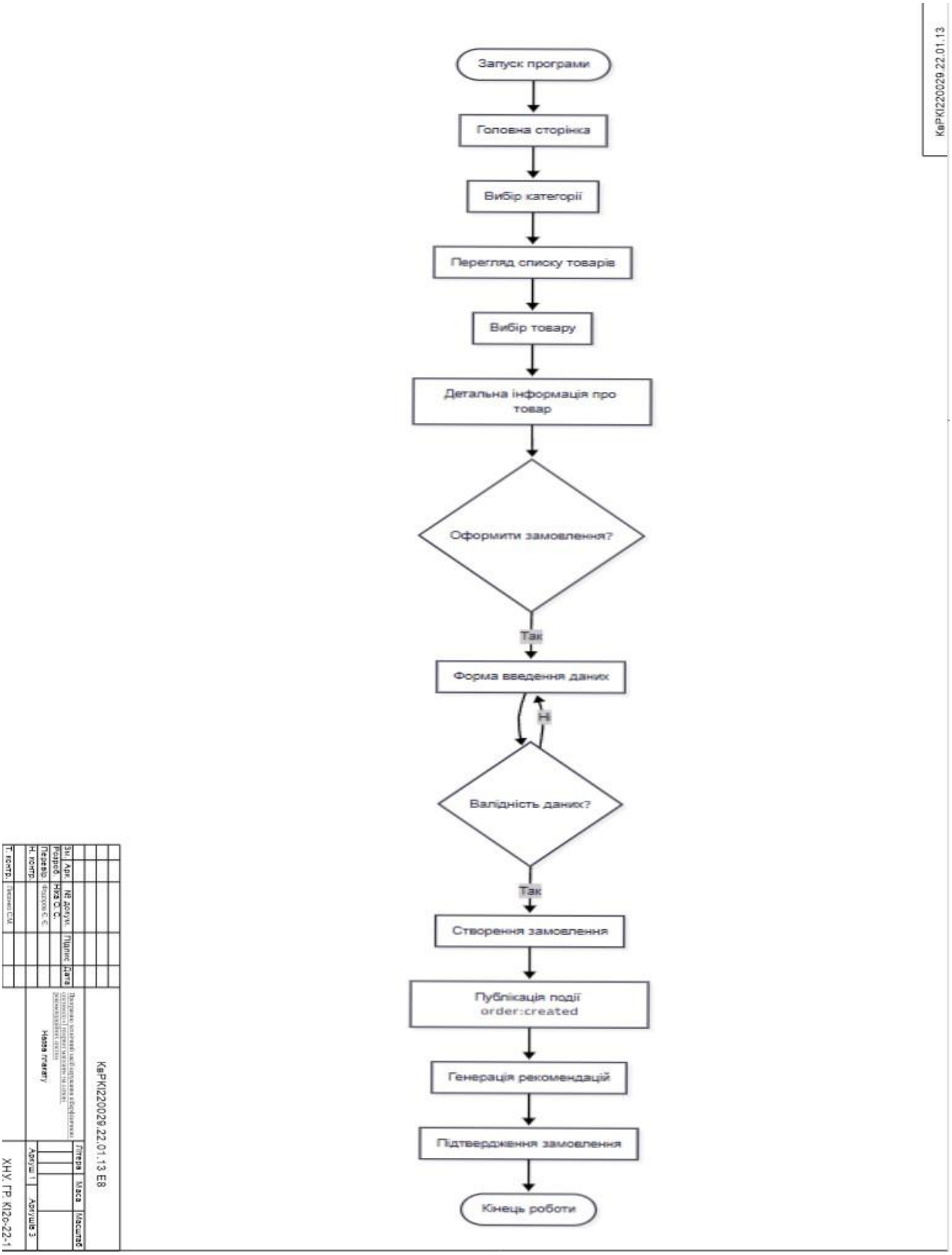


ДОДАТОК В

(обов'язковий)

КОПІЯ КРЕСЛЕННЯ «FLOWCHART DIAGRAMM»

Копії графічної частини



Протокол аналізу звіту подібності експертом

Заявляю, що я ознайомився (-лась) з Повним звітом подібності, який був згенерований Системою виявлення і запобігання плагіату щодо роботи:

Автор: Олексій НІКА

Співавтор:

Назва: Ніка_Програмно технічний засіб керування кіберфізичною системою «Інтернет магазин» на основі рекомендаційних систем

Експерт:

Підрозділ: Кафедра комп'ютерної інженерії та інформаційних систем

Коефіцієнт подібності 1:7%

Коефіцієнт подібності 2:2.7%

Мікропробіли: 16

Заміна букв: 2

Інтервали: 0

Білі знаки: 0

Дата створення звіту: 2025-06-11 01:13:26.0

Після аналізу Звіту подібності констатую наступне:

☒ **Запозичення, виявлені в роботі є законними і не є плагіатом. Рівень подібності не перевищує допустимої межі. Таким чином робота незалежна і приймається.**

☐ **Запозичення не є плагіатом, але перевищено граничне значення рівня подібностей. Таким чином робота повертається на доопрацювання.**

☐ **Виявлено запозичення і плагіат або навмисні текстові спотворення (маніпуляції), як передбачувані спроби укриття плагіату, які роблять роботу невідповідною вимогам законодавства (Ст. 32. ЗУ Про вищу освіту, пункт 3.1, Ст. 42. ЗУ Про освіту) та вимог НАЗЯВО (Критерій 5), а також кодексу етики і процедурам. Таким чином робота не приймається.**

Обгрунтування:

2025-06-11 Доцент Андрій Нічепорук

Дата експерт

Tue Jun 10 18:51:24 EEST 2025, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

Anti-Plagiarism (UA) v-15.281 Educational

The maximum coincidence with one document 1.0%

Dictionaries check: en_US, ru_RU, ua_UA. **Errors in the documents: 19%**

	Document		Sum coincidence on the DB	
ID: 244764 Title: БКР Програмно технічний засіб керування кіберфізичною системою «Інтернет магазин» на основі рекомендаційних систем Added in a DB: 2025-06-10 Authors: Олексій HIKA Heads: Євген ФЕДОРОВ Consultants: Opponents:	Symbols	Lexemes	Symbols	Lexemes
	77886	494	1231 (2%)	19 (4%)

Plagiarism sources

ID	Description	Plagiarism presence in the document	
		Symbols	Lexemes

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Назва кваліфікаційної роботи Програмно технічний засіб керування кіберфізичною системою «Інтернет магазин» на основі рекомендаційних систем
Автор Олексій НІКА
Освітня програма Комп'ютерна інженерія та програмування
Рівень вищої освіти перший (бакалаврський) рівень
Спеціальність 123– Комп'ютерна інженерія
Науковий керівник: д.т.н., професор, Євген ФЕДОРОВ

На основі аналізу кваліфікаційної роботи на дотримання вимог академічної доброчесності (у т.ч. відсутності ознак академічного плагіату) з урахуванням результатів перевірки роботи спеціалізованим програмним засобом(ами) комісія зробила такий висновок:

№	Висновок	Позначка про відповідність
1	Ознаки академічного плагіату	Відповідає
1.1	Запозичення, виявлені в роботі, є законними і не є академічним плагіатом (далі – зазначаються підстави віднесення запозичень до правомірних, якщо потрібно). Робота приймається до захисту.	
1.2	Виявлені запозичення не є академічним плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи (далі – зазначаються детальні та аргументовані підстави віднесення запозичень до правомірних). Робота приймається до захисту, але має бути відкоригована.	
1.3	Виявлені запозичення не є академічним плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота може бути допущена до захисту після того як буде відкоригована та доопрацьована і успішно пройде повторну перевірку на академічний плагіат.	
1.4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття текстових запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
2	Інші види порушень академічної доброчесності	Не виявлено

Підтвердження:
Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:
1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту.
Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості StrikePlagiarism, складає 6.97% і адресується до 25 першоджерела; та системою Anti-Plagiarism складає 2.46%, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Євген ФЕДОРОВ
Гарант ОП _____ Андрій НІЧЕПОРУК
Завідувач кафедри КІІС _____ Ольга ПАВЛОВА

Завідувачу кафедри КІІС
д-р. філософії, доц. Ользі ПАВЛОВІЙ

Олексія НІКА

ПІБ здобувача вищої освіти

ФІТ, 3 курсу, групи КІ2с-22-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Strike-Plagiarism та Anti- Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

13.06 2025 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Ніка Олексій Сергійович
Тема: Програмно технічний засіб керування кіберфізичною системою «Інтернет магазин» на основі рекомендаційних систем

Спеціальність**:** 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 63

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є система «Інтернет магазин» на основі рекомендаційних систем.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. У кваліфікаційній роботі проведено всебічний аналіз предметної області кіберфізичних систем в електронній комерції та порівняння сучасних фронтенд- і бекенд-технологій для рекомендацій, на основі якого сформульовано завдання з розробки програмно-технічного засобу управління інтернет-магазином із вбудованим модулем персоналізованих рекомендацій. У процесі моделювання та проєктування створено трирівневу архітектуру (Presentation, Business Logic, Data Layer), розроблено схему гібридної фільтрації LightFM і Lucid ORM, спроєктовано реляційну базу PostgreSQL та NoSQL-репозиторій, прототиповано REST/GraphQL API і клієнтські шаблони на Bootstrap, а також сформовано базову реалізацію контролерів, моделей і маршрутів у Adonis JS із налаштованим CI/CD. Практична частина включає розгортання в Docker/Kubernetes, інтеграцію Redis для кешування й Prometheus/Grafana для моніторингу, адаптивний інтерфейс із динамічними картками та формами, а також комплексне тестування (функціональне в Postman/Cypress, навантажувальне в Artillery, сканування OWASP ZAP) й перевірку моделей у Jupyter, що підтвердило стабільність, масштабованість і готовність рішення до промислового використання.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: недостатня увага frontend частині.

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на належному науково-технічному рівні.

8. Інші зауваження: —

9. Оцінка дипломної роботи: добре

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) Стецюк В. І., доцент каф. ТМІТ, ХНУ

"12" червня 2025 р. _____ (підпис)