Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2025 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація веб-орієнтованої системи управління онлайн-замовленнями з використанням інтегрованих блокчейн-механізмів захисту”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-24М
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Черновол В.Є.
«____» _____ 2025 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Якименко Н.М.
«____» _____ 2025 р.
Рецензент _____
_____

м. Кропивницький

АНОТАЦІЯ

Черновол В.Є. Розробка веб-орієнтованої системи електронної комерції з використанням технології блокчейн. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2025.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для захищеного проведення транзакцій та зберігання даних про замовлення в розподіленому реєстрі.

Метою розробки є дослідження та програмна реалізація системи електронної комерції з інтегрованим модулем блокчейн для забезпечення цілісності даних.

Об'єктом дослідження є процеси обробки та захисту інформації в сучасних веб-системах електронної комерції.

Предметом дослідження є методи криптографічного хешування та алгоритми побудови розподілених реєстрів (блокчейн).

Методи дослідження базуються на методах об'єктно-орієнтованого програмування, алгоритмах криптографічного захисту інформації, технологіях веб-розробки та методах роботи з базами даних.

Результат роботи – програмна реалізація веб-системи управління замовленнями з функцією фіксації транзакцій у блокчейні.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів.

Розроблено зручний веб-інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11 або Linux.

Програму розроблено мовою Python з використанням фреймворку Flask.

Ключові слова: комп'ютерна інженерія, блокчейн, електронна комерція, веб-розробка, хешування.

ABSTRACT

Chernovol V.Y. Development of a web-based e-commerce system using blockchain technology. 123 Computer Engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2025.

In this Master's thesis (second level of higher education), software designed for secure transaction processing and order data storage in a distributed ledger has been developed.

The purpose of the development is the research and software implementation of an e-commerce system with an integrated blockchain module to ensure data integrity.

The object of the research is the processes of information processing and protection in modern e-commerce web systems.

The subject of the research is methods of cryptographic hashing and algorithms for constructing distributed ledgers (blockchain).

The research methods are based on object-oriented programming techniques, cryptographic information security algorithms, web development technologies, and database management methods.

The result of the work is the software implementation of an order management web system with the function of recording transactions in the blockchain.

During the development of the software model, an analysis of existing hardware and software tools was performed.

A user-friendly web interface has been developed. Instructions for using the software tools are provided.

The software can be used on personal computers running Windows 10/11 or Linux operating systems.

The software was developed in Python using the Flask framework.

Keywords: computer engineering, blockchain, e-commerce, web development, hashing.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ 3
ВСТУП 4
1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ 6
1.1 Призначення системи 6
1.2 Область застосування 7
2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ 11
2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти 11
2.2 Обґрунтування вибору засобів для побудови системи та мови програмування 20
2.3 Розгорнута постановка завдання 25
3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ 26
3.1 Опис функціонування системи 26
3.2 Розробка структурної схеми 30
3.3 Розробка функціональної схеми 32
3.4 Розробка діаграми процесів 35
4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ 38
4.1 Розробка блок-схем та опис алгоритмів функціонування системи 38
4.2 Захист розробленого програмного забезпечення 52
5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ 56
6 НАУКОВА НОВИЗНА 62

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБҐРУНТУВАННЯ ІТ-ПРОЄКТУ 65
7.1 Визначення цільової аудиторії кінцевого готового продукту 65
7.2 Оцінка привабливості шляхом застосування методів експертних оцінок ... 66
7.3 Вибір методу оцінки вартості ПЗ 69
7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості 70
7.5 Пропозиція алгоритму просування проєкту розробки ПЗ 71
7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ 72
7.7 Визначення ключових факторів успіху конкретного проєкту 73
8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ 74
8.1 Вступ 74
8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером 76
8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста ... 77
8.4 Розробка заходів з умов поліпшення охорони праці 79
8.5 Розрахункова частина 79
9 ОСНОВНІ ВИСНОВКИ 82
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 84

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ПЗ	–	програмне забезпечення
ООП	–	об’єктно орієнтоване програмування
БД	–	база даних
ОС	–	операційна система
СУБД	–	система управління базами даних
ЦА	–	цільова аудиторія

ВСТУП

Актуальність теми. Сучасний розвиток електронної комерції супроводжується великими обсягами онлайн-транзакцій які потребують підвищення вимог до безпеки даних. Класичні веб-орієнтовані системи управління інтернет замовленнями, які створені на реляційних базах даних, мають слабину до нелегального втручання в історію операцій. Адміністратори баз даних чи зловмисники, які незаконно отримали вільний доступ до системи, можуть потаємно змінити суму оплати, статус замовлення, деталі доставки, тощо. Підключення технології блокчейн дозволяє уникнути проблеми цілісності даних та довіри. Впровадження хешування та зв'язності блоків робить будь-який намір підміни даних очевидним, тому що це порушує цілісність усього ланцюга блоків. Попри те використання публічних блокчейнів таких як Ethereum є дуже дорогим та повільним для стандартних інтернет-магазинів. Тому завдання розробки веб-системи з інтегрованим легковаговим механізмом блокчейн-захисту, що буде забезпечувати незмінність аудиту транзакцій без потреби використання криптовалют на наш час є актуальним.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконана згідно з напрямком наукових досліджень кафедри кібербезпеки та програмного забезпечення у сфері захисту інформаційних систем та впровадження новітніх веб-технологій.

Мета і задачі дослідження. Мета роботи є підвищення рівня захисту даних у сфері транзакції в системах електронної комерції через шлях дослідження та програмного виконання веб-орієнтованої системи управління замовленнями з використанням технології ланцюга блоків для того щоб фіксувати стани системи.

Для досягнення мети були поставлені такі завдання:

– Провести аналіз існуючих систем управління онлайн-замовленнями, також виявити їхні діри у сфері безпеки даних.

– Обґрунтувати вибір засобів розробки та технологій для реалізації захищеної веб-системи.

– Розробити Архітектуру системи разом з алгоритмом формування ланцюга блоків для того щоб захистити цілісність транзакцій.

– Зробити програмну реалізацію серверної частини на мові Python та клієнтської частини

– Провести роботу над тестуванням розробленої системи та зробити оцінку її ефективності.

Об’єкт дослідження. Процес зберігання та обробки даних про транзакції в веб-орієнтованих системах електронної комерції.

Предмет дослідження. Методи та програмні засоби забезпечення цілісності даних онлайн-замовлень за допомогою використання технологій блокчейну.

Методи дослідження. У роботі було використано методи системного аналізу для дослідження предметної області, методи ООП для реалізації програмних модулів, методи хешування криптографії (SHA-256) для гарного забезпечення захисту даних та методи тестування програмного забезпечення які заточені на перевірку правильності роботи системи.

Наукова новизна одержаних результатів полягає в тому що була проведена робота над удосконаленням методу захисту історії статусів замовлень у веб-системах, який на відміну від вже існуючих системах оплати транзакцій користується гібридною схемою зберігання даних, а саме реляційною базою даних для швидкодії та локального блокчейну для аудиту, що забезпечує перевірку цілісності без значних втрат ресурсів. Також програмне забезпечення має подльший розвиток застосування мікро-сервісної архітектори на базі Python для побудови захищених модулів електронної комерції.

Розроблена система може використовуватися середніми та малими підприємствами електронної комерції для забезпечення прозорості обробки замовлень та найголовніше захисту від шахрайства з третіх осіб та персоналу.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Розроблювана веб-орієнтована система, являє собою спеціалізоване програмне забезпечення, яке в свою чергу призначене для комплексної автоматизації процесів обробки, прийому, супроводу та зберігання історії онлайн замовлень в яких є посилений інтегрований контур інформаційної безпеки. Головне призначення цього продукту є вирішення проблеми довіри юзерів до даних у системах електронної комерції за допомогою ускладнення чи унеможливлення потаємної підробки історії транзакцій. Відмінно від усім відомих звичайних систем управління контентом або класичних CRM-систем що спрямовані тільки на зручності продажників, представлена розробка ставить на перше місце незмінність, цілісність та безпеку даних, що досягається за допомогою впровадження технологій розподіленого реєстру або їх полегшених аналогів.

Функції системи призначені охоплювати широкий спектр завдань, зв'язаних із життєвим циклом замовлення. По-перше, програмний комплекс забезпечує просту можливість взаємодії з кінцевим клієнтом через веб-інтерфейс, дозволяючи клієнтам зручно переглядати каталог товарів, створювати замовлення та робити свій кошик покупок. При цьому всьому система фіксує усі початкові параметри транзакції, такі як перелік товарів які обрав користувач, їх вартість на момент придбання, ідентифікатор користувача та часову мітку створення заявки. Ці вхідні дані слугують головним фундаментом для майбутньої обробки та підлягають суворому контролю цілісності.

Ключовий аспект призначений у системі - Це програмна реалізація механізмів захисту транзакцій на базі блокчейн технологій. Система виконує функцію створення криптографічно зв'язаного ланцюга блоків даних, Де кожен зроблений запис має унікальний цифровий відбиток попереднього запису. Такий

підхід дає гарантію на те що будь-яка спроба, показників або вилучення запису адміністратором БД чи зловмисником буде призводити до порушення математичної цілісності всього ланцюга. Це дозволяє використовувати дану систему як надійний інструмент аудиту, у якому історія операцій є виключно прозорою та захищеною від маніпуляцій.

Не рахуючи функцій захисту, система має забезпечувати робочий процес працівників, а саме менеджерів інтернет-магазину. Вона надає інструменти для зміни статусу замовлень відповідно до бізнес-логіки підприємства, наприклад, комплектацію товару на складі, передача конкретного товару у службу доставки та підтвердження оплати, тощо. Усі ці дії виконані що виконані уповноваженою особою, не просто оновлюють статус у реляційній БД, а генерую новий блок у захищеному реєстрі, що створює нерозривний ланцюг хронології подій, який як вже було сказано виключає втручання третьої особи. Саме таким чином, система виступає гарантом, що нинішній стан замовлення є результатом легальної послідовності дій, а не наслідком технічного збою або навмисного втручання.

Одним з найважливіших призначень розробки є також надання можливості верифікації даних. Система містить модулі програми, завдяки яким з'являється можливість у будь-який момент часу проводити автоматичну перевірку валідності всього ланцюга блоків. Це необхідно для оперативного виявлення інцидентів інформаційної безпеки. У момент виявлення розбіжності між збереженим хешем та дійсним станом даних, система подає сигнал адміністратору про дискредитацію, що дозволяє швидко реагувати на загрози. Через це програмний продукт поєднує в собі зручність веб-орієнтованих сервісів з надійністю криптографічних протоколів захисту інформації.

1.2 Область застосування

Область застосування системи яка розроблюється займає сферу роздрібної торгівлі та електронної комерції, але має специфіку у спрямуванні на сегменти

ринку, коли питання довіри та підтвердження правдивості транзакцій є дійсно критично важливими. У глобальному сенсі система орієнтована на використання організаціями та підприємствами малого та середнього бізнесів, що прагнуть забезпечити найкращий рівень безпеки своїх торговельних операцій уникаючи залучення зайвих ресурсів на впровадження важковагових корпоративних блокчейн-платформ, тобто максимально заощадити ресурси компанії.

Впровадження цієї системи має великий сенс в інтернет-магазинах, які спеціалізуються на торгівлі товарів великої вартості, таких як ювелірні вироби, антикваріат, техніка або спеціалізоване обладнання, тому що у таких сферах ціна помилки є надзвичайно високою, а шахрайство більш розповсюджене та дуже б'є по бізнесу. Через це наявність гарантованої історії замовлень, яку є неможливим підробити, стає конкурентною перевагою та одною з найнеобхідніших умов безпеки бізнесу. Також система може бути ефективно застосована у сфері логістики ланцюгів постачання, де треба фіксувати кожен етап переміщення товару та зміни відповідальних осіб з точною прив'язкою до часу.

З технічної точки зору, область застосування системи також поширюється на будь-які платформи, які здатні підтримувати роботу веб-сервера та інтерпретатора мови програмування Python. Це робить систему незалежною від специфічного обладнання та додає їй універсальності. Серверна частина може функціонувати під управлінням ОС сімейства Linux або Windows, що дозволяє розгортати цю систему як на особистих серверах підприємства, так і клауд сервісах. Для роботи з клієнтською частиною користувачам потрібен тільки веб-браузер, що забезпечує кросплатформеність та доступність сервісу з ПК, планшетів або смартфонів.

Окремою областю застосування системи є внутрішній корпоративний огляд та контроль дій персоналу. ПЗ може використовуватися як інструмент для моніторингу роботи менеджерів з продажу, виключаючи ситуації, коли працівники вносять незаконні зміни в замовлення заради власної вигоди або приховування власних помилок які були допущені з-за необачності співробітника. Це робить систему актуальною для компаній, що прагнуть мінімізувати ризики внутрішнього

шахрайства та підвищити коректність обробки заявок.

Економічна область застосування має обмеження від проектів, де використання публічних блокчейнів таких як Ethereum або Bitcoin є недоцільним через високу вартість комісії та низьку швидкість обробки даних. Запропоноване рішення займає нішу дозволених або приватних реєстрів, що дає нульову вартість при запису транзакцій у середині системи та миттєву обробку замовлень без затримок, що є одним з найважливішим фактором для динамічного середовища сучасної електронної комерції. Саме таким чином, система може бути рекомендована до використання як основна платформа онлайн-торгівлі або як додатковий модуль до вже існуючих архітектурних рішень.

Ще одним важливим шляхом застосування розроблюваної системи є забезпечення бази доказів у вирішенні непорозумінь чи спірних ситуацій між покупцем та продавцем, що часто виникають у процесі онлайн торгівлі. В умовах, коли офлайн контакт між учасниками угоди відсутній, то електронні записи про статус замовлення стають єдиним юридично значущим документом підтвердження угоди. Завдяки використанню технології реєстру який залишається незмінним, система може бути використана як інструмент при розгляді скарг від клієнтів стосовно невиконання термінів доставки чи некоректної комплектації замовлення. Наявність верифікованого ланцюга блоків дозволяє встановити точний час усіх змін стану замовлення, що унеможливлює маніпуляції з датами відправки, або отримання товару з боку недобросовісних працівників.

Окрім того, область використання системи поширюється на сценарії гарантійного обслуговування. Традиційні БД дозволяють ввести зміни в дату продажу, що може використовуватися для нелегального подовження гарантійного терміну для окремих клієнтів, а це грає на руку недобросовісних працівників та приносить збитки компанії. Використання технології блокчейн-фіксації у систему управління замовленнями робить дату гарантійного періоду незмінною. Саме це робить систему привабливою для використання в мережах сервісів контролю якості, забезпечуючи чесність гарантійної політики компанії.

Таким чином, сфера використання цього продукту виходить за рамки стандартного інтернет-магазину, перетворюючись у інструмент забезпечення цифрової довіри та корпоративної безпеки.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур та програмних рішень за профілем теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

В умовах стрімкої цифровізації світової економіки сектор електронної комерції демонструє стабільне зростання, стаючи основним каналом збуту як для ритейлу, так і для B2B-сегменту. Сучасні програмні рішення, що забезпечують функціонування інтернет-магазинів та торговельних майданчиків, досягли високого рівня технологічної зрілості з точки зору зручності користувача (UX), маркетингових інструментів та інтеграції з платіжними шлюзами. Проте, аналіз архітектурних особливостей найбільш поширених систем управління контентом (CMS) та спеціалізованих E-commerce платформ виявляє фундаментальну проблему, яка часто ігнорується на етапі проектування - це проблема гарантування цілісності та незмінності історії транзакцій в умовах довіри до адміністратора системи. Більшість існуючих на ринку рішень, що відносяться до покоління Web 2.0, базуються на класичній клієнт-серверній архітектурі з використанням реляційних систем управління базами даних (СУБД). Така архітектура передбачає наявність централізованого сховища даних, повний контроль над яким має адміністратор сервера або власник бізнесу. З точки зору інформаційної безпеки це створює критичну вразливість: особа, що володіє привілейованим доступом (root, superadmin), має технічну можливість виконувати операції модифікації (UPDATE) або видалення (DELETE) записів у базі даних без залишення слідів у журналах аудиту, які часто зберігаються на тому ж самому сервері. В умовах зростання ризиків інсайдерського шахрайства та кібератак, спрямованих на підміну фінансової звітності, такий підхід стає неприйнятним. Для обґрунтування необхідності створення спеціалізованої системи із захищеним реєстром доцільно провести

детальний аналіз архітектури та механізмів безпеки трьох найбільш популярних платформ: Magento (Adobe Commerce), WooCommerce та OpenCart.

Однією з найбільш потужних та функціональних систем на ринку є платформа Magento, яка нині розвивається під егідою Adobe. З архітектурної точки зору Magento є складним модульним рішенням, побудованим на базі Zend Framework (Laminas), що використовує патерн MVC (Model-View-Controller). Ключовою особливістю даної системи є використання моделі даних EAV (Entity-Attribute-Value), яка дозволяє зберігати дані про об'єкти (товари, замовлення, клієнтів) у максимально гнучкому вигляді, розподіляючи атрибути по багатьох зв'язаних таблицях. Ця особливість забезпечила Magento лідерство у сегменті середнього та великого бізнесу, оскільки система дозволяє масштабуватись та кастомізуватись під будь-які потреби. Плагінна архітектура дозволяє розширювати функціонал без втручання в ядро системи, а розвинена екосистема розширень покриває практично всі маркетингові та логістичні задачі. Однак, саме складність архітектури Magento створює специфічні проблеми в контексті контролю цілісності даних. Розподіл інформації про одне замовлення по десятках таблиць (таких як sales_flat_order, sales_flat_order_item, sales_flat_order_payment тощо) ускладнює проведення ручного аудиту бази даних.

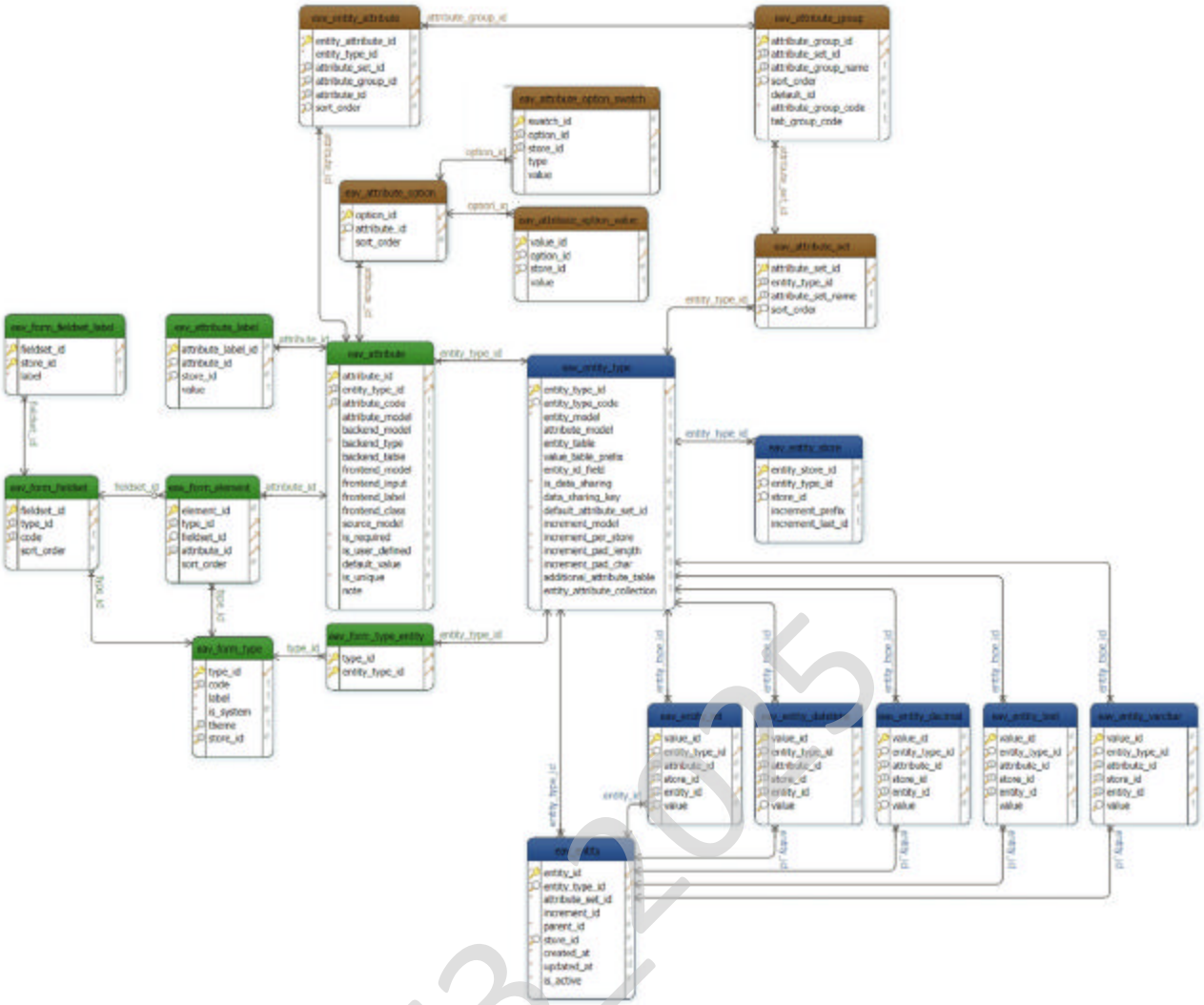


Рисунок 2.1 – Фрагмент структури БД Magento

Привілейований користувач, який має доступ до панелі адміністрування або безпосередньо до бази даних MySQL, може змінити статус замовлення або суму транзакції, і виявити таку зміну серед сотень тисяч записів EAV-моделі вкрай важко.

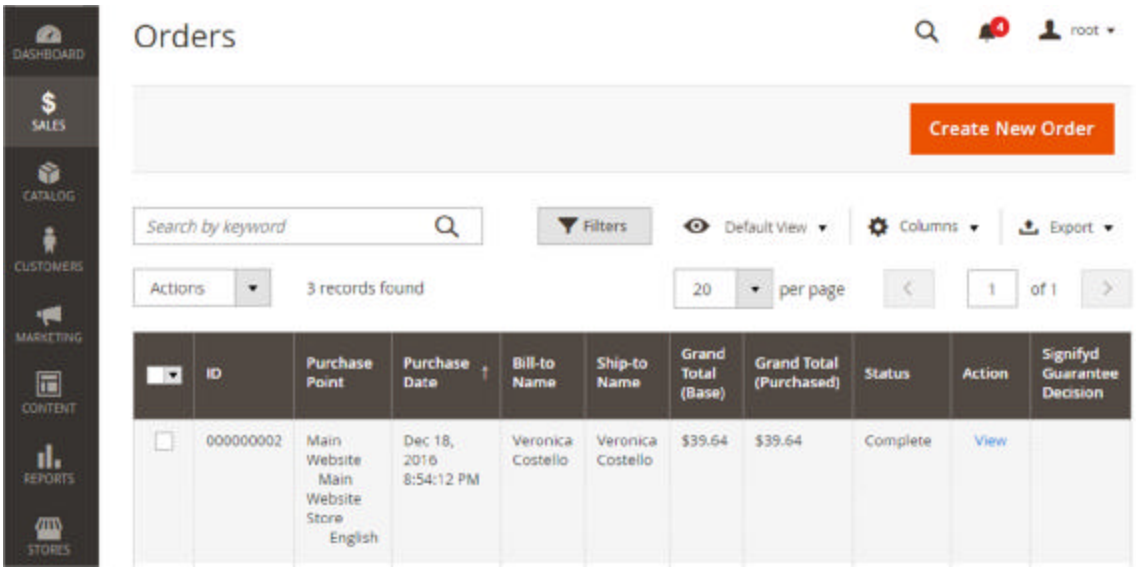


Рисунок 2.2 - Інтерфейс управління замовленнями Magento

Стандартні засоби логування в Magento фіксують події на рівні додатка, записуючи їх у текстові файли або окремі таблиці БД, які не захищені від редагування. У випадку компрометації облікового запису адміністратора зловмисник отримує повний контроль над історією операцій, маючи можливість підмінити дані для приховування крадіжки товарів або фінансових махінацій. Відсутність вбудованого механізму хешування ланцюжка транзакцій робить неможливим математичне доведення автентичності історії замовлень.

Другим об'єктом аналізу є система WooCommerce, яка фактично є плагіном для найпоширенішої у світі CMS WordPress. Завдяки низькому порогу входження, безкоштовній моделі розповсюдження та величезній спільноті розробників, WooCommerce займає значну частку ринку малого бізнесу. Архітектура системи базується на ядрі WordPress, яке використовує подійно-орієнтований підхід (Event-driven architecture) та систему хуків (Hooks API). Дані про замовлення зберігаються у загальних таблицях CMS, призначених для контенту (wp_posts) та метаданих (wp_postmeta). Така уніфікація спрощує розробку, але є катастрофічною з точки зору безпеки даних.

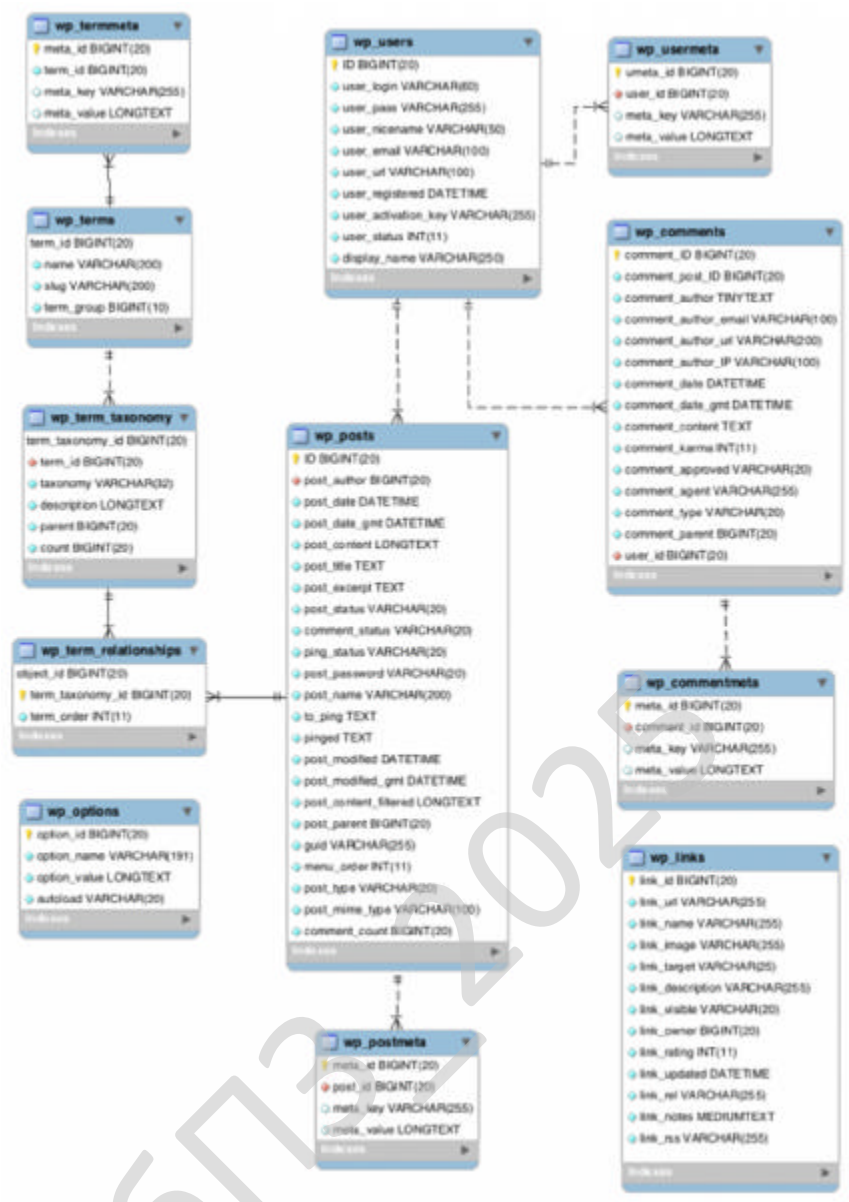


Рисунок 2.3 - Структура БД WordPress

Головним недоліком WooCommerce в розрізі даного дослідження є архітектурна вразливість структури бази даних. Оскільки вся фінансова інформація, включаючи суми замовлень, статуси оплат та персональні дані клієнтів, зберігається у таблиці метаданих у вигляді пар "ключ-значення", це відкриває широкі можливості для маніпуляцій. Виконання одного SQL-запиту UPDATE дозволяє змінити вартість товару в уже оформленому замовленні, і система сприйме це як легітимну дію.

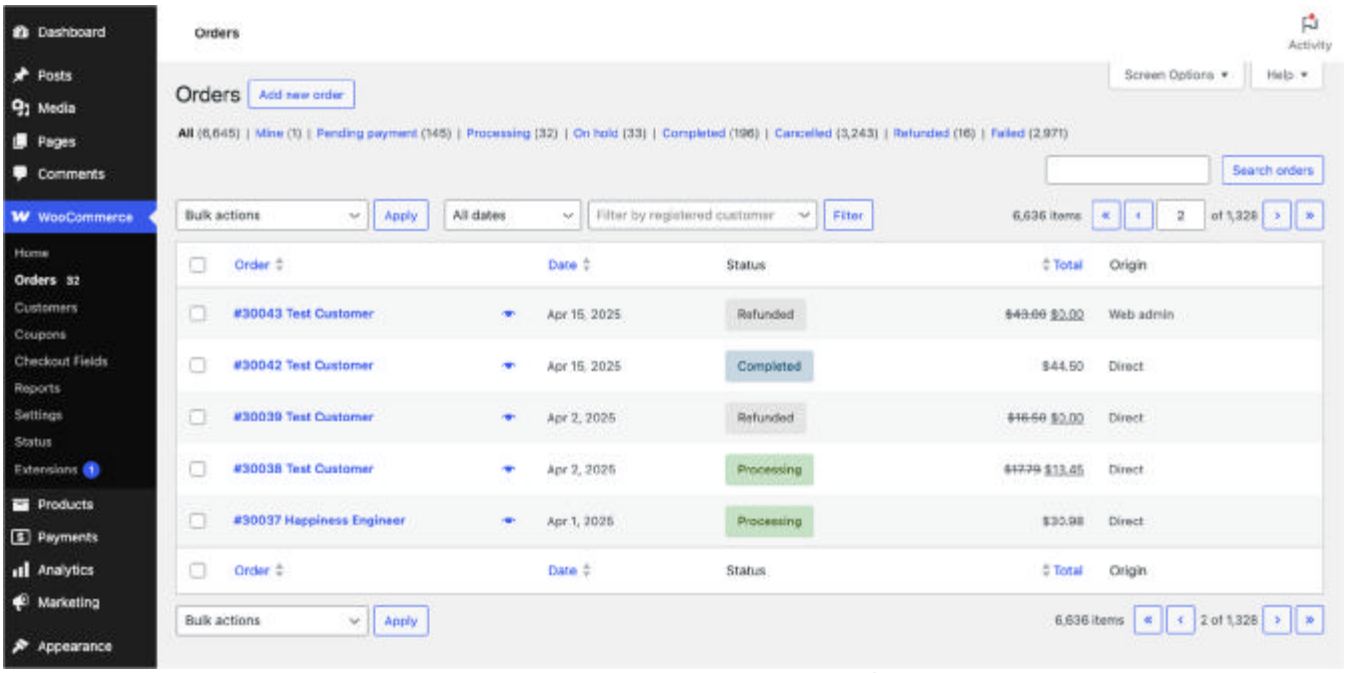


Рисунок 2.4 – Інтерфейс редагування замовлення, що дозволяє зміну фінансових даних

Більше того, величезна кількість сторонніх плагінів, які часто пишуться розробниками з низькою кваліфікацією, створює численні вектори атак типу SQL Injection. Якщо зловмисник отримує доступ до бази даних через вразливий плагін, він може непомітно модифікувати історичні дані. У системі відсутнє поняття незмінного журналу подій (Immutable Log): історія змін замовлення ("Order Notes") є просто текстовим полем, яке також може бути відредаговане або видалене адміністратором. Це робить WooCommerce абсолютно непридатною для використання в середовищах, де вимагається сувора фінансова звітність та гарантія відсутності інсайдерських зловживань.

Третьою розглянутою системою є OpenCart - спеціалізована платформа для електронної комерції, яка позиціонується як легке та швидке рішення. Архітектура OpenCart базується на шаблоні MVC-L (Model-View-Controller-Language) і відрізняється від попередніх систем чіткою, нормалізованою структурою реляційної бази даних.

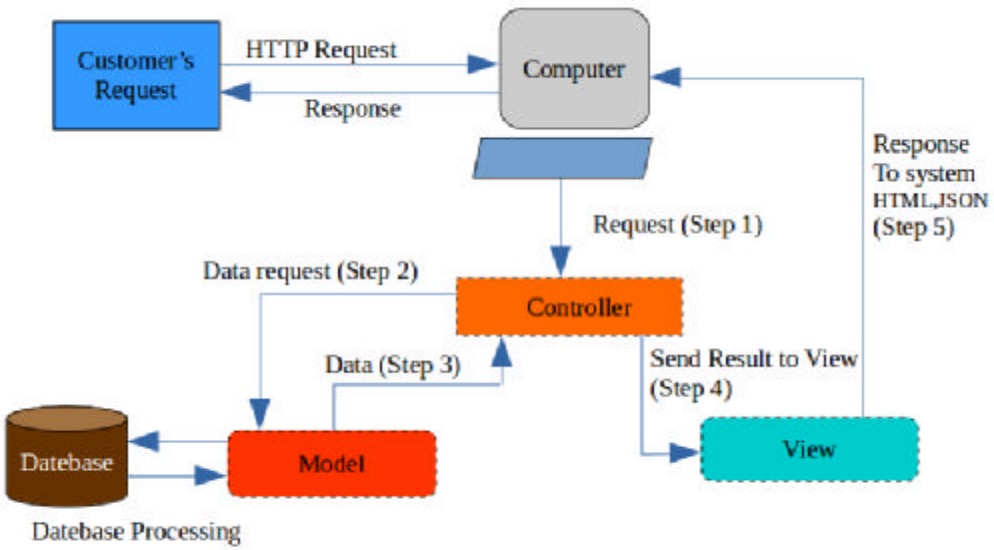


Fig: Opencart MVC Architecture Pattern

Рисунок 2.5 – Архітектура обробки запиту в системі OpenCart

Це спрощує роботу з системою для розробників та знижує навантаження на сервер, що є безперечною перевагою для невеликих магазинів. Система має вбудовану систему звітів та базовий функціонал для управління замовленнями "з коробки".

Втім, аналіз механізмів безпеки OpenCart вказує на ті ж самі концептуальні недоліки, притаманні централізованим системам. Специфічною проблемою OpenCart є система модифікаторів OCMOD (або застаріла VQMOD), яка дозволяє змінювати вихідний код ядра системи "на льоту" за допомогою XML-інструкцій. Це створює небезпечний вектор атаки: інсайдер або зловмисник може завантажити модифікатор, який буде перехоплювати дані транзакцій або змінювати логіку розрахунку вартості безпосередньо в оперативній пам'яті сервера, не змінюючи файли на диску. Як і в попередніх випадках, контроль цілісності бази даних покладається виключно на довіру до персоналу. Таблиця історії замовлень oc_order_history є звичайною таблицею MySQL, записи в якій не захищені криптографічними підписами. Видалення запису про зміну статусу замовлення не призводить до порушення цілісності структури БД, тому виявити факт "чистки"

історії постфактум практично неможливо без наявності зовнішніх бекапів, до яких адміністратор також зазвичай має доступ.

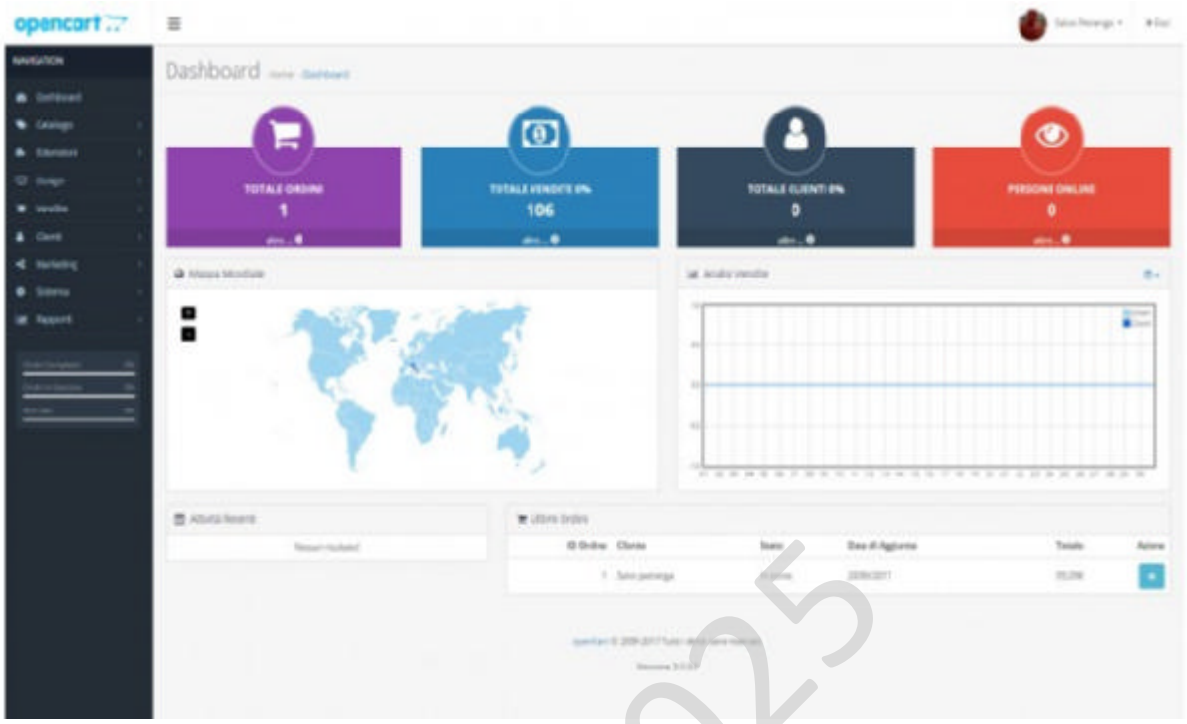


Рисунок 2.6 – Структура взаємозв’язків таблиць замовлень у OpenCart.

Слід також згадати про існування хмарних SaaS-рішень (наприклад, Shopify), які вирішують проблему безпеки периметра, делегуючи її провайдеру послуг. Однак такі системи є "чорною скринькою" для власника бізнесу. Користувач не має прямого доступу до бази даних, але так само не має і інструментів для незалежного аудиту. Цілісність даних гарантується лише репутацією провайдера, що не вирішує проблему, якщо загроза виходить від співробітника компанії, який має легітимний доступ до панелі управління магазином. Крім того, SaaS-модель створює залежність від постачальника (Vendor Lock-in) та унеможливлює впровадження кастомних алгоритмів захисту на рівні ядра.

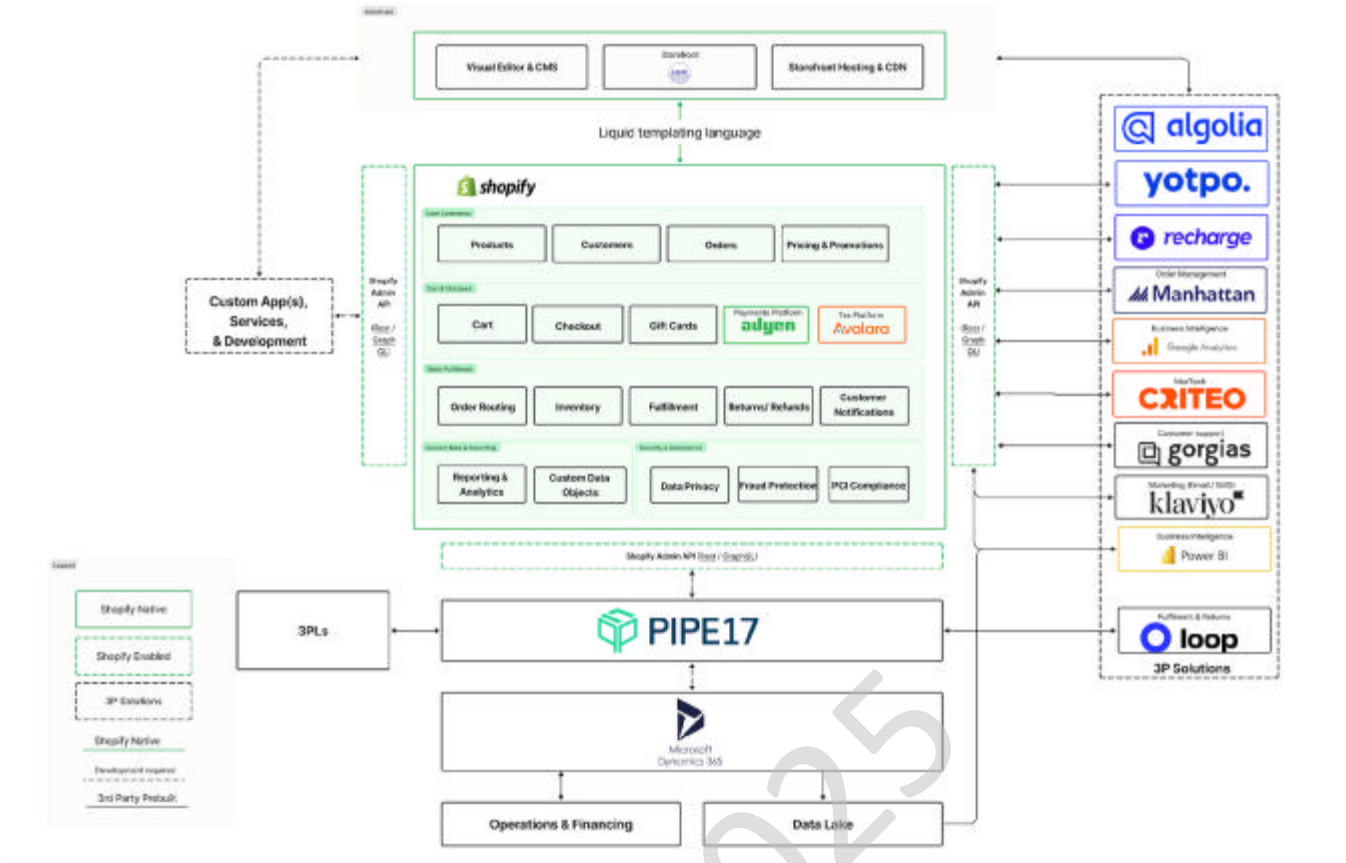


Рисунок 2.5 – Узагальнена схема взаємодії у хмарній архітектурі Shopify

Підсумовуючи проведений аналіз архітектурних рішень та функціональних можливостей провідних систем електронної комерції, можна констатувати наявність системної проблеми у всій галузі. Жодна з розглянутих платформ (Magento, WooCommerce, OpenCart) не забезпечує гарантії незмінності (immutability) даних на рівні архітектури. Всі вони використовують парадигму CRUD (Create, Read, Update, Delete), яка дозволяє безслідно змінювати історичні записи. Покладання на стандартні механізми логування та розмежування прав доступу є неефективним в умовах, коли потенційним зловмисником є адміністратор системи або привілейований інсайдер. Відсутність механізмів ланцюгового хешування або використання технології розподіленого реєстру робить ці системи вразливими до фальсифікації фінансової історії.

Таким чином, ми бачимо, що існуючі на ринку комерційні продукти не задовольняють вимогам щодо гарантованого захисту транзакцій від модифікації. Це

обумовлює актуальність та необхідність розробки власної програмної системи, яка б поєднувала зручність класичного веб-інтерфейсу з надійністю та прозорістю, яку забезпечує інтеграція блокчейн-модуля для фіксації контрольних сум транзакцій

2.2. Обґрунтування вибору засобів для побудови системи та мови програмування

Ефективність функціонування будь-якої програмної системи, особливо в сфері електронної комерції з інтегрованими криптографічними модулями, безпосередньо залежить від раціонального вибору технологічного стеку. Процес вибору інструментальних засобів базується на комплексному аналізі технічного завдання, вимог до швидкодії, рівня безпеки та можливостей подальшого масштабування. Враховуючи специфіку проектованої системи - необхідність взаємодії з розподіленим реєстром (блокчейном) та обробку значної кількості асинхронних запитів від клієнтів - вибір серверної платформи та системи зберігання даних є критичним етапом проектування.

При виборі мови програмування для реалізації серверної логіки нами було виділено наступні ключові критерії оцінки:

– Швидкість обробки I/O операцій (введення-виведення), оскільки система повинна постійно звертатися до бази даних та зовнішніх вузлів мережі блокчейн.

– Наявність та зрілість бібліотек для роботи з криптографією та смарт-контрактами.

– Можливість забезпечення високої конкурентності (concurrency) при великій кількості одночасних підключень.

– Швидкість розробки та підтримки коду.

На етапі вибору мови програмування для серверної частини (Backend) нами було розглянуто три найбільш популярні технології: PHP, Node.js та Python. Мова PHP, незважаючи на широке розповсюдження у сфері веб-розробки, має обмежені можливості для ефективного виконання складних математичних обчислень,

необхідних для майнінгу та хешування блоків, оскільки її архітектура оптимізована переважно під генерацію HTML-сторінок. Платформа Node.js забезпечує високу швидкість обробки асинхронних запитів, проте її однопотокова природа може стати вузьким місцем при виконанні ресурсомістких завдань Proof-of-Work (доказ виконання роботи), блокуючи обробку вхідних запитів від користувачів. Проведений порівняльний аналіз показав, що найбільш оптимальним варіантом для реалізації поставлених задач є мова програмування Python версії 3.10 або вище.

Ми зупинили свій вибір на Python, оскільки він є стандартом де-факто для вирішення алгоритмічних та криптографічних задач у сучасному програмуванні. Ключовим фактором стало наявність потужних вбудованих бібліотек, які дозволяють реалізувати структуру блокчейну без залучення громіздких сторонніх залежностей. Зокрема, бібліотека hashlib надає оптимізовані реалізації алгоритмів хешування, включаючи SHA-256, який є базовим елементом захисту цілісності блоків у нашій системі. Бібліотека json забезпечує швидку серіалізацію та десеріалізацію об'єктів блоків для їх передачі по мережі або запису у базу даних. Крім того, об'єктно-орієнтована природа Python дозволяє зручно описати сутність "Блокчейн" у вигляді класу з методами для перевірки валідності ланцюжка та додавання нових транзакцій. Використання версії Python 3.10+ також обумовлено покращеними механізмами типізації (Type Hinting) та новими синтаксичними конструкціями, що суттєво спрощує написання та відлагодження складних алгоритмів консенсусу, роблячи код більш зрозумілим та безпечним.

Наступним критичним етапом став вибір системи управління базами даних (СУБД). У процесі проектування ми порівнювали реляційні (SQL) та нереляційні (NoSQL) рішення. Хоча NoSQL бази, такі як MongoDB, пропонують гнучкість у роботі з неструктурованими даними, для систем електронної комерції та фінансового обліку критично важливою є сувора структура та цілісність даних. Тому для зберігання метаданих замовлень (ідентифікатор, сума, статус, дані клієнта) нами було обрано реляційну модель та мову запитів SQL. Для етапу розробки та тестування доцільно використовувати SQLite, а для продакшн-середовища -

PostgreSQL. Вибір на користь SQL обумовлений необхідністю забезпечення атомарності транзакцій. Статуси замовлень, такі як "Pending", "Paid" або "Shipped", вимагають чіткої схеми таблиць та дотримання принципів ACID, що гарантує реляційна база. Це дозволяє уникнути дублювання даних та забезпечити узгодженість інформації, що є неможливим при використанні документо-орієнтованих баз даних без додаткових надбудов на рівні коду.

Для реалізації клієнтської частини (Frontend) нами було обрано класичну тріаду технологій: HTML5, CSS3 та JavaScript. Цей вибір забезпечує максимальну сумісність з усіма сучасними браузерами та не вимагає встановлення додаткового програмного забезпечення на стороні клієнта. Мова розмітки HTML5 використовується для створення семантично правильної структури сторінок, а каскадні таблиці стилів CSS3 відповідають за адаптивний дизайн, що дозволяє коректно відображати інтерфейс системи як на моніторах, так і на мобільних пристроях. Особлива роль у нашій архітектурі відводиться мові програмування JavaScript. Вона використовується не лише для візуальних ефектів, а як основний інструмент взаємодії з сервером через API. Саме клієнтські скрипти на JavaScript відповідають за збір даних з форм замовлення, їх валідацію на стороні браузера, пакування у формат JSON-об'єкта та ініціювання асинхронних запитів (fetch) до Python-сервера при натисканні користувачем кнопки підтвердження операції.

Таким чином, проведений аналіз та обґрунтування дозволяють стверджувати, що обраний технологічний стек, який складається з мови Python для криптографічної логіки, SQL-бази даних для надійного зберігання структурованої інформації та JavaScript для динамічної взаємодії з користувачем, є оптимальним. Він забезпечує необхідний баланс між обчислювальною потужністю, необхідною для підтримки функціонування розподіленого реєстру, та надійністю зберігання критично важливих фінансових даних.

2.3. Розгорнута постановка завдання

Основуючись на результатах аналізу недоліків які існують в комерційних платформах та обґрунтуванні вибору інструментального стеку засобів розробки, основною метою магістерської роботи визначається проектування та програмна реалізація спеціалізованої веб-орієнтованої системи електронної комерції з інтегрованим механізмом захисту цілісності даних. Ключовою відмінністю розроблюваного комплексу від стандартних інтернет-магазинів є інтегрування гібридної архітектури зберігання даних, яка поєднує в собі гнучкість реляційної БД для оперативної роботи та надійність послідовного криптографічного реєстру для фіксації історії транзакцій. Поставлене завдання потребує реалізації комплексу взаємопов'язаних функціональних блоків, які повинні забезпечити безперервність бізнес-процесів торгівлі при одночасному дотриманні суворих вимог інформаційної безпеки.

Найпершою вимогою до системи є реалізація повноцінного механізму взаємодії з користувачем, який включає процеси автентифікації, реєстрації та управління сесіями. Відмінно від простих інформаційних сайтів, проектована система повинна буде підтримувати розмежування прав доступу на рівні коду, виділяючи щонайменше дві ролі: звичайний клієнт, який зможе здійснювати покупки та адміністратор, який повинен керувати товарним асортиментом і переглядати замовлення. Дуже важливою вимогою є те, щоб навіть роль адміністратора не надавала технічної можливості безслідно змінювати дані про здійснені транзакції. Система автентифікації повинна бути стійкою до базових веб-атак, таких як підбір паролів або перехоплення даних, це забезпечується використанням алгоритмів хешування паролів та механізмів управління сесійними токенами. Серцевина системи відповідає повному циклу обробки замовлень, від вибору товару до завершення оплати та доставки. Це включає розробку для функціонального управління каталогом товарів з можливістю редагування, покупки кошика зі збереженням вибраних товарів між сесіями та процесу оформлення замовлення. Ключовою вимогою є незмінність даних замовлення. Кожну зміну статусу необхідно призвести до оновлення даних у реляційній базі даних SQLite та

одночасного створення нового запису в захисному реєстрі. Система повинна забезпечувати атомарність цих операцій, щоб запис у базі даних та запис у блоку були частиною однієї транзакції, яка була успішною лише за умови успішного виконання обох дій.

Основна мета полягає в розробці та реалізації алгоритму, що формує ланцюжок блоків за допомогою криптографічного хешування. Ця система буде автоматично генерувати унікальний цифровий підпис для кожної події. Структура кожного блоку даних має включати повний набір метаданих: унікальний ідентифікатор транзакції, точний до секунди часовий штамп, деталі операції, і, що міститься, у попередньому блоці. Включення хешу попереднього обов'язкового запису є обов'язковою умовою для забезпечення нерозривності ланцюжка. Будь-яка спроба модифікації даних у минулому неминуче призведе до зміни хешу відповідного блоку, що, у своєму випадку, порушить цілість хешів усіх наступних блоків, створивши факт втручання математично доведеним. Важливим завданням є створення незалежного модуля верифікації цільності реєстру. Цей компонент, відокремлений від основного процесу продажів, забезпечує можливість проведення аудиту даних у будь-який час. Алгоритм перевірки після наступного зчитування кожного блоку, починаючи з початкового, та обчислення його хеш-суми. Обчислена хеш-сума порівнюється зі збереженою. У випадку виявлення розбіжностей система не тільки повідомляє про помилку, але й надає точну інформацію про блок, у якому виявлено порушення, що значно порушує процес локалізації та ідентифікації моменту несанкціонованого втручання. Крім функціональних завдань, система повинна відповідати нефункціональним критеріям продуктивності та стійкості. Зважаючи на розширене початкове завантаження від генерації криптографічних хешів SHA-256 для кожної транзакції, архітектурна добавка потребує оптимізації для збереження швидкості перегляду інтерфейсу. Це означає застосування ефективних алгоритмів обробки рядків та мінімізацію дискових операцій. Інтерфейс користувача має підтримувати високу чутливість та інтуїтивність, абстрагуючи кінцевого користувача від внутрішніх процесів хешування. Відсутність відчутної

різниці у швидкості роботи разом із звичайними веб-сайтами є критичною умовою для рішення комерційної життєздатності.

У процесі постановки завдання слід врахувати вимоги до масштабності та портативності розробленого рішення. Код програми має бути організований таким чином, щоб у перспективі була можливість заміни локального файлового сховища блокчейну на розподілену мережеву базу даних або інтеграції із зовнішніми інтерфейсами прикладного програмування API без необхідності повної реорганізації ядра системи. Застосування стандартизованих протоколів комунікації та демонструє жорстку взаємодію від апаратної конфігурації конкретного сервера дозволяє розгортати створену систему як на високопродуктивних дислокованих серверах, так і в хмарних інфраструктурах або на локальних машинах для проведення тестування. Таким чином завдання полягає у створенні програмного рішення для електронної комерції, яке усуває проблему довіри шляхом забезпечення криптографічної незмінності даних. Буде продемонстрована практична застосовність та доцільність використання криптографічних механізмів, аналогічних тим, що застосовуються в криптовалютах, в рамках класичної архітектури веб-додатків. Така інтеграція дозволить досягти вищого рівня захисту від інсайдерських загроз та фальсифікацій порівняно з комерційними CMS-платформами. Реалізація цього завдання передбачає комплексний підхід, що включає розробку зручного користувацького інтерфейсу, надійного серверного компонента на Python та побудову блокчейн-реєстру з використанням суворої математичної логіки.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1. Опис функціонування схеми

Розробка сучасної системи електронної комерції з високими вимогами до безпеки та цілісності даних вимагає застосування комплексного архітектурного підходу, який буде базуватися на принципах модульності, використанні багаторівневої моделі та чіткому розмежуванні зон відповідальності. Система яка проектується – це гібридне веб-орієнтоване рішення, що поєднує в собі стандартні патерни проектування веб-додатків з елементами технології розподільного реєстру. Ось така архітектурна комбінація допомагає зберегти високу швидкість обробку запитів, що притаманна централізованим системам, одночасно забезпечуючи критично важливі властивості незмінності історії транзакцій. Логіку цього програмного комплексу слід розглядати з боку трирівневої архітектури, що включає в себе рівень авторизації, який відповідає за взаємодію з кінцевим користувачем, рівень даних, що забезпечує двоготривале зберігання інформації у незалежних форматах та рівень бізнес-логіки, в якому відбувається основні обчислювальні процеси. Взаємозв'язок між цими компонентами визначає загальну працездатність та надійність системи. Функціонування розробленої системи електронної комерції базується на принципах інтерактивної взаємодії користувача з веб-інтерфейсом та автоматизованої фонової обробки транзакцій на сервері. Логіку роботи системи можна розділити на два основні режими: режим клієнтської взаємодії (Frontend-процеси) та режим серверної обробки даних (Backend-процеси). З точки зору користувача, функціонування системи починається з процедури реєстрації або автентифікації. Користувач вводить облікові дані, які передаються на сервер захищеним каналом. Після успішного входу система надає доступ до каталогу товарів та особистого кабінету. Основний бізнес-процес полягає у виборі товару та оформленні замовлення. У момент натискання кнопки "Купити" ініціюється

ланцюжок подій: веб-додаток фіксує параметри запиту (ID товару, поточну ціну, ID користувача) і відправляє їх на обробку. Серверна частина системи функціонує як диспетчер даних. Отримавши запит на створення замовлення, система виконує подвійну операцію. По-перше, створюється стандартний запис у реляційній базі даних, який відображає статус замовлення для відображення в інтерфейсі (наприклад, "Оплачено"). По-друге, паралельно запускається модуль блокчейн-фіксації. Цей модуль бере ключові параметри транзакції, додає до них часову мітку та хеш попереднього запису в системі, і на основі цих даних генерує новий криптографічний блок.

Окремо варто наголосити на специфіці реалізації зворотного зв'язку між підсистемою блокчейну та основною реляційною базою даних, що є визначальною рисою запропонованої архітектури. Процес побудовано таким чином, що одразу після успішної генерації хеш-суми нового блоку криптографічним модулем, цей унікальний ідентифікатор (фактично -цифровий підпис транзакції) повертається до контролера додатку. На цьому етапі ініціюється SQL-транзакція типу UPDATE, яка фіксує отриманий хеш у спеціально зарезервованому полі таблиці замовлень (tx_hash). Цей крок є фундаментальним для забезпечення цілісності, адже він формує жорстку, математично обґрунтовану прив'язку між динамічним записом у базі даних (який потенційно вразливий до редагування через SQL-інструменти) та статичним записом у захищеному файловому реєстрі. Фактично, система переходить на режим подвійної верифікації, де база даних відповідає за оперативність пошуку та фільтрації контенту для користувача, тоді як блокчейн слугує гарантом валідності цієї інформації. Не менш важливим аспектом, який визначає стабільність роботи комплексу, виступає підсистема керування сесіями користувачів та реалізація політики розмежування прав доступу (Role-Based Access Control -RBAC). Беручи до уваги, що протокол HTTP, який лежить в основі веб-взаємодії, є протоколом без збереження стану (stateless), для ідентифікації клієнта між окремими запитами застосовано механізм захищених сесійних файлів cookie. У момент проходження процедури автентифікації сервер генерує унікальний сесійний

токен, підписаний секретним ключем додатку, що унеможливлює його фальсифікацію. Цей токен інтегрується у заголовки кожного наступного запиту від браузера клієнта. На стороні сервера розгорнуто проміжне програмне забезпечення (middleware), завданням якого є перехоплення запитів, дешифрування токена та визначення поточної ролі користувача (адміністратор чи клієнт). Базуючись на визначеній ролі, система динамічно генерує HTML-відповідь, приховуючи або, навпаки, візуалізуючи специфічні елементи інтерфейсу, такі як панель адміністрування товарів чи модуль аудиту блокчейну.

Втім, захисний периметр системи не обмежується виключно процедурами хешування транзакцій. Суттєвим компонентом загальної стратегії безпеки є попередня валідація вхідних потоків даних безпосередньо в момент їх надходження від клієнта. Контролер системи забезпечує сувору типізацію та санітизацію всіх параметрів, що передаються через веб-форми. Це передбачає не лише перевірку типів даних (наприклад, вартість товару повинна бути числом з плаваючою комою, а ідентифікатор -цілим числом), але й контроль допустимого діапазону значень та очистку від спецсимволів, які потенційно можуть бути використані для реалізації атак класу SQL Injection або Cross-Site Scripting (XSS). Дані передаються на рівень бізнес-логіки для формування транзакції виключно після успішного проходження всіх етапів фільтрації. Такий підхід дозволяє відсіяти потенційно шкідливі запити ще до того, як вони почнуть взаємодіяти з базою даних або реєстром блокчейну. Гарантом надійності функціонування розробленого програмного продукту виступає модуль аудиту цілісності (Audit Watchdog). Функціонал цього компонента передбачає роботу у двох режимах: синхронному (ініціюється запитом користувача при відкритті профілю) та асинхронному (як фоновий сервіс). Алгоритмічна база роботи модуля полягає у повному перерахунку криптографічних залежностей. Процес аудиту послідовно зчитує блоки з файлового реєстру, починаючи з Genesis-блоку. Для кожного елемента виконується рекурсивне обчислення хеш-суми за алгоритмом SHA-256, базуючись на збережених у ньому даних. Результат порівнюється з хешем, що зафіксований у заголовку наступного блоку. Виявлення

розбіжності навіть в одному біті призводить до зупинки верифікації та зміни статусу системи на "Компрометовано". Крім перевірки внутрішньої структури ланцюжка, модуль виконує перехресну звірку з базою даних SQL, перевіряючи наявність відповідного хешу транзакції в таблиці замовлень. Це дозволяє детектувати випадки прямої модифікації даних у БД в обхід програмного інтерфейсу.

Також слід зазначити механізми обробки виключних ситуацій та забезпечення відмовостійкості. Оскільки архітектура передбачає запис у два різнорідні сховища (файл блокчейну та файл SQLite), існує теоретична ймовірність розсинхронізації через апаратні збої. Для нівелювання цього ризику реалізовано принцип атомарності операцій. Логіка збереження налаштована так, що фіксація даних у БД вважається завершеною лише після підтвердження запису в блокчейн. При виникненні помилки на етапі криптографічної обробки або доступу до файлу реєстру, система автоматично запускає процедуру відкату (Rollback), повертаючи базу даних до попереднього стабільного стану.

З технічної точки зору, середовище виконання базується на WSGI-сумісному серверному додатку, що дає змогу ефективно обробляти конкурентні запити. Хоча природа блокчейну вимагає послідовного запису, веб-сервер використовує механізми короткострокових файлових блокувань (file locking) на час генерації блоку. Це дозволяє обслуговувати велику кількість користувачів у режимі читання ("read operations"), блокуючи ресурс лише на мілісекунди для операцій запису ("write operations"). Насамкінець, система спроектована з урахуванням можливостей масштабування. Модульна архітектура дозволяє у перспективі винести модуль блокчейну в окремий мікросервіс, взаємодія з яким відбуватиметься через REST API. Це дозволить оптимізувати розподіл навантаження на процесор (задіяний у хешуванні) та оперативну пам'ять, що є необхідною умовою для трансформації пілотного проекту у високонавантажену промислову систему.

3.2 Розробка структурної схеми

Етап у якому буде виконуватися розробка структурної схеми є визначальним у процесі проектування, тому що саме він буде фіксувати логічну ієрархію компонентів та встановлювати маршрути потоків даних які перетікають по модулям. Якщо враховувати специфічні тонкощі до забезпечення незмінності транзакцій, то кращим рішенням було обрати гібридну архітектурну модель. Яка являє собою синтез класичного клієнт-серверного підходу, характерного для веб-застосунків і елементів розподіленого реєстру, що виконує роль захищеного архіву. Створена схема базується на модульному принципі, у якому кожен сегмент системи відповідає за незначне коло задач, зо узгоджується патерном єдиної відповідальності. Центральним вузлом структури виступає серверний контролер обробки запитів, що реалізований на базі Flask, який виконує функцію направляючого: він приймає вхідні сигнали від інтерфейсів користувача, проводить їх первинну фільтрацію та розподіляє навантаження між підсистемами зберігання. Ключова особливість архітектурної особливості є диференціація рівня даних на два ізольовані блоки, що дозволяє отримати певний баланс між швидкодією та безпекою.

– Оперативний контур побудований на базі SQLite. Відповідає за зберігання змінної інформації, до якої необхідний миттєвий доступ для читання та редагування

– Захищений контур структурно виділений модуль, який працює в режимі лише додавання. Його задача – акумулювати хешовані транзакції. Важливо зазначити, що структурно такий контур не має прямих інтерфейсів для зовнішнього доступу; комунікація з ним можлива виключно через внутрішні захищені канали ядра системи.

Сторона клієнта реалізована за концепцією тонкого клієнта. Де Веб-бразуер юзера не робить ніяких критичних обчислень, а слугує лише для візуалізації отриманих HTML-сторінок та відправки форм. Зв’язок між клієнтом та ядром системи забезпечується через протокол HTTP/HTTPS, що робить систему

незалежною від апаратної платформи користувача. Графічна візуалізація, що демонструє ієрархію модулів та вектори взаємодії між ними, показана на рисунку 3.1.

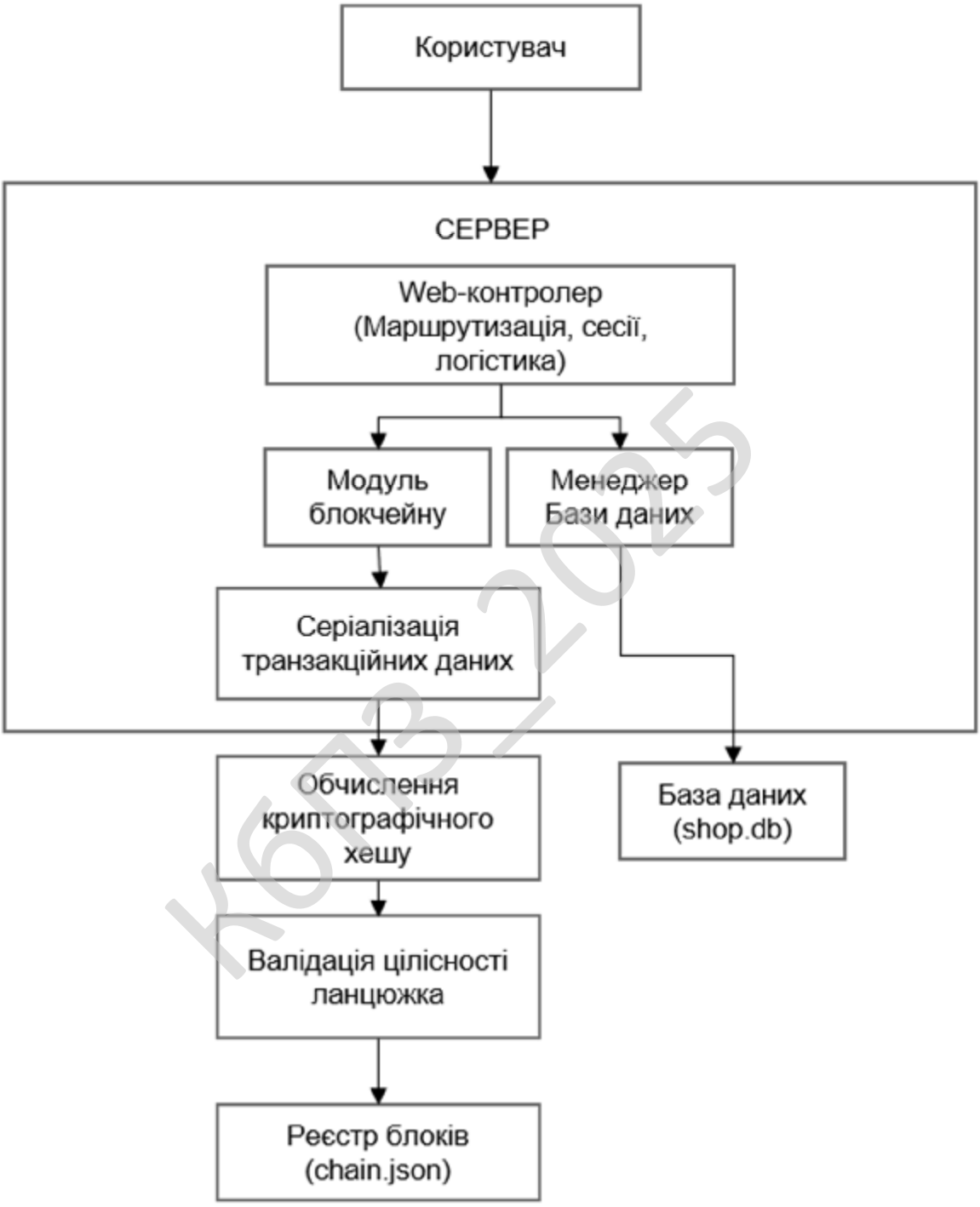


Рисунок 3.1 – Структурна схема системи

Схема що представлена вище демонструє чітке відокремлення логіки

відображення від бізнес-логіки та механізмів криптографічного захисту. Саме таке архітектурне рішення створить можливості для безпечного горизонтального масштабування системи в майбутньому без необхідності валідації ядра.

3.3 Розробка функціональної схеми

Розробка функціональної схеми ПЗ являє собою детальне моделювання інформаційних потоків і структур даних, що в свою чергу забезпечують створення бізнес-процесів системи. Якщо врахувати гібридний характер архітектури, який поєднує в собі оперативну обробку замовлень із довгостроковим захищеним зберіганням історії, функціональну схема була розділена на два взаємозв'язані рівні: ревень даних розподіленого реєстру і рівень реляційних даних. Основою першого рівня проектування є проектування БД, яка має містити в собі всі вимоги 3NF для уникнення надлишковості інформації. Для реалізації підсистеми електронної комерції було розроблено схему, яка базується на взаємодії трьох головних сутностей:

Сутність Користувачі.

Відповідає за функціонал автентифікації та розмежування прав доступа. Ця сутність зберігає ідентифікаційні дані та хеші паролів, що забезпечує безпеку входу до системи.

Сутність Товари.

Ця сутність виконує функцію цифрового каталогу та зберігаю актуальну інформацію про номенклатуру, цінову політику та складські залишки.

Сутність Замовлення.

Це центральний елемент функціональної схеми, який комбінує інформацію про покупця і товар.

Одною з найважливіших особливостей цієї схеми є інтеграція поля tx_hash напряму в структуру таблиці замовлень. Це поле не бере участі в класичних реляційних зв'язках, однак виконує функцію моста до підсистеми безпеки. Цей

процес зберігає унікальний цифровий відбиток, отриманий від блокчейн-модуля, що дозволяє системі в будь-який момент верифікувати автентичність запису. Графічне зображення зв'язків між сутностями та атрибутивний склад таблиць показано на рисунку 3.2

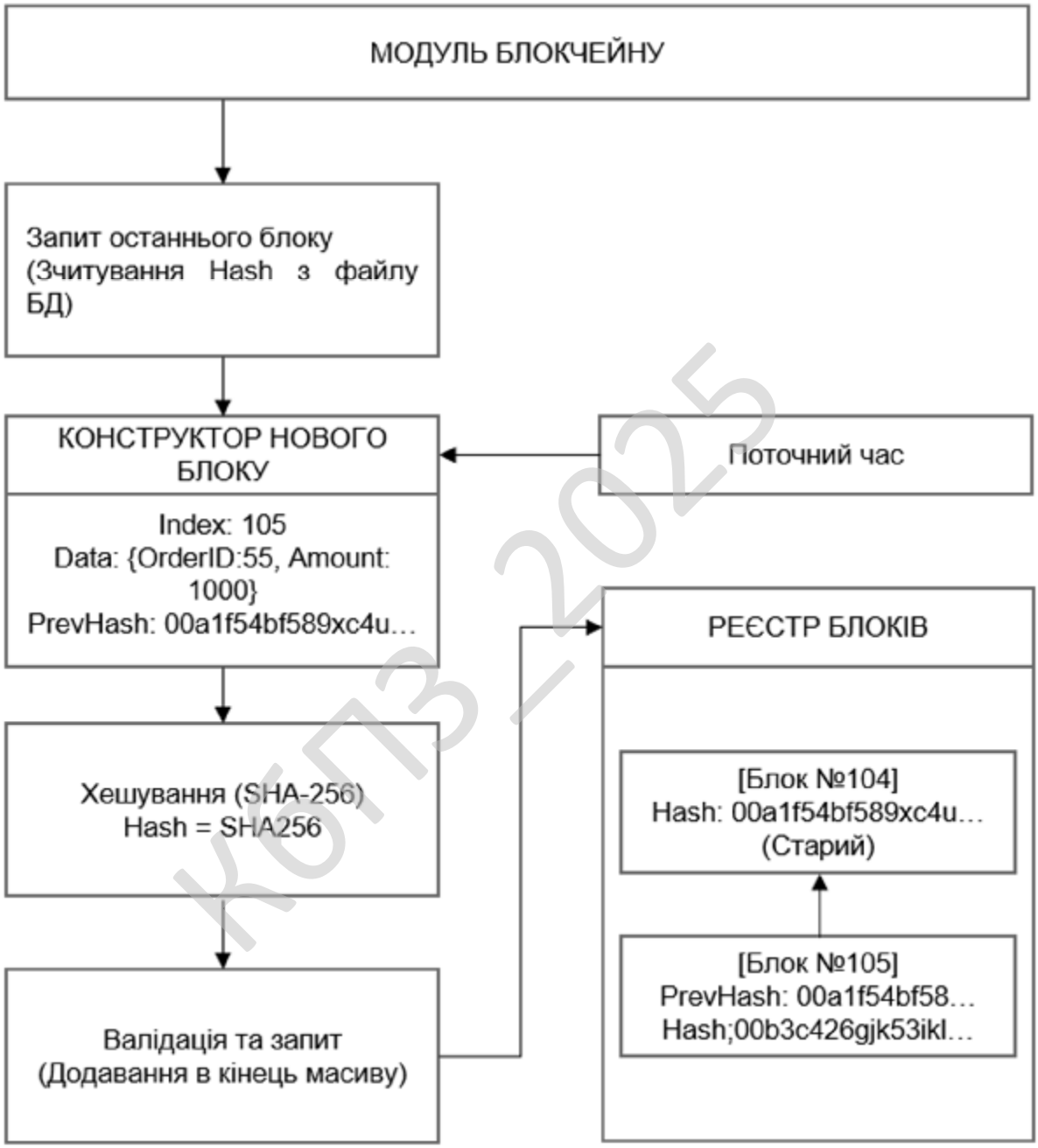


Рисунок 3.2 – Функціональна схема системи

Другий рівень функціональної схеми - це структура криптографічного

реєстру, на відміну від БД, де зв'язки сформовані на основі індексів, тут функціональна залежність вибудовується на основі хешів. Кожен блок у такій схемі представляється як контейнер, який містить дані про транзакцію та службову інформацію. Функціональна логіка побудови ланцюга передбачає те, що кожен свіжий елемент містить в собі цифровий підпис попереднього елемента, таким чином створюється лінійна, хронологічно впорядкована структура, де неможливо вилучити або змінити проміжний елемент без руйнування цілісності всього масиву даних. Конкретно ця особливість, відома в криптографії як лавинний ефект, він є фундаментальним для розробки функціональної моделі. Навіть найменша зміна вхідних даних призводить до повної зміни хеш-значення відповідного блоку. З точку зору функціональності це механізм автоматичного захисту, тому шо змінений хеш історичного блоку більше не відповідатиме попередньому значенню, яке було записано в заголовку наступного блоку, після цього зв’язок розривається і це автоматично робить недійсним модифікований запис та все наступні що йдуть після нього. Таким чином, спроба локальної фіксації миттєво переростає в глобальну помилку послідовності всього реєстру.

Схематичне зображення принципу формування ланцюга блоків наведено на рисунку 3.3

Рисунок 3.3 – Схема формування блоків

Поєднання наведених вище схем дозволяє реалізувати систему, шо буде функціонально забезпечувати швидку обробку клієнтських запросів та гарантовану незмінність фінансової звітності. Більше того, описана модель ефективно вирішує

одну с з головних проблем інтеграції криптографічних технологій у реальний бізнес – це проблема затримок обробки даних. У звичайних знайомих нам децентралізованих системах процес запису блоку може займати значний час, що є недопустимим для динамічного середовища електронної комерції, де клієнт очікує миттєвого відгуку від інтерфейсу. Ця схема обходить таке обмеження шляхом асинхронного розділення потоків запису. Функціонал побудований так, що взаємодія користувача відбувається переважно з високошвидкісним реляційним сегментом, що оптимізований для процесу читання та пошуку.

Ось так розроблена схема є досить збалансованим архітектурним рішенням, що в свою чергу не вимагає компромісів між зручністю використання та вимогами ІБ, створюючи надійний фундамент для побудови довіреного середовища онлайн-торгівлі.

3.4 Розробка діаграми процесів

Створення діаграми взаємодії процесів є останнім етап моделювання, що допомагає візуалізувати динаміку обміну даними між функціональними компонентами системи, відмінно від структурної схеми, яка демонструє ієрархію модулів, така діаграма тримає фокус на станах системи і логікі переходів між ними під час обробки транзакцій. Головним елементом є ядро системи – ПЗ обробки захищених транзакцій. Саме цей вузол є диспетчером, що координує роботу всіх допоміжних підсистем, логіка взаємодії процесів створена радіальним принципом, де головний процес ініціює звернення до периферійних модулів, що залежить від етапу життєвого циклу замовлення.

Процес роботи починається з ініціалізації сесії клієнта. Після успішного входу стає активним процес валідації даних, який фільтрує некоректні запити ще до того як вони потраплять на обробку. Найголовнішу роль у забезпеченні безпеки велику роль грає група криптографічних процесів, процес генерації структури блоку поєднує дані про замовлення та передає їх на хешування, в той час результат

обчислень йде до процесу синхронізації БД та Блокчейну, що забезпечує атомарність запису: дані водночас фіксуються в двох процесах – запис у реляційну БД та Архівування у JSON. Окремий контур взаємодії створює процес аудиту цілісності, що працює паралельно з головним потоком обробки. Він здійснює періодичне опитування сховищ даних для виявлення розбіжностей.

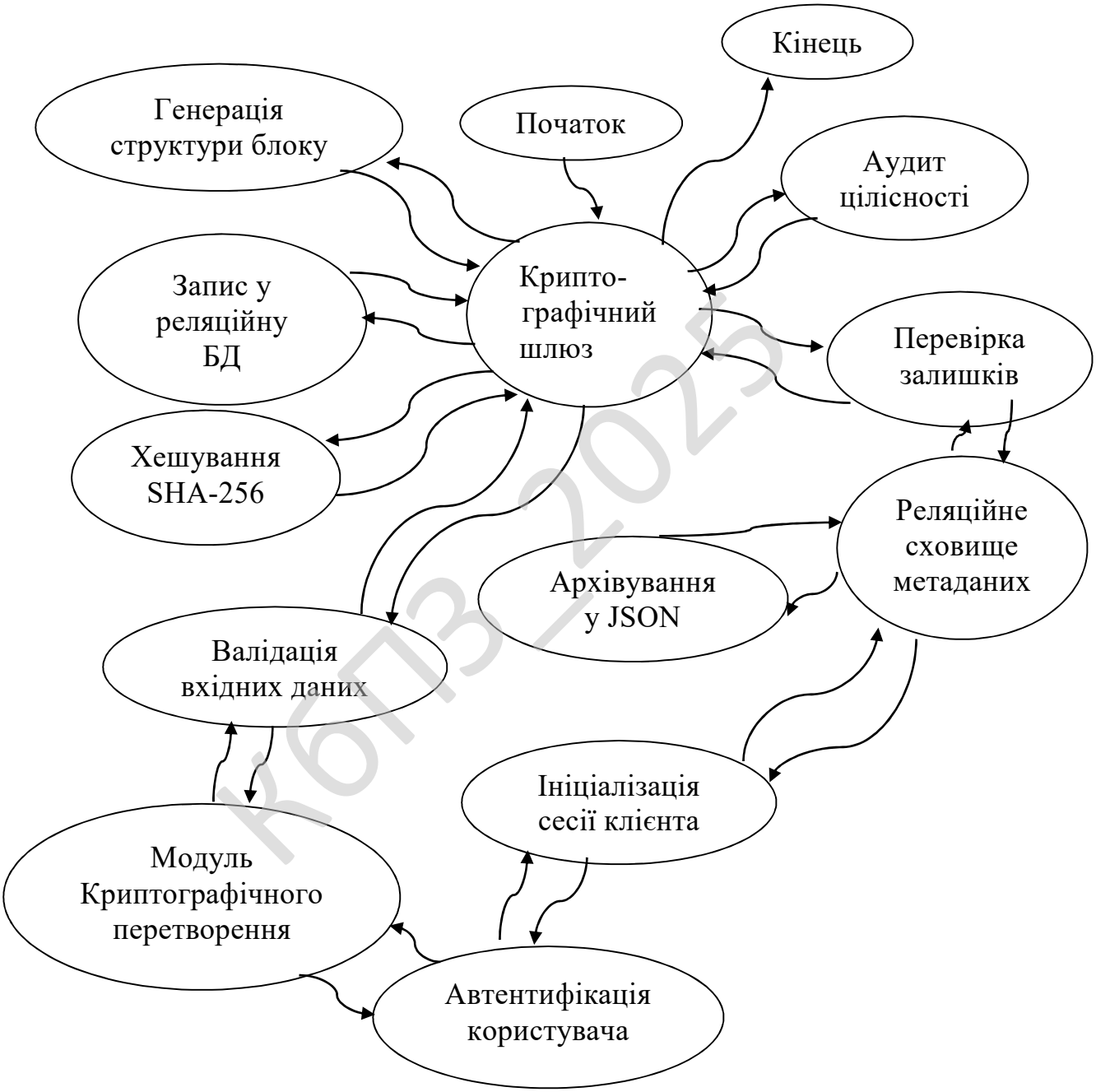


Рисунок 3.4 – Діаграма взаємодії процесів

Показана діаграма демонструє замкнутість циклів обробки даних і наявність перевірок на кожному з етапів роботи системи, що в свою чергу робить неможливим

непомітну модифікацію фінансової інформації.

У цьому розділі було виконано комплексне проєктування архітектурних рішень для створення веб-орієнтованої системи управління замовленнями з інтегрованими механізмами блокчейну-захисту. В ході розробки структурної схеми була обрана та реалізована гібридна архітектурна модель, що поєднує класичну трирівневу взаємодію з елементами технології розподіленого реєстру. Цей підхід допоміг вирішити критичну проблему продуктивності, притаманну блокчейн-системам, через розділення контурів зберігання даних на реляційну БД та файловий реєстр, такий підхід забезпечив миттєву реакцію інтерфейсу на дії користувача при одночасному гарантуванні незмінності історії фінансових транзакцій.

Створена схема системи базується на використанні алгоритму хешування SHA-256 для формування ланцюга блоків. Реалізований механізм подвійного запису та перехресної верифікації створює математично описаний захист від несанкціонованої зміни даних. Було доведено, що спроба підробки будь-якого запису в БД призведе до розриву криптографічного ланцюга, що автоматично детектиться системою аудиту. За допомогою діаграми даних були формалізовані алгоритми обробки транзакції та візуалізовано життєвий цикл замовлення від моменту ініціалізації клієнтом до фінальної архівації. Модель взаємодії процесів, що була створена передбачає в собі наявність механізмів влаідації даних, синхронізації сховищ та обробки виключних ситуацій, що мінімізує ризики крашу системи. Ось таким чином, запропоновані у розділі проєктні рішення повністю відповідають вимогам щодо безпеки, швидкості роботи та цілісності даних, та створюють надійний фундамент для програмної реалізації захищеної системи електронної комерції.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Розробка блок-схем та опис алгоритмів функціонування системи

Реалізація магістерської дипломної роботи являє собою створення складного програмного комплексу, який поєднує в собі інтерактивний веб-інтерфейс, серверну бізнес-логіку обробки транзакцій та спеціалізовані криптографічні алгоритми які в свою чергу забезпечують цілісність даних. Період програмної реалізації є критично важливим, тому що саме на цьому етапі абстрактні архітектурні моделі та схеми, створенні у попередніх розділах, трансформуються у працездатний програмний продукт. Для забезпечення надійності, стійкості до відмови, швидкодії та можливості подальшого масштабування системи був проведений глибокий порівняльний аналіз сучасних засобів розробки та зібрано оптимальний технологічний стек.

Вибір засобів реалізації базувався на наступних критеріях:

– Швидкість розробки. Можливість швидкого прототипування та внесення змін у кодову базу.

– Наявність спеціалізованих бібліотек. Підтримка криптографічних примітивів.

– Підтримка веб-технологій. Наявність розвинених фреймворків для обробки HTTP-запитів.

– Кросплатформеність. Можливість запуску системи на різних ОС без необхідності адаптації коду.

В якості стандартного інструменту розробки було обрано мову програмування Python. Цей вибір не випадковий і зумовлений рядом технічних та економічних факторів, які роблять цю мову програмування стандартом у галузі

розробки серверних додатків та систем аналізу даних. Мова Python загального призначення та вона підтримує мультипарадигмальний підхід. Це значить, що у розробника є можливість використовувати процедурний стиль, ООП, та функціональний стиль. У контексті даної роботи ООП став ключовим, він дозволив реалізувати сутність користувача, блоку, замовлення та ланцюжка у вигляді програмних класів, що містять у собі дані та методи їх обробки, цей підхід підвищує читабельність коду то спрощує його підтримку.

Критично важливим аргументом на користь Python стала його стандартна бібліотека, що виділяється своєю повнотою, ще для реалізації ядра блокчейн-системи був використаний вбудований модуль hashlib, він є інтерфейсом для надійних реалізацій алгоритмів хешування на мові C. Це створює умови для високої продуктивності обчислень, що наближається до низькорівневих мов, залишаючи при цьому зручність написання коду на високому рівні. Для порівняння, реалізація аналогічного функціоналу на мовах C++ чи Java просила б написання значно більшого обсягу коду, тим самим це ускладнило б процес розробки в рамках обмеженого часу.

Ще одним важливим компонентом є наявність екосистеми бібліотек Python, які надають широкі можливості для аналізу та моделювання даних. Спеціалізовані інструменти, включаючи pandas та numpy для обробки та аналізу даних, а також matplotlib та seaborn для візуалізації, дозволяють студентам ефективно обробляти великі обсяги фінансової інформації та виявляти глибокі тенденції. Це розвиває не лише технічні навички, але й критичне мислення та здатність проводити інноваційний аналіз. Велика та активна спільнота користувачів забезпечує швидке вирішення проблем під час навчання, що особливо цінно в освітньому середовищі, оскільки знижує ризик ізоляції студентів, розвиває навички професійного спілкування та сприяє обміну досвідом. Крім того, інтеграція Python з іншими технологіями та багатоплатформні можливості роблять його універсальним інструментом для розробки складних фінансових додатків та аналітичних систем, що є важливим практичним аспектом для фінансових установ з різноманітними IT-

екосистемами. Міждисциплінарність є основною педагогічною умовою успішного впровадження Python у підготовку фінансових та технологічних фахівців. Ця практика передбачає поєднання знань та методів у програмуванні, фінансах, економіці та статистиці, що допомагає студентам визначати зв'язки між теоретичними концепціями та практичними інструментами. Така інтеграція досягається завдяки одночасному розвитку як технічних, так і аналітичних компетенцій. Python сприяє впровадженню міждисциплінарної практики, дозволяючи студентам отримати глибокі знання з фінансів, економіки та інформатики, необхідні для успішної роботи в сучасних фінансових компаніях. Ця мова програмування підтримує автоматизацію фінансових процесів, що актуально для розвитку навичок застосування цифрових технологій у реальних професійних ситуаціях та створення інноваційних рішень. Ця здатність автоматизувати та аналізувати дані, у поєднанні зі здатністю вирішувати багатокомпонентні завдання у складних та динамічних фінансових галузях, робить Python основним інструментом для підготовки висококваліфікованих фахівців, які можуть швидко адаптуватися до динаміки фінансово-технічної сфери. Таким чином, Python, як універсальний інструмент, поєднує різні аспекти навчального процесу: від основ програмування до модулів з фінансової аналітики та автоматизації бізнес-процесів. Така практика вимагає від студентів і викладачів глибокого розуміння не лише предмета, а й навичок синтезу знань з різних дисциплін у межах єдиної освітньої траєкторії, що сприяє розвитку комплексного професійного мислення.

Також варто зазначити динамічну типізацію Python. Незважаючи на те, що в великих корпоративних системах це може вважатися недоліком, у рамках наукових досліджень це є великою перевагою. Динамічна типізація дозволяє легко працювати з різними структурами даних, наприклад, словниками та списками, які часто використовуються для створення JSON-структури блоку транзакції. Відсутність необхідності вказувати типи змінних заздалегідь дозволяє більше уваги приділяти самій суті задачі, а не формальним правилам синтаксису.

Для упорядкування взаємодії між користувачем та серверною частиною

системи було використано архітектурний патерн MVT (Model-View-Template), який є адаптацією класичного патерну MVC (Model-View-Controller) для вебу. Ця архітектура передбачає яскраве розділення областей відповідальності:

– Model: Компонент, що відповідає за оперування даними. У збудованій системі функцію моделі виконують класи бази даних (SQLAlchemy Models) і логіка блокчейну.

– View: Компонент, що містить основну бізнес-логіку. Він отримує вхідні запити від клієнта, запитує дані у моделі й передає їх у шаблон.

– Template: Компонент, який відповідає за презентацію інформації (HTML-код).

Для реалізації такої архітектури було обрано мікрофреймворк Flask. У відміну від “монолітних” фреймворків типу Django, що включають у себе неїмовірно велику кількість непотрібних модулів та диктують розробнику жорстку структуру проєкту, Flask приділений принципу мінімалізму. Flask надає користувачеві тільки базовий набір засобів: маршрутизацію URL, обробку HTTP-запитів і контекст виконання. Всі інші частини додавати у вигляді розширень при необхідності. Така архітектура доволі добре підходить для розробки гібридних систем. Оскільки використовується своєрідний метод збереження даних, а саме їх ініціалізація запису в реєстри SQL та на файл, то використання легкого фреймворку відкрило можливості створення власної системи обробки транзакцій, не маючи справи із вбудованими обмеженнями. Крім цього, Flask підтримує серверний інтерфейс WSGI, який можна використовувати та будь-якому веб-сервері, такому як Nginx або Apache. Найважливішою частиною реалізації системи електронної комерції є вибір надійної СУБД, враховуючи гібридну архітектуру проєкту, що передбачає наявність декількох варіантів зберігання, було прийнято рішення використати реляційну модель для зберігання даних про замовлення та користувачів. В якості СУБД було обрано SQLite – це компактна, вбудована реляційна БД, що вбудовується напряму в програмне забезпечення, вибір цієї БД був обумовлений наступними технічними характеристиками:

– Архітектура без сервера: Відмінно від клієнт-серверних систем, SQLite не вимагає окремого процесу сервера, що потрібно окремо встановлювати, адмініструвати та налаштовувати. Двигун БД працює як складова частина програми, шо спрощує розгортання системи на тестових стендах.

– Цілісність транзакцій: БД гарантує відповідність всім потрібним стандартам, це значить, що у випадку аварійного завершення роботи програми або збою, цілісність даних не буде порушена, що є найважливішим аспектом для фінансових систем.

– Файлова організація: Вся БД міститься в єдиному кросплатформеному файлі на диску – це ідеально співіснує з концепцією блокчейну, що також є файловим реєстром. Така структура дозволяє легко виконувати резервне копіювання.

Для взаємодії програмного коду з БД було використано технологію ORM, реалізовану за допомогою бібліотеки SQLAlchemy, використання ORM дозволило відійти від написання сирих запитів на базі SQL і спокійно працювати з записами БД як з класами Python, це надало такі переваги:

– Переносимість: При необхідності масштабування проєкту перехід на кращу та потужнішу СУБД вимагатиме зміни лише одного рядка конфігурації, без необхідності переписування коду запитів.

– Безпека: Бібліотека автоматично екранує вхідні дані, що робить систему невразливою до найпоширенішого типу атак.

Через те що система орієнтована на кінцевого користувача, дуже важливим етапом реалізації було створення інтуїтивно зрозумілого графічного інтерфейсу. Реалізація клієнтської частини було створена на використанні сучасних стандартів веб-розробки: HTML, CSS3 та мови шаблонів Jinja2, а за для забезпечення адаптивності інтерфейсу був використаний фреймворк Bootstrap 5. Використання цього фреймворку дозволило:

– Забезпечити єдиний стиль елементів управління.

– Реалізувати інтерактивні елементи.

– Використати готову сітку для розташування елементів на сторінці.

Основним класом, що реалізує логіку захищеного реєстру, є клас Blockchain. Його алгоритм ініціалізації передбачає створення первинного блоку (Genesis Block) з фіксованими параметрами, що слугує точкою відліку для всього ланцюга. Алгоритм додавання нової транзакції (замовлення або зміни статусу) реалізовано наступним чином:

Система отримує вхідні дані про транзакцію (ID замовлення, тип операції, timestamp, ID відповідальної особи).

– Дані проходять валідацію на коректність типів та повноту.

– Виконується пошук хешу останнього блоку в ланцюзі (last_hash).

– Формується структура нового блоку, що включає вхідні дані та last_hash.

– Обчислюється хеш-сума нового блоку за алгоритмом SHA-256.

– Блок записується у JSON-реєстр, а хеш транзакції дублюється у реляційну базу даних для швидкого пошуку.

Для візуалізації логіки роботи програмного забезпечення розроблено блок-схему алгоритму додавання захищеної транзакції.

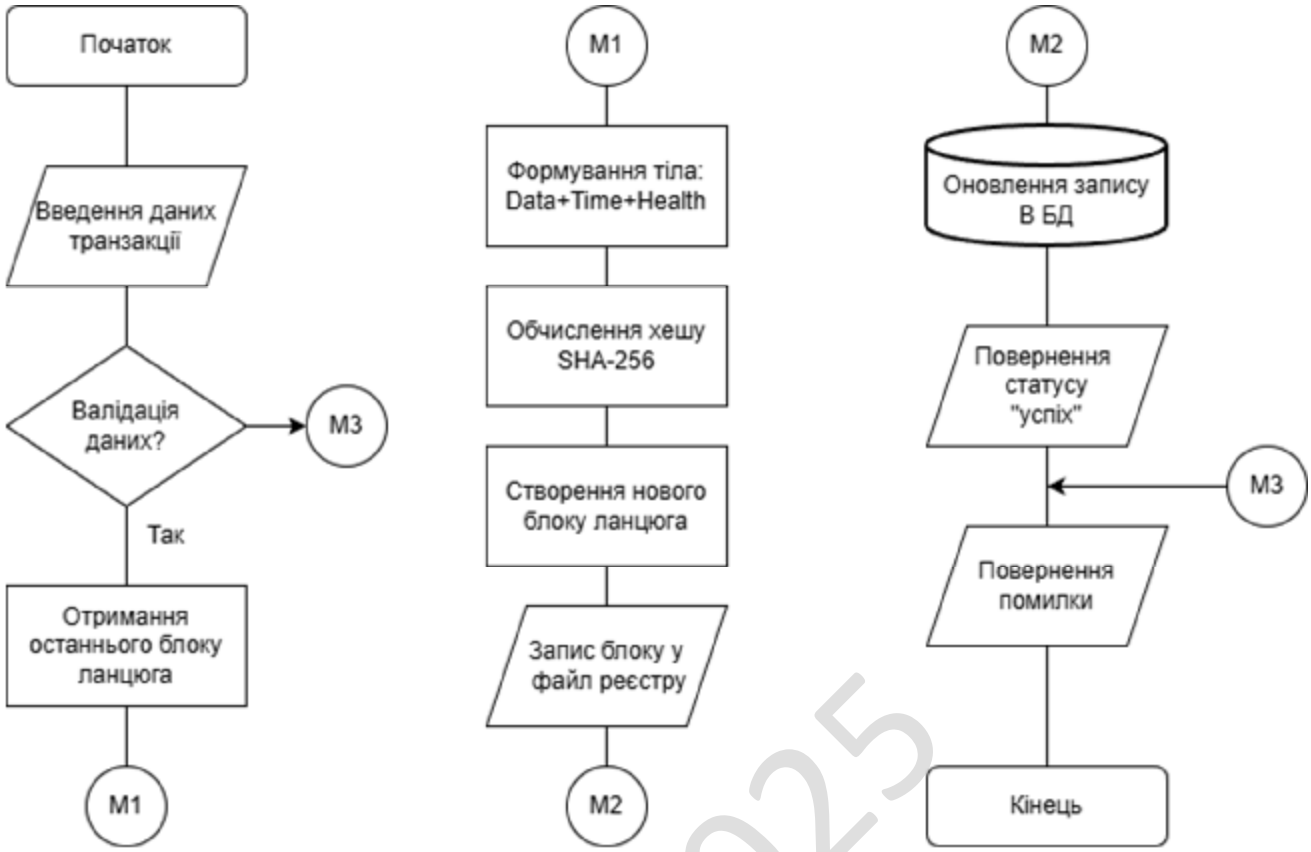


Рисунок 4.1 – Блок-схема алгоритму додавання захищеної транзакції

Другим критично важливим алгоритмом є процедура верифікації цілісності даних. Оскільки система призначена для виявлення несанкціонованих змін, алгоритм перевірки повинен проходити весь ланцюг і перераховувати хеші. Реалізація алгоритму аудиту (функція is_chain_valid) виконує ітеративний обхід списку блоків. Для кожного блоку з індексом *i* виконується перевірка умови:

$$Hash(Block_{i-1}) == Block_i.previous\_hash$$

Якщо умова не виконується, алгоритм перериває роботу і повертає індекс скомпрометованого блоку. Функціонування розробленої системи базується на подійно-орієнтованій моделі (Event-driven architecture). Будь-яка зміна стану системи, така як створення замовлення, зміна його статусу або редагування товарної позиції, розглядається як атомарна подія (транзакція), що підлягає валідації, серіалізації та фіксації у розподіленому реєстрі.

Вибір послідовності дій Валідація, Формування блоку, Хешування, Персистентне

збереження - обумовлений необхідністю мінімізації обчислювальних витрат. Криптографічні перетворення, зокрема хешування, є найбільш ресурсоємними операціями, тому їх виконання доцільне лише після успішного проходження вхідними даними всіх етапів логічного контролю. Математично вхідну транзакцію T можна представити як вектор параметрів у багатовимірному просторі станів системи:

$$T = \{id, u, P, A, t, s\}$$

Процес фіксації транзакції є центральним алгоритмом системи, що забезпечує перехід системи з поточного стану S_i у новий стан S_{i+1} із гарантією збереження цілісності історичних даних. Розглянемо детальний опис етапів цього алгоритму.

Етап 1: Валідація вхідних даних.

На цьому етапі відбувається перевірка належності компонентів вектора T допустимим областям значень. Функція валідації $V(T)$ визначається як предикат:

$$V(T) = \begin{cases} 1 \\ 0 \end{cases}$$

У разі, якщо $V(T) = 0$, алгоритм перериває виконання, генеруючи виключну ситуацію типу ValidationError, що запобігає потраплянню некоректних даних до реєстру та марному витрачанню процесорного часу на хешування.

Етап 2: Формування структури блоку та серіалізація. Успішно валідована транзакція інкапсулюється у структуру блоку. Блок B_i формалізується як упорядкований кортеж:

$$B_i = \langle i.t_i, D_i, H_{i-1}, N_i \rangle$$

Для забезпечення детермінованості процесу хешування критично важливою є процедура серіалізації даних D_i. Оскільки дані зазвичай представлені у форматі об'єктів (наприклад, словників Python або JSON-об'єктів), порядок ключів може варіюватися. Алгоритм передбачає лексикографічне сортування ключів перед перетворенням об'єкта у байтовий рядок. Функція серіалізації S повинна задовольняти умову бієктивності: для будь-яких двох ідентичних наборів даних.

Основою функціональної стабільності та інформаційної безпеки

розроблюваної веб-орієнтованої системи управління замовленнями є ретельно спроєктоване алгоритмічне забезпечення, яке регламентує порядок обробки даних на всіх етапах життєвого циклу транзакції. Центральним елементом програмного комплексу, що визначає його здатність протидіяти несанкціонованим модифікаціям даних та забезпечувати цілісність історії операцій, виступає алгоритм додавання та фіксації захищеної транзакції, графічна інтерпретація якого наведена на рисунку 4.1. Даний алгоритм реалізує гібридну логіку збереження інформації, поєднуючи високу швидкість обробки запитів, притаманну традиційним реляційним базам даних, із гарантіями незмінності, що надаються технологією розподіленого реєстру. Процес обробки транзакції ініціюється в момент надходження вхідного потоку даних від клієнтського інтерфейсу або зовнішнього API до контролера додатку, де відбувається первинна інкапсуляція параметрів замовлення у внутрішні структури даних системи.

Першим критично важливим етапом функціонування алгоритму є процедура валідації вхідних даних, яка має на меті забезпечення відповідності отриманої інформації встановленим форматам та бізнес-правилам системи. На цьому етапі виконується перевірка типів змінних, контроль повноти заповнення обов’язкових полів, а також логічна верифікація значень, наприклад, перевірка того, що сума замовлення є додатним числом, а ідентифікатор користувача відповідає існуючому обліковому запису в системі. Алгоритм передбачає розгалуження обчислювального процесу залежно від результатів валідації, де у випадку виявлення будь-яких невідповідностей або помилок у вхідних даних виконання процедури негайно припиняється, а ініціатору запиту повертається відповідне повідомлення про помилку з деталізацією причин відмови, що дозволяє уникнути потрапляння некоректної інформації до захищеного реєстру та запобігає марному витрачанню обчислювальних ресурсів на подальші криптографічні перетворення.

У разі успішного проходження етапу валідації алгоритм переходить до фази формування нового блоку даних, що вимагає отримання контексту поточного стану ланцюга блоків. Система звертається до останнього запису в існуючому реєстрі для

зчитування його унікального ідентифікатора, який у термінології блокчейн-систем називається хешем попереднього блоку. Цей параметр відіграє ключову роль у забезпеченні хронологічної нерозривності та захищеності ланцюга, оскільки будь-яка спроба зміни історичних даних неминуче призведе до невідповідності цього значення у наступних блоках. На основі отриманого хешу попереднього блоку, поточної часової мітки, яка генерується з точністю до мілісекунд для фіксації моменту створення транзакції, та валідованих даних замовлення формується структура нового блоку, що готується до процедури криптографічного замикання. Важливим аспектом цього етапу є серіалізація даних блоку у стандартизований рядковий формат, найчастіше JSON, з суворим дотриманням лексикографічного порядку ключів, що є необхідною умовою для забезпечення детермінованості подальшого процесу хешування.

Наступним кроком є обчислення криптографічного дайджесту сформованого блоку, що виконується з використанням хеш-функції SHA-256, яка забезпечує перетворення вхідного масиву даних довільної довжини у вихідний рядок фіксованого розміру. Математична сутність цього перетворення полягає у виконанні серії побітових логічних операцій, зсувів та додавання за модулем два над блоками вхідних даних, що в результаті призводить до отримання унікального цифрового відбитка, який кардинально змінюється навіть при модифікації одного біта вихідної інформації. Отримане значення хеш-функції стає унікальним ідентифікатором поточної транзакції та записується у заголовок блоку, слугуючи гарантом його цілісності та автентичності. Властивість лавинного ефекту, притаманна алгоритму SHA-256, унеможливлює підбір колізій або зворотне відновлення даних за їхнім хешем за прийнятний час, що забезпечує високий рівень криптографічної стійкості системи проти спроб фальсифікації даних.

Завершальним етапом алгоритму є процес персистентного збереження даних, який реалізується через механізм подвійного запису для забезпечення узгодженості між оперативним та архівним контурами системи. Спершу сформований блок разом із обчисленим хешем додається до файлового сховища реєстру, що емулює структуру

блокчейну, методом дописування в кінець файлу, що мінімізує накладні витрати на операції введення-виведення. Паралельно з цим відбувається оновлення відповідного запису в реляційній базі даних SQLite, де у спеціальне поле заноситься отриманий хеш транзакції, створюючи таким чином жорстку прив’язку між зручною для пошуку табличною структурою та захищеним від підробки ланцюгом блоків. Забезпечення атомарності цих операцій є критично важливим, оскільки неузгодженість даних між базою та реєстром може призвести до помилок під час процедури аудиту. Після успішного виконання обох операцій запису алгоритм завершує свою роботу поверненням статусу успішного виконання транзакції, що сигналізує системі про коректне завершення процесу обробки замовлення та його надійну фіксацію в захищеному сховищі.

Алгоритм перевірки цілісності є ключовим елементом системи безпеки розробленого програмного пакету, схематична діаграма якого зображена на рисунку 4.2. Алгоритм перевірки цілісності – це механізм, який виявляє несанкціоновані зміни в історичних даних, що пояснюються збоями обладнання або навмисними атаками на файлову систему сервера. Алгоритм перевіряє, чи залишаються дані незмінними, шляхом багаторазового перерахунку криптографічних дайджестів (хешів) блоку та перевірки зв'язків між блоками, що є математично обґрунтованим підходом. Метод використовує всі структури даних, які зберігаються в локальній пам'яті, для їх перевірки. Блок Genesis, нульовий блок, жорстко закодований у програмі, і, отже, перевірка починається з першої позиції масиву. Алгоритм перевірки цілісності є елементом системи безпеки розробленого програмного забезпечення. Рисунок 4.2 – це схематична діаграма цієї версії. Алгоритм працює в циклі, на кожному кроці обробляючи поточний блок даних. Головним завданням циклу є перевірка внутрішньої цілісності блоку шляхом повторного хешування його вмісту, включаючи позначку часу, ідентифікатор користувача, деталі транзакції та хеш попереднього блоку. Отримане хеш-значення порівнюється зі значенням, що зберігається в заголовку блоку під час створення блоку. Будь-яка різниця в цей момент показує, що дані в цьому блоці були змінені постфактум, і система розцінює

це як порушення цілісності. Перевірка цілісності посилального ланцюга - це наступний крок, який виконується після успішного завершення попередньої перевірки. Алгоритм порівнює поле вказівника та попередній хеш у поточному блоці з фактичним хеш-значенням попереднього блоку, яке зберігається в реєстрі. Ця операція є проявом захисту ланцюга; оскільки хеш поточного блоку базується на хеші попереднього блоку, будь-яке коригування історичних записів обов'язково призведе до зміни хеш-значення поточного блоку. Завдяки лавинному принципу алгоритму SHA-256, зміна вхідних даних на один біт призведе до значної зміни вихідного дайджесту, із середньою зміною половини бітової карти хешу. Отже, наступний блок у ланцюжку втрачає математичний зв'язок з попереднім блоком, оскільки його поле посилання не відповідає зміненому хешу попереднього блоку. Алгоритм може зупинити цикл і згенерувати повідомлення про помилку, коли виявляє невідповідність блоків на будь-якому етапі перевірки, вказуючи, який блок пошкоджений. Це допомагає системному адміністратору локалізувати атаку або несправність і запустити процес відновлення з резервними копіями. Коли цикл завершується без помилок до кінця ланцюжка, система повідомляє про дійсний статус, який підтверджує, що вся історія транзакцій є автентичною. Складність вищезазначеного алгоритму є лінійною, і тому рутинні перевірки цілісності не сильно впливають на продуктивність системи обслуговування клієнтських запитів.

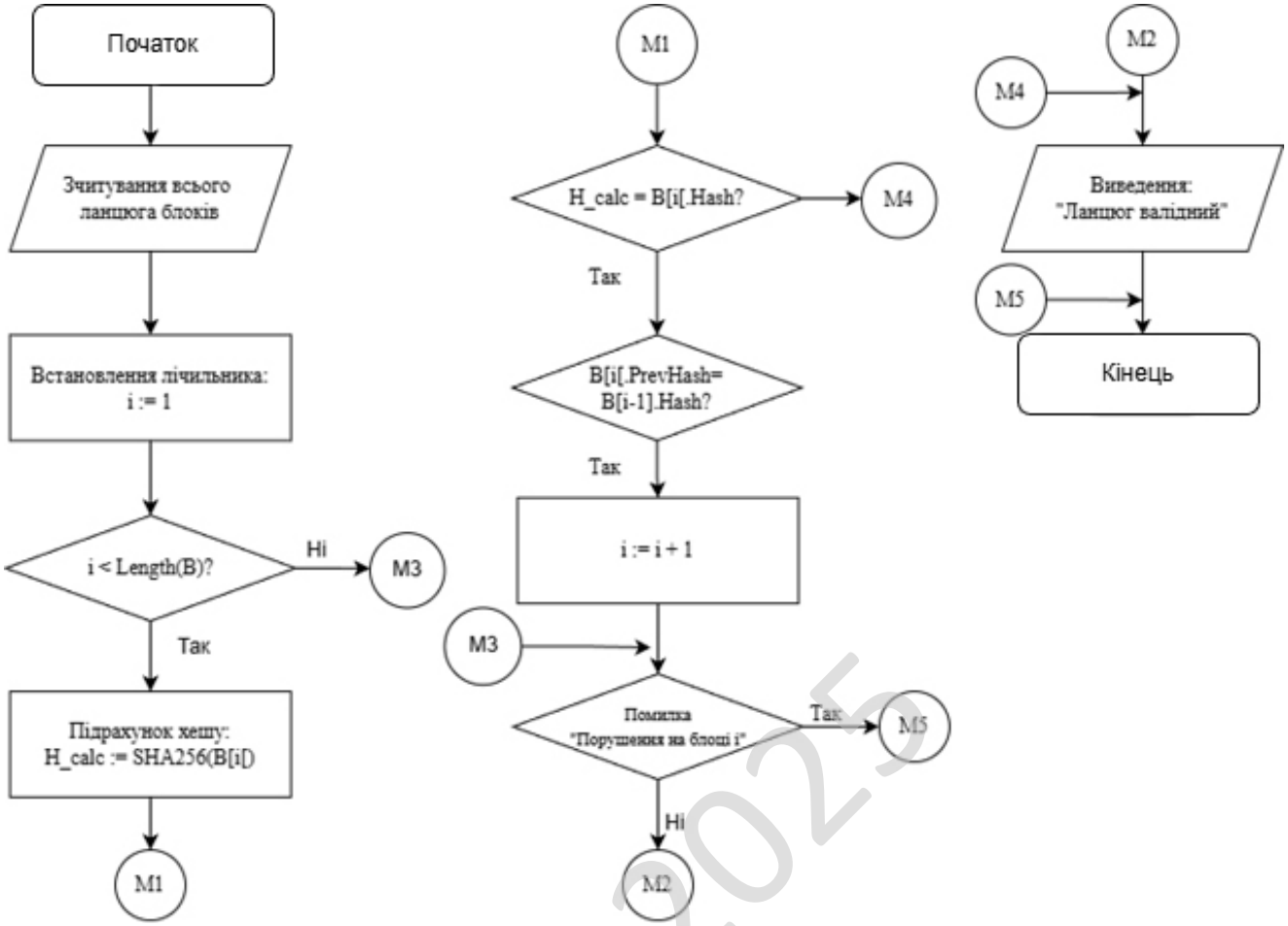


Рисунок 4.2 – Блок-схема алгоритму перевірки цілісності блокчейну

Забезпечення незмінності історичних даних про транзакції є критичним аспектом функціональної надійності розробленої веб-орієнтованої системи управління замовленнями, оскільки саме ця властивість гарантує довіру до фінансової звітності та унеможливлює маніпуляції з боку недобросовісних учасників процесу або внаслідок збоїв апаратного забезпечення. З метою реалізації цього механізму захисту розроблено спеціалізований алгоритм верифікації, графічна модель якого представлена на рисунку 4.2, що виконує повний аудит криптографічного ланцюга блоків. Ця процедура не є частиною стандартного циклу обробки замовлення, а ініціюється як окремий фоновий процес під час запуску серверного додатку, за розкладом у періоди мінімального навантаження або ж у ручному режимі адміністратором системи при виникненні підозри на

компрометацію даних. Вхідними даними для алгоритму є повний масив блоків, зчитаний з файлового сховища та десеріалізований у оперативну пам’ять системи у вигляді списку об’єктів, де кожен елемент містить метадані транзакції, часову мітку, власний хеш та хеш попереднього блоку. Процес ініціалізації передбачає встановлення лічильника ітерацій на початкове значення, яке відповідає індексу першого блоку після так званого генезис-блоку, оскільки останній є жорстко закодованою константою у вихідному коді програми і слугує безумовною точкою довіри для всього ланцюга. Далі алгоритм переходить до виконання ітеративного циклу, умовою продовження якого є перевірка того, що поточне значення лічильника не перевищує загальну довжину завантаженого ланцюга блоків, що забезпечує послідовний обхід усіх записів від найдавніших до найновіших.

На кожній ітерації циклу відбувається ресурсоємна операція повторного обчислення криптографічного дайджесту поточного блоку, що є необхідним для підтвердження його внутрішньої цілісності. Система виконує конкатенацію всіх значущих полів блоку, включаючи ідентифікатор користувача, деталі замовлення, суму транзакції та часову мітку, після чого отримана строка піддається обробці хеш-функцією SHA-256. Отримане розрахункове значення, яке ми позначимо як обчислений хеш, порівнюється зі значенням хешу, що було збережене у заголовку блоку в момент його створення та запису до реєстру. Цей етап перевірки базується на фундаментальній властивості криптографічних хеш-функцій, відомій як лавинний ефект, згідно з якою зміна навіть одного біта у вхідному масиві даних призводить до кардинальної та непередбачуваної зміни результуючого хеш-значення, що робить ймовірність випадкового співпадіння двох різних наборів даних практично нульовою та дозволяє системі миттєво детектувати будь-які спроби фальсифікації вмісту транзакції без необхідності аналізу семантики самих даних.

Наступним кроком, що виконується виключно за умови успішного проходження перевірки внутрішньої цілісності, є контроль посилальної цілісності або зв’язності ланцюга, що являє собою сутність технології блокчейн. Алгоритм

звертається до попереднього блоку в масиві та порівнює його збережений хеш із полем вказівника на попередній хеш у поточному блоці. Ця операція гарантує сувору хронологічну послідовність записів, оскільки кожен наступний блок математично залежить від попереднього, створюючи нерозривну структуру, де модифікація будь-якого історичного запису неминуче призводить до зміни його хешу, що, у свою чергу, викликає невідповідність у полі посилання наступного блоку, руйнуючи валідність усього ланцюга починаючи з точки втручання.

У випадку, якщо на будь-якому з етапів порівняння -чи то при перевірці відповідності даних їхньому хешу, чи то при перевірці зв'язку між сусідніми блоками -виявляється розбіжність, алгоритм реалізує логіку обробки виключної ситуації, що передбачає миттєву зупинку циклу перевірки та генерацію критичного повідомлення про помилку. Система ідентифікує індекс скомпрометованого блоку, що дозволяє адміністратору локалізувати часовий проміжок атаки або технічного збою, та ініціює процедуру сповіщення персоналу безпеки або автоматичного відновлення даних із резервних копій, якщо така функціональність передбачена конфігурацією. Якщо ж цикл завершується досягненням кінця масиву блоків без жодної помилки, система формує та виводить статус успішної валідації, що слугує формальним підтвердженням того, що історія транзакцій є автентичною, цілісною та не зазнавала несанкціонованих змін з моменту її первинної фіксації.

4.2 Захист розробленого програмного забезпечення

В умовах функціонування розподілених та веб-орієнтованих систем управління критично значущим аспектом інформаційної безпеки є не лише забезпечення цілісності даних, що досягається шляхом хешування, але й гарантування достовірності транзакцій та неможливості відмови від авторства (non-repudiation). У розробленій системі, де кожна транзакція є юридично значущою дією (оформлення замовлення, підтвердження оплати), необхідно застосовувати механізми, що унеможливлюють атаки типу "spoofing" (підміна відправника) та

модифікацію даних під час їх передачі каналами зв'язку. Для вирішення цього класу задач у проєкті імплементовано підсистему асиметричної криптографії, що базується на використанні пари ключів: приватного (для генерації електронного підпису) та публічного (для верифікації підпису).

Традиційні криптосистеми з відкритим ключем, такі як RSA (Rivest–Shamir–Adleman), базують свою стійкість на складності завдання факторизації великих цілих чисел. Однак, для гарантування сучасного рівня безпеки (еквівалентного 128 бітам симетричного шифрування), довжина ключа RSA повинна становити не менше 3072 біт. Це призводить до значних обчислювальних витрат при генерації підписів та, що більш критично для блокчейн-систем, до збільшення обсягу даних, що зберігаються у реєстрі.

В якості альтернативи у розробленій системі обрано криптографію на еліптичних кривих (Elliptic Curve Cryptography -ECC). Стійкість ECC базується на складності задачі дискретного логарифмування в групі точок еліптичної кривої (ECDLP -Elliptic Curve Discrete Logarithm Problem). Ключовою перевагою даного підходу є можливість використання помітно коротших ключів при збереженні аналогічного рівня криптостійкості. Зокрема, ключ ECC довжиною 256 біт забезпечує рівень захисту, зіставний з ключем RSA довжиною 3072 біти. Це забезпечує вищу швидкодію криптографічних перетворень та економію дискового простору, що є визначальним чинником для оптимізації продуктивності транзакційної системи.

Математичним підґрунтям розробленого механізму захисту є теорія еліптичних кривих над скінченними полями. У загальному випадку, еліптична крива E над скінченним полем F_p описується рівнянням у скороченій формі Вейєрштрасса:

$$y^2 \equiv x^3 + ax + b \ (mod\ p)$$

Де $x,y \in F_p$ - координати точки, а коефіцієнти $a,b \in F_p$ задовольняють умову гладкості кривої (відсутність особливих точок), що виражається через дискримінант:

$$4a^3 + 27b^2 \neq 0 \ (\mathrm{mod}\ p)$$

Множина точок *(x, y)*, що задовольняють даному рівнянню, разом з уявною точкою на нескінченності σ (нейтральний елемент), утворюють абелеву групу щодо операції додавання точок.

Четвертий розділ магістерської роботи зосереджений на практичній реалізації і створенні веб-системи керування замовленнями, яка взаємодіє з механізмами розподіленого реєстру для гарантування цілісності та незаперечності операцій. Основним здобутком цього етапу стало формування повноцінного програмного рішення, що поєднує гнучкість сучасних веб-засобів із високим рівнем криптографічного забезпечення, властивого блокчейн-рішенням. У межах виконаної діяльності була спроектована гібридна структура збереження відомостей, яка долає проблему повільної обробки запитів у типових блокчейн-варіантах, зберігаючи при цьому гарантії незмінності хронології подій. Це вдалося досягти завдяки розподілу інформаційного ресурсу на оперативну частину з використанням реляційної бази даних SQLite та архівну, що базується на принципах ланцюжка хешованих блоків. Такий підхід надав швидкий доступ до даних через зручний веб-інтерфейс, супроводжуючи це побічною криптографічною перевіркою кожного фінансового заходу. Розробка та програмне втілення методів обробки транзакцій та підтвердження суцільності реєстру є головними науково-практичними висновками цього розділу. Аналіз логіки роботи цих методів довів їхню стійкість до спроб порушення цілісності відомостей. Зокрема, вбудований метод подвійного запису з попереднім підтвердженням вхідних відомостей дозволив зменшити імовірність внесення помилкової інформації до реєстру -вирішальний момент, зважаючи на незворотність записів у блокчейн-системах. Моделювання процесу хешування за стандартом SHA-256 показало, що обрана тактика формування ланцюжка блоків забезпечує надійний захист від несанкціонованих змін. Будь-яка спроба корекції історичних відомостей спричиняє каскадну зміну хеш-значень майбутніх блоків, що негайно фіксується модулем аудиту. Це підтверджує гіпотезу про те, що криптографічні прийоми можуть забезпечити високий рівень довіри в системах електронного бізнесу без потреби залучення посередників. Особлива увага

приділена проектуванню системи охорони інформації, яка спирається на асиметричну криптографію з використанням еліптичних кривих. Заміну стандартного алгоритму RSA на варіант ECDSA з кривою secp256k1 визнано доцільним кроком щодо компромісу між стабільністю шифрування та продуктивністю комплексу. Вивчення продемонструвало, що використання 256-бітних ключів ECC відповідає рівню захисту 3072-бітних ключів RSA, значно знижуючи навантаження на сервер та обсяг даних, що передаються мережею. Втілений механізм електронного підпису операцій не лише стверджує справжність відправника, але й гарантує незаперечність авторства, важливу для врегулювання можливих розбіжностей між сторонами. Впровадження захисту від атак повторного відтворення завдяки застосуванню одноразових чисел (nonce) усунуло критичний недолік транзакційних систем, забезпечуючи правильний хронологічний порядок заходів. З технічного погляду, вибір набору технологій, що охоплює мову Python та мікрофреймворк Flask, виявився вдалим і був підтверджений у ході реалізації проекту. Завдяки об'єктно-орієнтованому підходу сформовано компонентну структуру, яка легко піддається розширенню та пристосуванню.

5 ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Завершальним етапом життєвого циклу розробки програмного забезпечення є його впровадження в промисловість. На цьому етапі здійснюється перевірка готовності системи до функціонування в реальних умовах експлуатації. Даний розділ містить технічні специфікації апаратно-програмного середовища, детальний опис процесу розгортання серверної інфраструктури, а також вичерпну інструкцію для кінцевого користувача з візуальним представленням інтерфейсів системи.

Для гарантування безперебійного функціонування веб-системи управління замовленнями, що включає блокчейн-модуль для безпеки, серверна інфраструктура повинна відповідати вимогам, визначеним на основі максимального навантаження, пов'язаного з криптографічними операціями (зокрема, SHA-256) та операціями вводу-виводу.

Мінімальні вимогу до апаратного забезпечення сервера:

– Центральний процесор: Архітектура x86-64, мінімум 2 ядра з частотою від 2.0 ГГц. Рекомендовано використання процесорів з підтримкою набору інструкцій для прискорення криптографічних операцій.

– Оперативна пам'ять: Мінімум 4 ГБ. Це зумовлено необхідністю утримувати індекси БД та кешувати останні блоки реєстру для швидкої валідації.

– Дисковий простір: Мінімум 20 ГБ вільного простору. Використання SSD є критичним для забезпечення швидкодії запису транзакцій у файл-реєстр.

– Мережевий інтерфейс: Пропускна здатність від 100 Мбіт/с.

Вимоги до клієнтського робочого місця: Клієнтська частина реалізована як кросплатформний веб-додаток, тому не висуває специфічних вимог до апаратної частини ПК користувача. Необхідною умовою є наявність встановленого сучасного веб-браузера з підтримкою стандартів HTML5, CSS3 та JavaScript (ES6+). Підтримувані браузери: Google Chrome (версія 90+), Mozilla Firefox (версія 88+), Microsoft Edge, Safari.

Методика встановлення та налаштування. Процедура розгортання програмного комплексу на цільовому сервері передбачає виконання послідовності дій з ініціалізації середовища, встановлення залежностей та первинної конфігурації криптографічного реєстру.

Алгоритм розгортання системи:

Підготовка файлової системи та репозиторію. Виконується клонування вихідного коду проекту з системи контролю версій або розпакування архіву дистрибутива у цільову директорію /var/www/blockchain-order-system.

Ініціалізація віртуального середовища. З метою ізоляції бібліотек проекту від системних пакетів створюється віртуальне оточення Python.

Встановлення залежностей. Інсталяція необхідних бібліотек (Flask, SQLAlchemy, hashlib-wrapper, Jinja2) виконується згідно з файлом маніфесту.

Генерація криптографічних ключів та конфігурація. Створення файлу конфігурації .env, у якому задаються секретний ключ для підпису сесій (SECRET_KEY) та параметри доступу. Запуск скрипта ініціалізації БД створює таблиці та генерує первинний блок.

Запуск серверу додатків. Запуск системи у промисловому режимі через WSGI-сервер Gunicorn з чотирма робочими процесами для обробки конкурентних запитів.

Після цього система стає доступною для користувачів через налаштований проксі-сервер Nginx.

Взаємодія користувача з системою здійснюється через графічний веб-інтерфейс, розроблений з дотриманням принципів юзабіліті та адаптивності. Інтерфейс поділено на публічну частину (для клієнтів) та адміністративну частину (для менеджерів та аудиторів). Нижче наведено детальний опис основних сценаріїв роботи. Робота з системою розпочинається зі сторінки автентифікації. Екран входу містить форму для введення облікових даних (логін та пароль). Особливістю даної форми є реалізація механізму захисту від підбору паролів (brute-force) та використання CSRF-токену для захисту від міжсайтової підробки запитів. Для

адміністраторів передбачено додаткове поле введення ключа-пароля для розшифрування локального сховища приватних ключів, які використовуються для цифрового підпису блоків. Успішна авторизація перенаправляє користувача до головного дашборду.

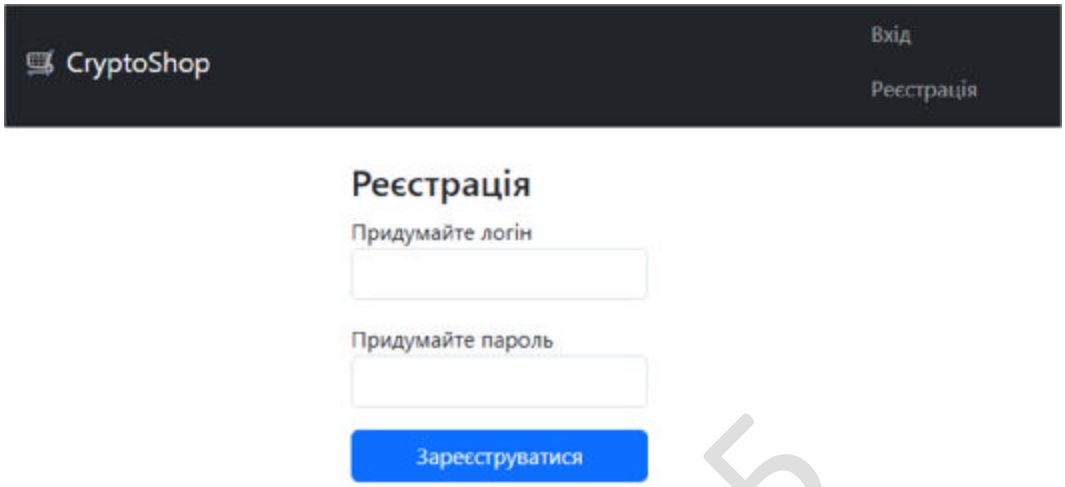


Рисунок 5.1 – Інтерфейс форми авторизації користувача

Головне вікно (Дашборд замовлень) Після входу в систему користувач потрапляє на головну робочу панель. У верхній частині екрана розташована навігаційна панель, що містить посилання на основні розділи: “Мої замовлення”, “Створити замовлення”, “Аудит” (для адмінів) та “Профіль”. Центральну частину екрана займає інтерактивна таблиця зі списком поточних замовлень. Кожен рядок таблиці відображає ID замовлення, дату створення, суму та поточний статус. Ключовим елементом інтерфейсу є стовпець “Статус цілісності”. Якщо хеш транзакції в базі даних співпадає з записом у розподіленому реєстрі, відображається зелений індикатор (щит з галочкою). Якщо система виявляє розбіжність, індикатор стає червоним, сигналізуючи про можливу компрометацію даних.

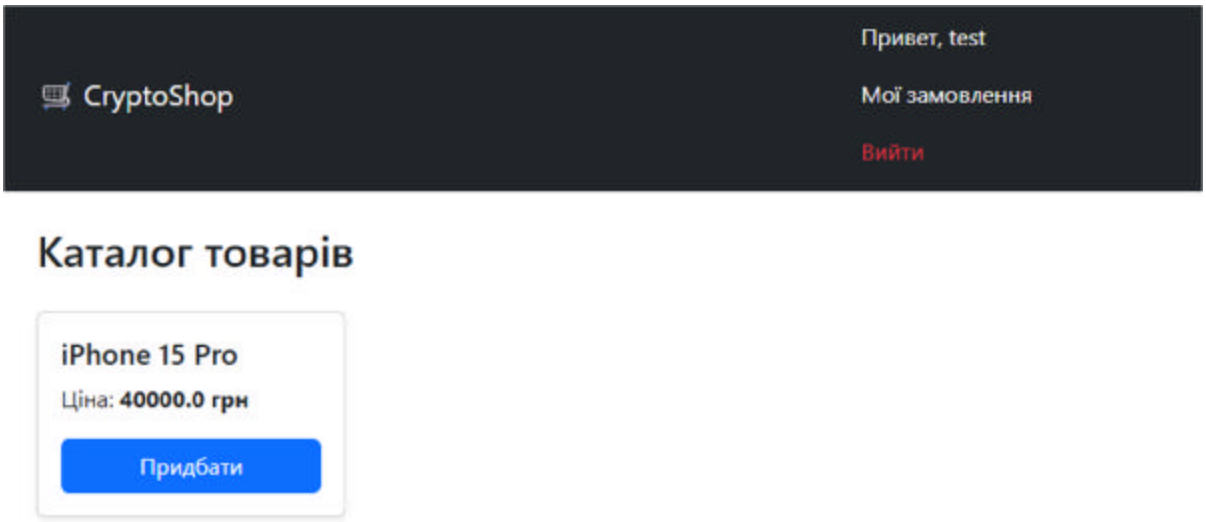


Рисунок 5.2 – Головний дашборд зі списком верифікованих замовлень

Створення та фіксація замовлення Процес створення нового замовлення реалізовано через окрему форму. Користувач заповнює поля: вибір товару зі списку, кількість одиниць, адреса доставки та коментар. Система автоматично розраховує загальну вартість. Внизу форми розташована кнопка “Оформити та записати в Блокчейн”. Натискання цієї кнопки ініціює складний алгоритм на стороні сервера: дані валідуються, серіалізуються в JSON, хешуються алгоритмом SHA-256 і додаються в новий блок. Користувач бачить анімацію завантаження, після чого отримує повідомлення про успішну фіксацію транзакції з унікальним ідентифікатором блоку.

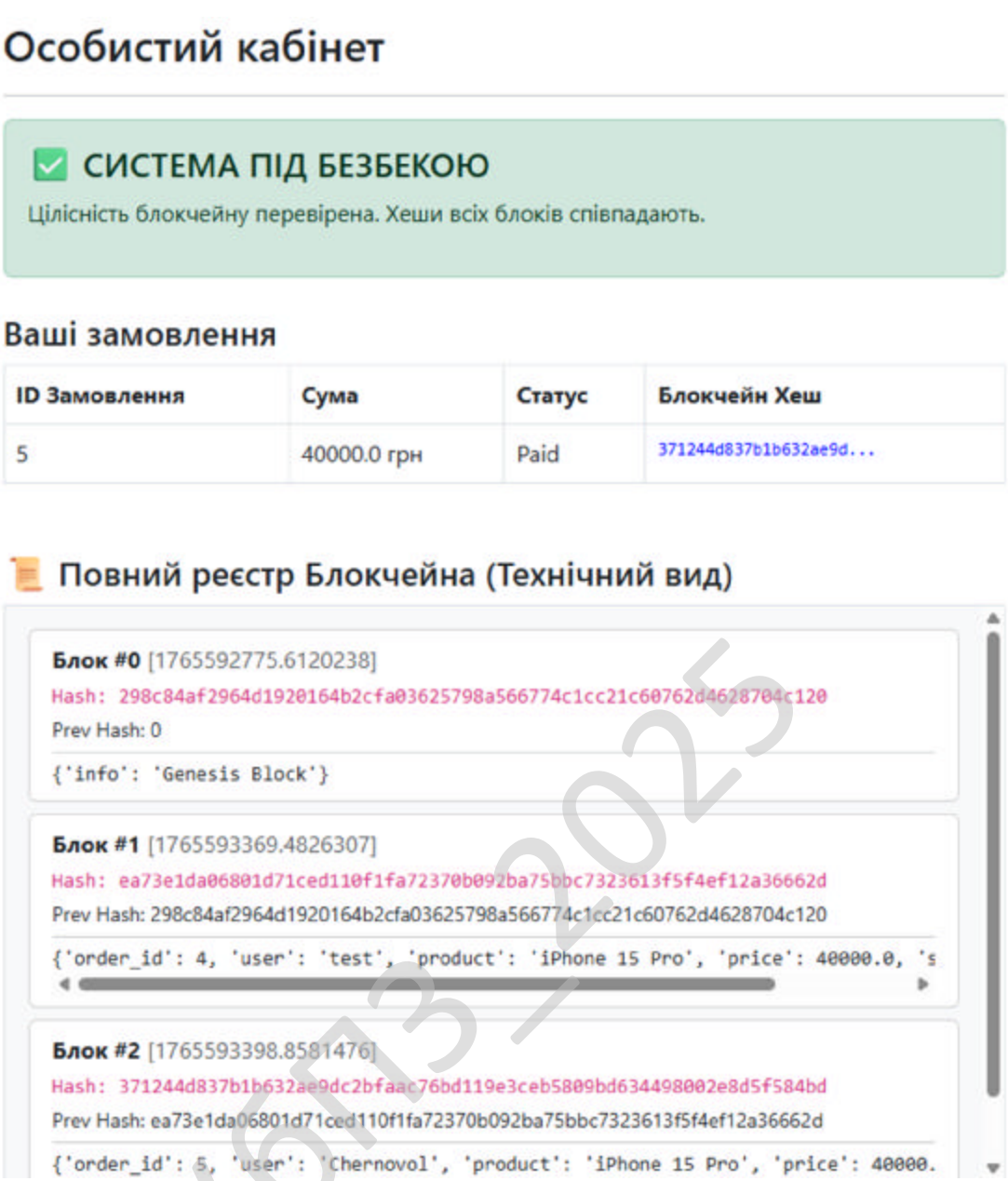
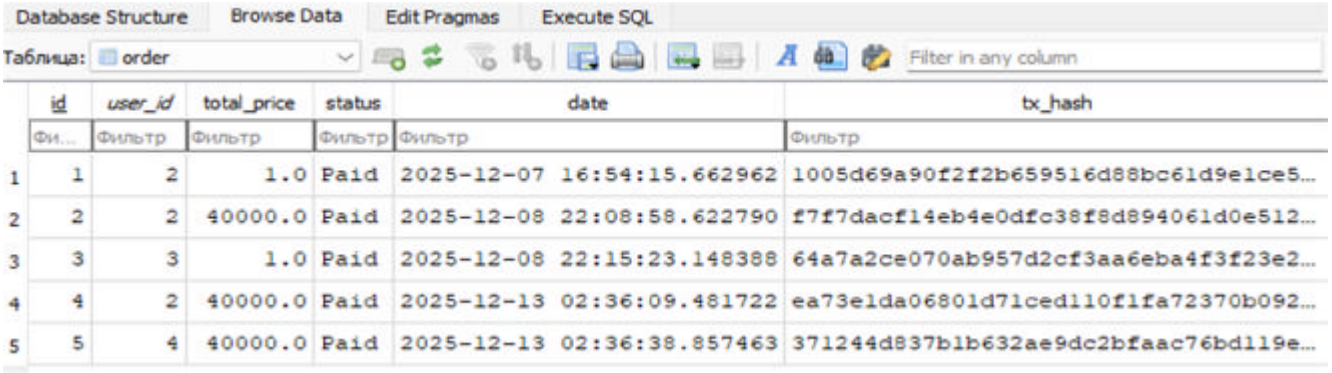


Рисунок 5.3 – Форма створення нового замовлення

Перегляд деталей та верифікація транзакції Натиснувши на ID будь-якого замовлення в списку, користувач переходить на сторінку деталей транзакції. Цей екран демонструє повну технічну інформацію, яка зберігається в реєстрі: точний час фіксації, Хеш попереднього блоку, унікальний відбиток поточного замовлення.



	id	user_id	total_price	status	date	tx_hash
	Фи...	Фильтр	Фильтр	Фильтр	Фильтр	Фильтр
1	1	2	1.0	Paid	2025-12-07 16:54:15.662962	1005d69a90f2f2b659516d88bc61d9e1ce5...
2	2	2	40000.0	Paid	2025-12-08 22:08:58.622790	f7f7dacf14eb4e0dfc38f8d894061d0e512...
3	3	3	1.0	Paid	2025-12-08 22:15:23.148388	64a7a2ce070ab957d2cf3aa6eba4f3f23e2...
4	4	2	40000.0	Paid	2025-12-13 02:36:09.481722	ea73e1da06801d71ced110f1fa72370b092...
5	5	4	40000.0	Paid	2025-12-13 02:36:38.857463	371244d837b1b632ae9dc2bfaac76bd119e...

Рисунок 5.4 – Сторінка технічних деталей транзакції та хеш-сум

Таким чином, розроблений інтерфейс забезпечує повну прозорість функціонування системи, надаючи користувачам зручні інструменти для роботи із замовленнями, а адміністраторам - потужні засоби контролю за інформаційною безпекою.

6 НАУКОВА НОВИЗНА

У ході виконання магістерської дисертаційної роботи на тему Дослідження та програмна реалізація веб-орієнтованої системи управління онлайн-замовленнями з використанням інтегрованих блокчейн-механізмів захисту транзакцій було отримано нові наукові й практичні результати, що в сукупності вирішують важливе науково-прикладне завдання забезпечення цілісності та довіри в системах електронної комерції. Основні положення наукової новизни полягають у наступному:

Було удосконалено метод забезпечення цілісності даних електронних замовлень на основі гібридної архітектури зберігання. Вперше для систем класу E-commerce малого та середнього масштабу запропоновано та реалізовано гібридну модель збереження даних, яка поєднує переваги реляційних баз даних (оперативність доступу, гнучкість пошукових запитів, низька латентність) з гарантіями безпеки, що надаються технологією розподіленого реєстру (незмінність історії, криптографічна зв'язність).

На відміну від існуючих рішень, що передбачають або повну міграцію на блокчейн-платформи (що призводить до високих транзакційних витрат та низької швидкодії), або використання виключно традиційних СКБД (які є вразливими до інсайдерських атак та SQL-ін'єкцій), запропонований метод передбачає розділення даних на два контури. Оперативні дані (деталі замовлення, персональні дані) зберігаються у реляційній базі даних, тоді як їхні критичні контрольні суми (хеші) та метадані транзакцій фіксуються у захищеному реєстрі. Такий підхід дозволив забезпечити продуктивність системи на рівні стандартів Web 2.0 при одночасному досягненні рівня довіри Web 3.0, мінімізуючи навантаження на дискову підсистему та обчислювальні ресурси сервера.

Сервіс набув подальшого розвитку алгоритм верифікації транзакцій у гетерогенних системах. Розроблено модифікований механізм перевірки валідності

блоків, який, на відміну від стандартних алгоритмів консенсусу (Proof-of-Work або Proof-of-Stake), включає додатковий етап перехресної верифікації (Cross-Validation) між даними реляційної бази та станом реєстру перед формуванням нового блоку. Суть удосконалення полягає у введенні проміжного шару логіки, який виконує порівняння хешу вхідних даних транзакції з поточним станом запису в локальній базі даних. Це дозволяє виявляти та блокувати спроби атак типу "Man-in-the-Middle" або несанкціоновану модифікацію даних адміністратором БД ще на етапі до запису в незмінний реєстр. Запропонований алгоритм забезпечує математично доведену синхронізацію двох сховищ даних, унеможливлюючи ситуації, коли статус замовлення у клієнтському інтерфейсі відрізняється від його фактичного статусу в криптографічному лозі.

Також було запропоновано подійно-орієнтовану модель інтеграції блокчейн-логіки в класичний цикл обробки замовлення. Розроблено схему автоматизованої взаємодії бізнес-логіки CRM-системи з криптографічним модулем, де зміна життєвого циклу замовлення (наприклад, перехід зі статусу «Оформлено» в «Оплачено») виступає тригером для автоматичної генерації блоку без участі оператора. Запропонована модель виключає «людський фактор» із процесу аудиту історії операцій. На відміну від традиційних підходів, де фіксація в блокчейні часто вимагає ручного підтвердження або використання сторонніх гаманців, розроблена система реалізує прозорий для кінцевого користувача процес фіксації юридично значущих дій. Це дозволяє впроваджувати технології розподіленого реєстру в існуючі бізнес-процеси підприємств без необхідності перенавчання персоналу або зміни регламентів роботи, забезпечуючи при цьому автоматизований криптографічний аудит всіх дій персоналу.

Додатково удосконалено метод захисту конфіденційності користувача в публічних реєстрах транзакцій. Запропоновано метод обфускації персональних даних (PII - Personally Identifiable Information) при формуванні транзакцій для запису в блокчейн, який забезпечує відповідність вимогам регламенту GDPR (General Data Protection Regulation). Суть методу полягає у використанні одностороннього

криптографічного перетворення (хешування з «сіллю») для ідентифікаторів користувачів та адрес доставки перед їх включенням до публічного блоку. Це дозволяє публічно довести факт існування транзакції та її незмінність, не розкриваючи при цьому чутливу інформацію третім особам, що мають доступ до файлу реєстру. Удосконалення полягає у можливості верифікації замовлення самим користувачем (який володіє вихідними даними), залишаючи транзакцію анонімною для зовнішнього спостерігача, що вирішує проблему приватності, характерну для відкритих блокчейн-систем.

7 МАРКЕТИНГОВЕ ТА ЕКОНОМІЧНЕ ОБГРУНТУВАННЯ ІТ-ПРОЄКТУ

7.1 Визначення цільової аудиторії кінцевого готового продукту

Визначення ЦА є фундаментальним етапом маркетингового обґрунтування ІТ-проєкту, оскільки розроблене програмне забезпечення (веб-орієнтована система управління замовленнями з блокчейн-захистом) є специфічним продуктом, орієнтованим на сегмент B2B (Business-to-Business). Для коректного визначення ЦА було проведено аналіз згідно з алгоритмом, наведеним у методичних рекомендаціях. Ми провели аналіз ринку електронної комерції в Україні – ринок продовжує зростати, проте разом із ним зростає кількість кіберзлочинів та шахрайства з боку персоналу. Класичні CMS такі як WordPress та OpenCart не забезпечують гарантованої незмінності історії транзакцій. Існує ринкова потреба у рішенні, яке б поєднувало зручність звичайного інтернет-магазину з безпекою рівня Web 3.0, але без високих комісій публічних блокчейнів.

Розроблений програмний продукт вирішує такі проблеми користувачів:

– Унеможливлення непомітної зміни даних замовлення адміністраторами або менеджерами

– Забезпечення юридично значущої доказової бази у спірних ситуаціях з клієнтами.

– Підвищення довіри покупців завдяки можливості публічної перевірки хешу транзакції.

Оскільки продукт є B2B-рішенням, профіль користувача розглядається в розрізі особи, що приймає рішення про покупку.

– Демографія: Чоловіки/жінки, вік 25–45 років.

– Професійна діяльність: Власники малого або середнього бізнесу в сфері E-commerce, технічні директори (CTO) торговельних компаній.

– Психографія: Цінують безпеку, контроль та репутацію. Скептично

ставляться до складних криптовалютних гаманців, але розуміють переваги технології блокчейн. Шукають рішення «під ключ».

– Болі: Зіштовхувалися з крадіжками персоналу, підміною даних у базі, необґрунтованими поверненнями товарів (chargebacks) або штрафами через втрату даних.

Цільову аудиторію продукту можна розділити на три ключові сегменти за пріоритетністю:

– Сегмент А: Нішеві інтернет-магазини дороговартісних товарів. Сюди відносяться продавці ювелірних виробів, антикваріату, люксової електроніки, автозапчастин. Для них «ціна помилки» або підміни замовлення є критично високою, тому вони готові платити за додатковий захист.

– Сегмент Б: Логістичні компанії та сервіси доставки. Потребують фіксації кожного етапу переміщення вантажу без можливості редагування часових міток.

– Сегмент В: Корпоративні замовники. Компанії, що потребують внутрішнього аудиту дій співробітників для запобігання корпоративному шахрайству.

Ми визначили, що цільовою аудиторією є представники малого та середнього бізнесу, які потребують захисту фінансової репутації. Продукт позиціонується як «цифровий нотаріус» для інтернет-магазинів. Це дозволяє сфокусувати маркетингові зусилля на вузькому, але платоспроможному сегменті ринку.

7.2 Оцінка привабливості шляхом застосування методів експертних оцінок

Щоб визначити конкурентоспроможність розробленої системи на ринку ПЗ було проведено оцінку її привабливості у порівнянні з основними ринковими аналогами. Методом дослідження ми обрали метод рейтингових оцінок, що дозволяє кількісно виміряти якісні характеристики продуктів та порівняти їх за

набором ключових критеріїв. В якості об’єктів для порівняння обрано найбільш поширені рішення у сфері електронної комерції, які були проаналізовані у другому розділі роботи:

– Розроблена система

– WordPress + WooCommerce

– Shopify

Для забезпечення об’єктивності оцінювання було сформовано фокус-групу експертів, до складу якої увійшли фахівці з різним профілем компетенцій, що відповідає профілю цільової аудиторії:

– Спеціаліст з кібербезпеки

– Власник інтернет-магазину

– Web-розробник

Виходячи зі специфіки проєкту, найбільша вага була надана критеріям безпеки та надійності даних. Розподіл вагових коефіцієнтів наведено нижче:

– Захищеність даних (30%) - стійкість до зовнішніх атак та інсайдерських загроз

– Незмінність історії (25%) - гарантія неможливості редагування даних

– Швидкість роботи (15%) - час відгуку інтерфейсу та обробки транзакції

– Вартість впровадження (15%) - сукупна вартість володіння за перший рік

– Простота інтерфейсу (15%) - зручність для кінцевого користувача

Сума вагових коефіцієнтів: 0.3 + 0.25 + 0.15 + 0.15 + 0.15 = 1.0 (100%).

Оцінювання проводилося за 10-бальною шкалою, де 1 - найгірший показник, 10 - найкращий показник. Результати експертного оцінювання зведено в таблицю 7.1.

Таблиця 7.1 – Порівняльна оцінка привабливості програмних продуктів

Критерій	Вага (W_i)	Розроблена система	WordPress	Shopify
Захищеність даних	0,30	9	5	8
Незмінність історії	0,25	10	2	4
Швидкість роботи	0,15	9	6	8
Вартість впровадження	0,15	8	9	6
Простота інтерфейсу	0,15	7	8	10
Інтегральний показник	1,00	8,8	5,45	7,0

Розрахуємо зважену оцінку (I) для кожного продукту за формулою:

$$I = \sum (B_i \cdot W_i)$$

де B_i – бал за і-м критерієм, W_i – вага критерію.

1. Розроблена система:

$$I_{dev}(9 \cdot 0.3) + (10 \cdot 0.25) + (9 \cdot 0.15) + (8 \cdot 0.15) + (7 \cdot 0.15)$$
$$= 2.7 + 2.5 + 1.35 + 1.2 + 1.05 = 8.8$$

2. WordPress:

$$I_{dev}(5 \cdot 0.3) + (2 \cdot 0.25) + (6 \cdot 0.15) + (9 \cdot 0.15) + (8 \cdot 0.15)$$
$$= 1.5 + 0.5 + 0.9 + 1.35 + 1.2 = 5.45$$

3. Shopify:

$$I_{dev}(8 \cdot 0.3) + (4 \cdot 0.25) + (8 \cdot 0.15) + (6 \cdot 0.15) + (10 \cdot 0.15)$$
$$= 2.4 + 1.0 + 1.2 + 0.9 + 1.5 = 7.0$$

За результатами експертного оцінювання, розроблений програмний продукт отримав найвищий інтегральний показник привабливості (8.8), випередивши найближчого конкурента Shopify (7.0) та масове рішення WooCommerce (5.45). Такий результат досягнуто завдяки фокусуванню на критично важливих для цільової аудиторії параметрах - безпеці та незмінності даних.

Незначне відставання у зручності інтерфейсу компенсується унікальними функціями захисту, які відсутні у конкурентів. Це свідчить про високий ринковий потенціал розробки в сегменті захищених E-commerce систем.

7.3 Вибір методу оцінки вартості ПЗ

Оцінка вартості розробки програмного забезпечення є фундаментальним етапом економічного обґрунтування ІТ-проєкту, що дозволяє визначити собівартість продукту та спланувати цінову стратегію. Оскільки дана магістерська робота є індивідуальним проєктом із фіксованим обсягом завдань, використання комплексних моделей оцінки, таких як метод функціональних точок (FPA) або конструктивна модель (COCOMO II), є недоцільним через їхню надлишкову складність та орієнтацію на великі команди розробників. Найбільш оптимальним підходом для оцінки вартості розробки в рамках кваліфікаційної роботи обрано метод оцінки на основі трудовитрат. Цей метод базується на прямому розрахунку фактично витраченого часу на проєктування, кодування та тестування системи, що забезпечує максимальну прозорість та об'єктивність економічних показників. Собівартість продукту в даному випадку визначається як добуток загальної трудомісткості на погодинну ставку фахівця з урахуванням накладних витрат.

1. Визначення загальної трудомісткості. На основі декомпозиції робіт (аналіз вимог, проєктування архітектури, backend-розробка, frontend-розробка, тестування) загальний фонд робочого часу оцінено у 320 людино-годин, що відповідає двом місяцям повноцінної розробки.

2. Встановлення вартості нормо-години. Виходячи із середньоринкових показників оплати праці розробника рівня Junior/Middle Python Developer в Україні, тарифну ставку прийнято на рівні 400 грн/год.

3. Розрахунок прямих та накладних витрат. Підсумкова вартість буде сформована шляхом додавання до фонду оплати праці коефіцієнта накладних витрат (амортизація обладнання, електроенергія, використання ПЗ), який становить

20% від прямих витрат.

Застосування даного підходу дозволяє сформувати обґрунтовану базову вартість програмного продукту, яка стане основою для розрахунку показників ефективності та терміну окупності проєкту в наступному підрозділі.

7.4 Розрахунок економічної ефективності від впровадження реалізованого ПЗ як фактору його привабливості

Економічна ефективність ІТ-проєкту є інтегральним показником, що відображає співвідношення між витраченими ресурсами на розробку програмного продукту та фінансовими результатами від його комерційної експлуатації. Для інвестора або власника продукту цей показник виступає ключовим індикатором доцільності запуску проєкту на ринок. У даному підрозділі проведено розрахунок собівартості розробленої системи, визначено точку беззбитковості та спрогнозовано рентабельність інвестицій (ROI). Першим кроком є розрахунок повної собівартості розробки (*S*), яка базується на методі Time and Material. Вона формується з фонду оплати праці розробника та накладних витрат, таких як амортизація обладнання, комунальні послуги та ліцензії на програмне забезпечення. Виходячи з попередньо визначеної загальної трудомісткості у 320 людино-годин, вартості нормо-години на рівні 400 грн та коефіцієнта накладних витрат 1,2 (20%), розрахунок собівартості виконується наступним чином:

$$S = (320 \cdot 400) \cdot 1{,}2 = 153\,600 \text{ грн.}$$

Таким чином, обсяг початкових інвестицій, необхідних для створення першої робочої версії (MVP) системи, становить 153 600 грн.

Наступним етапом є обґрунтування ціноутворення та визначення точки беззбитковості. Враховуючи орієнтацію на B2B-сегмент, зокрема інтернет-магазини з високим середнім чеком, обрано модель монетизації через продаж безстрокової ліцензії. Ринкова вартість однієї ліцензії (*P*) встановлена на рівні 25 000 грн, що є конкурентоспроможною ціною порівняно з річними підписками на Enterprise-

тарифи SaaS-платформ.

Для оцінки ефективності інвестицій розглянуто консервативний сценарій продажів, який передбачає реалізацію однієї ліцензії на місяць, що генерує річний дохід у розмірі 300 000 грн. Чистий прибуток за перший рік складе 146 400 грн. На основі цих даних розраховується коефіцієнт рентабельності інвестицій (ROI), який становить:

$$ROI = \frac{146\,400}{153\,600} \cdot 100\% \approx 95{,}3\%$$

Період окупності проєкту, розрахований як відношення початкових інвестицій до щомісячного доходу, становить приблизно 6,2 місяці. Проведені розрахунки демонструють високу економічну ефективність проєкту. При реалізації навіть консервативного сценарію продажів проєкт повністю окупається за пів року та забезпечує рентабельність інвестицій на рівні 95,3%, що підтверджує доцільність комерціалізації розробленого програмного забезпечення.

7.5 Пропозиція алгоритму просування проєкту розробки ПЗ

Розробка ефективної стратегії просування програмного продукту на ринок є необхідною умовою його комерційного успіху, особливо в насиченому сегменті B2B-рішень для електронної комерції. Запропонований алгоритм просування розробленої системи базується на комбінованому підході, що поєднує інструменти вхідного маркетингу (Inbound Marketing) для формування експертного іміджу та прямі канали комунікації для залучення корпоративних клієнтів. Процес виведення продукту на ринок доцільно розділити на послідовні етапи, починаючи з підготовчої фази, яка включає детальний аналіз конкурентного середовища та формування унікальної торговельної пропозиції, що акцентує увагу на гібридній архітектурі та гарантіях незмінності даних, які не можуть забезпечити класичні CMS-системи.

Наступним кроком реалізації алгоритму є створення цифрової присутності продукту, що передбачає розробку спеціалізованого посадкового веб-ресурсу (Landing Page) з детальною технічною документацією, демонстрацією роботи

блокчейн-аудиту та можливістю замовлення демо-версії. Ключовим інструментом залучення трафіку на цьому етапі виступає пошукова оптимізація (SEO) за специфічними низькочастотними запитами, пов'язаними з безпекою інтернет-магазинів та захистом від інсайдерського шахрайства, а також запуск контекстної реклами у пошукових системах. Важливим елементом стратегії є контент-маркетинг, який полягає у публікації аналітичних статей та кейс-стаді на профільних ресурсах та у професійних соціальних мережах, що дозволяє сформувати довіру до розробника як до експерта у сфері кібербезпеки.

7.6 Оптимізація каналів збуту та шляхів реалізації ПЗ

Для забезпечення ефективної комерціалізації розробленого програмного продукту доцільно застосувати стратегію змішаної дистрибуції, що поєднує прямі продажі та використання партнерських мереж. Одним із найбільш перспективних напрямів є співпраця з веб-студіями та системними інтеграторами, які спеціалізуються на розробці інтернет-магазинів «під ключ». Вони можуть пропонувати розроблену систему як додатковий модуль безпеки (Add-on) для своїх клієнтів, інтегруючи захищений реєстр безпосередньо на етапі створення торгової площадки, що значно знижує бар'єри входу для кінцевого користувача.

Інший варіант оптимізації шляхів реалізації - це впровадження моделі розповсюдження SaaS (Software-as-a-Service), паралельно з продажем класичних ліцензій. У такому форматі малі підприємства отримують доступ до функціоналу блокчейн-аудиту через хмарну інфраструктуру з щомісячною оплатою, що усуває необхідність адміністрування власного серверного обладнання та налаштування складного програмного оточення. Це дозволяє масштабувати продукт на сегмент малого бізнесу, для якого критичною є мінімізація капітальних витрат на старті.

Також важливо розвивати власні прямі канали збуту, зокрема офіційний веб-ресурс із демонстраційним стендом, де потенційний замовник може у реальному часі протестувати роботу алгоритмів хешування та верифікації цілісності даних.

Наявність відкритої технічної документації, API-довідників та кейсів успішного запобігання інсайдерському шахрайству не лише підвищує рівень довіри до криптографічних механізмів системи, а й дозволяє технічним спеціалістам замовника самостійно оцінити легкість інтеграції продукту в існуючу IT-інфраструктуру.

7.7 Визначення ключових факторів успіху конкретного проєкту

Ключовими факторами успіху є стабільність роботи гібридної архітектури при пікових навантаженнях та простота інтеграції з існуючими бізнес-процесами електронної комерції. Система повинна забезпечувати високу швидкість хешування даних, не створюючи затримок при оформленні замовлень, та підтримувати можливість горизонтального масштабування. Це дозволить їй безболісно впроваджуватись як у невеликих інтернет-магазинах, так і на торговельних майданчиках із значним потоком транзакцій.

Важливим чинником є також надійність і доведена безпека - користувачі мають бути впевнені, що механізм подвійного запису дійсно унеможливлює непомітну фальсифікацію фінансової історії. Для цього необхідно реалізувати ефективну систему автоматичного аудиту цілісності реєстру та забезпечити прозору візуалізацію статусу верифікації для клієнта. Гарантія незмінності даних виступає головним ціннісним активом продукту, що формує довіру до нього.

Не менш важливою умовою успіху є якісна технічна підтримка та супровід. Наявність детальної документації та команди, яка швидко реагує на інциденти безпеки і проводить регулярні оновлення криптографічних протоколів, формує позитивний імідж продукту. І, звісно, значною конкурентною перевагою буде економічна ефективність рішення - відсутність транзакційних комісій, характерних для публічних блокчейнів, та швидка окупність інвестицій роблять систему привабливою для малого та середнього бізнесу.

8 ЗАХОДИ ЩОДО ОХОРОНИ ПРАЦІ І ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона праці є невід’ємною складовою виробничої діяльності будь-якого підприємства, незалежно від форми власності та галузевої приналежності. Забезпечення конституційного права громадян на належні, безпечні і здорові умови праці, гарантоване статтею 43 Конституції України, реалізується через комплекс правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів. Головною метою охорони праці в галузі інформаційних технологій є створення такого робочого середовища, яке сприяє збереженню здоров'я, працездатності розробників програмного забезпечення та мінімізації ризиків професійних захворювань.

Правовою основою організації охорони праці на підприємстві є Закон України Про охорону праці, Кодекс законів про працю України та відповідні державні стандарти. Згідно з чинним законодавством, роботодавець зобов'язаний створити на робочому місці в кожному структурному підрозділі умови праці відповідно до вимог нормативно-правових актів, а також забезпечити додержання вимог законодавства щодо прав працівників у галузі охорони праці. Це досягається шляхом впровадження системи управління охороною праці (СУОП), проведення регулярних інструктажів, навчання персоналу та атестації робочих місць.

Специфіка роботи програмістів та інженерів-розробників полягає у тривалій взаємодії з електронно-обчислювальними машинами (ЕОМ), що класифікується як зорова напружена робота в умовах гіподинамії та нервово-емоційного навантаження. Регулювання безпеки праці для даної категорії працівників здійснюється на основі спеціалізованих нормативних документів, зокрема НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» та ДСанПіН 3.3.2-007-98 Державні санітарні правила і

норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин.

У даному розділі магістерської дисертації проводиться комплексний аналіз умов праці у відділі розробки програмного забезпечення, штат якого складає 12 осіб, а площа приміщення - 70 м². Основними завданнями розділу є ідентифікація шкідливих та небезпечних виробничих факторів, перевірка відповідності параметрів виробничого середовища санітарно-гігієнічним нормативам, а також розробка інженерно-технічних рішень та організаційних заходів, спрямованих на оптимізацію трудового процесу та забезпечення пожежної безпеки об'єкта.

8.2 Аналіз санітарно-гігієнічних умов праці на робочому місці

Об’єктом дослідження є робоче приміщення відділу розробки програмного забезпечення, в якому, згідно з завданням, працюють 12 програмістів. Загальна площа приміщення становить 70 м². Робочі місця обладнані персональними електронно-обчислювальними машинами (ПЕОМ) із рідкокристалічними моніторами, периферійними пристроями та офісними меблями. Робота програміста належить до категорії видів діяльності, пов’язаних із високим нервово-емоційним напруженням, значним зоровим навантаженням та гіподинамією. Відповідно до ДСН 3.3.6.042-99 “Санітарні норми мікроклімату виробничих приміщень”, праця розробників програмного забезпечення відноситься до категорії важкості Іа (легкі фізичні роботи з енерговитратами до 120 ккал/год), що виконуються в положенні сидячи та не потребують систематичного фізичного напруження.

Враховуючи специфіку праці з комп’ютерною технікою, на працівників відділу можуть впливати наступні небезпечні та шкідливі виробничі фактори:

– Фізичні: підвищений рівень електромагнітних випромінювань (від системних блоків, моніторів, кабельних мереж); підвищена температура повітря робочої зони внаслідок тепловиділення від обладнання; недостатня вологість повітря; підвищений рівень шуму від систем охолодження системних блоків та

серверного обладнання; небезпека ураження електричним струмом (обладнання живиться від мережі 220 В, 50 Гц).

– Психофізіологічні: розумове перенапруження; перенапруження зорового аналізатора (робота з дисплеями); монотонність праці; статичні перевантаження опорно-рухового апарату та кистей рук.

Аналіз просторових параметрів приміщення показує, що на одне робоче місце припадає приблизно 5,83 м² площі. Згідно з вимогами ДСанПіН 3.3.2.007-98, нормативне значення становить не менше 6,0 м². Наявне незначне відхилення від норми компенсується використанням сучасних компактних РК-моніторів, що дозволяє оптимізувати робочий простір, та організацією ефективної схеми розміщення робочих столів для забезпечення вільного доступу до них. Освітлення у приміщенні є комбінованим (природне та штучне). Природне світло потрапляє через віконні прорізи, що забезпечує необхідну інсоляцію у світлий час доби. Штучне освітлення реалізоване за допомогою системи загального рівномірного освітлення, яка забезпечує нормований рівень освітленості на робочій поверхні (не менше 300–500 лк для зорових робіт високої точності).

Мікрокліматичні умови у приміщенні (температура, відносна вологість, швидкість руху повітря) підтримуються системами центрального опалення в холодний період року та кондиціювання повітря в теплий період. Джерелами шуму виступають вентилятори охолодження комп'ютерної техніки та принтери, проте їхній сумарний рівень не перевищує гранично допустимих значень для приміщень програмістів (50 дБА).

8.3 Розробка заходів з поліпшення стану охорони праці

З огляду на результати аналізу, проведеного у п. 8.2 (незначне відхилення питомої площі та значна кількість теплогенеруючого обладнання), для забезпечення нормованих параметрів мікроклімату та освітлення необхідно провести інженерні розрахунки систем вентиляції та штучного освітлення.

Розрахунок теплових надлишків. У приміщенні відділу основними джерелами тепловиділень є: співробітники (12 осіб), комп'ютерна техніка (12 ПЕОМ + периферія), система штучного освітлення та сонячна радіація через вікна. Сумарні надлишки явного тепла розраховуються за формулою:

$$Q_{\text{надл}} = Q_{\text{надл}} + Q_{\text{обл}} + Q_{\text{осв}} + Q_{\text{сон}}, \text{Вт}$$

Таблиця 8.1 — Розрахунок надлишків явного тепла у приміщенні

Джерело тепловиділень	Розрахункова Формула	Вихідні дані та хід розрахунку	Результат (Q_i), Вт
Люди	$Q_{\text{люд}} = n \cdot q_{\text{л}}$	$12 \cdot 90$	1080
Обладнання	$Q_{\text{обл}} = n \cdot q_{\text{пк}} \cdot \beta$	$12 \cdot 300 \cdot 0{,}5$	1800
Штучне освітелння	$Q_{\text{осв}} = S \cdot q_{\text{осв}} \cdot \eta$	$70 \cdot 20 \cdot 0{,}55$	770
Сонячна радіація	$Q_{\text{сон}} = S_{\text{вік}} \cdot q_{\text{скл}} \cdot k$	$12 \cdot 150 \cdot 0{,}6$	1080
Всього	$\sum Q_i$	Сумарні тепловиділення	4730

Для забезпечення допустимих метеорологічних умов у робочій зоні необхідно видаляти надлишки тепла за допомогою системи штучної вентиляції. Розрахунок необхідного повітрообміну проводиться за двома умовами: за санітарними нормами подачі свіжого повітря на одну людину та за умовою асиміляції надлишків тепла. До впровадження приймається більше з двох отриманих значень.

Таблиця 8.2 – Розрахунок необхідного повітрообміну

Умова розрахунку	Формула розрахунку	Вихідні дані та розрахунок	Результат
За санітарними нормами	$L_{\text{сан}} = N \cdot L_{\text{норм}}$	$12 \cdot 30$	360
За надлишками тепла	$L_{\text{сан}} = \frac{3{,}6 \cdot Q_{\text{надл}}}{c \cdot p \cdot (t_{\text{вид}} - t_{\text{прит}})}$	$\frac{3{,}6 \cdot 4730}{1{,}005 \cdot 1{,}2 \cdot 5}$	2824

Прийняте значення	$L = \max(L_{\text{сан}}, L_{\text{тепл}})$	Максимальне з розрахованих	2824

На основі розрахунків, для приміщення об'ємом 210 м³ (70 м² × 3 м) необхідна припливно-витяжна система вентиляції продуктивністю не менше 2824 м³/год, що забезпечить кратність повітрообміну на рівні 13,5 разів на годину.

Наступним етапом є розрахунок системи штучного освітлення. Оскільки природного світла в осінньо-зимовий період недостатньо для виконання зорових робіт високої точності, проектується система загального рівномірного освітлення. Розрахунок кількості світильників виконується методом коефіцієнта використання світлового потоку. Для освітлення обрано растрові LED-світильники (світловий потік однієї лампи 2500 лм, 2 лампи у світильнику).

Таблиця 8.3 – Розрахунок системи штучного освітлення

Параметр	Позначення	Значення
Площа приміщення	S	70 м^2
Нормована освітленість	$E_{\text{н}}$	400 лк
Коефіцієнт запасу	k	1.4
Коефіцієнт нерівномірності	Z	1,1
Коефіцієнт використання потоку	η	0,55
Світловий потік однієї лампи	$F_{\text{н}}$	2500 лм
Кількість ламп у світильнику	m	2 шт.
Розрахункова кількість ламп	$N = \frac{E \cdot S \cdot k \cdot Z}{F_{\text{л}} \cdot \eta}$	32 шт.
Необхідна кількість світильників	$N_{\text{св}} = N/m$	16 шт.

Таким чином, для забезпечення нормованого рівня освітленості необхідно встановити 16 світильників, розміщених у 4 ряди.

8.4 Техніка безпеки та протипожежна профілактика

Забезпечення електробезпеки у відділі розробки здійснюється відповідно до вимог НПАОП 40.1-1.21-98. Оскільки комп'ютерна техніка живиться від мережі змінного струму напругою 220 В, існує ризик ураження електричним струмом у разі пошкодження ізоляції. Для запобігання електротравматизму передбачено:

– Захисне заземлення (занулення) всіх металевих неструмоведучих частин обладнання;

– Використання розеток із захисним контактом (євростандарт);

– Недопущення прокладання кабелів живлення через проходи;

– Регулярний візуальний огляд цілісності ізоляції проводів.

Пожежна безпека об’єкта забезпечується згідно з Кодексом цивільного захисту України та НАПБ А.01.001-2014. Приміщення з комп'ютерною технікою відноситься до категорії В за вибухопожежною небезпекою. Для протипожежного захисту передбачено:

– Оснащення приміщення первинними засобами пожежогасіння (вуглекислотні вогнегасники типу ВВК-3, які не пошкоджують електроніку);

– Встановлення автоматичної пожежної сигналізації з димовими датчиками;

– Наявність плану евакуації людей та матеріальних цінностей.

8.5 Розрахункова частина

Розрахунок часу евакуації з приміщення. Відповідно до індивідуального завдання, необхідно виконати перевірку своєчасності евакуації персоналу у випадку пожежі. Розрахунок проводиться за методикою ГОСТ 12.1.004-91.

Вихідні дані:

– Розташування приміщення: 2-й поверх.

– Габаритні розміри: довжина 36 м, ширина 10 м.

– Площа приміщення: S = 360 м².

– Кількість людей: N = 60 осіб.

– Ширина евакуаційного виходу (дверей): δ = 1,2 м.

Хід розрахунку:

Визначення щільності людського потоку D. Щільність потоку визначається як відношення сумарної площі горизонтальної проекції людей до площі шляху евакуації. Площа проекції однієї людини приймається f = 0,125 м².

$$D = \frac{N \cdot f}{S} = \frac{60 \cdot 0{,}125}{360} = \frac{7{,}5}{360} = 0{,}02$$

Визначення швидкості та інтенсивності руху.

Для щільності потоку D = 0,02 (згідно з табличними даними ГОСТ 12.1.004-91) параметри руху становлять:

– Швидкість руху: V = 100 м/хв.

– Інтенсивність руху: q = 2,0 м/хв.

Розрахунок часу евакуації (t_p). Розрахунковий час евакуації складається з часу руху від найвіддаленішої точки до дверей ($t_{\text{рух}}$) та часу проходження через дверний отвір ($t_{\text{лв}}$). Довжина шляху евакуації (L) приймається як діагональ приміщення:

$$L = \sqrt{36^2 + 10^2} \approx 37{,}4 \text{ м}$$

Час руху до виходу:

$$t_{\text{рух}} = \frac{L}{V} = \frac{37{,}4}{100} = 0{,}374 \text{ хв}$$

Пропускна здатність дверей при ширині 1,2 м становить приблизно 60 осіб/хв. Час виходу 60 осіб:

$$t_{\text{дв}} = \frac{N}{60} = \frac{60}{60} = 1{,}0 \text{ хв}$$

Загальний час евакуації:

$$t_p = t_{\text{рух}} + t_{\text{дв}} = 0{,}374 + 1{,}0 \approx 1{,}37 \text{ хв}$$

Розрахунковий час евакуації (1,37 хв) не перевищує допустимого нормативного часу для будівель даного ступеня вогнестійкості (як правило, 2-3 хв).

Умови безпечної евакуації виконуються.

У розділі виконано аналіз умов праці у відділі розробки програмного забезпечення. Встановлено, що для компенсації тепловиділень від 12 одиниць обчислювальної техніки та персоналу необхідна система вентиляції продуктивністю 2824 м³/год. Для забезпечення нормованої освітленості робочих місць (400 лк) розраховано систему штучного освітлення, що складається з 16 растрових світильників. Розрахунок евакуації для приміщення площею 360 м² з чисельністю персоналу 60 осіб показав, що при ширині виходу 1,2 м повна евакуація людей займає 1,37 хвилини, що відповідає нормам пожежної безпеки. Запропоновані технічні та організаційні заходи дозволяють створити безпечні умови праці, що відповідають чинним нормативно-правовим актам України.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для управління онлайн-замовленнями з використанням інтегрованих механізмів блокчейн-захисту транзакцій. В межах України в недостатній мірі представлені доступні вітчизняні розробки в цій області, які б дозволяли малому та середньому бізнесу використовувати переваги технології розподіленого реєстру без значних капіталовкладень. У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів забезпечення цілісності та незмінності даних в системах електронної комерції. Рішення даного завдання полягало у вирішенні наступних задач: – Був проведений огляд існуючих систем управління замовленнями та аналіз їх вразливостей до інсайдерських атак. – Досліджена гібридна архітектура зберігання даних, що поєднує реляційні бази даних та технологію ланцюжка блоків. – На основі отриманих результатів досліджень створена програмна реалізація веб-орієнтованої системи із захищеним реєстром транзакцій.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми хешування та перехресної верифікації дозволяють успішно вирішувати завдання захисту фінансової історії від несанкціонованої модифікації. Проведено аналіз предметної галузі, в ході якого були виявлені об'єкти (замовлення, транзакції, блоки), взаємодія яких носить істотний характер для функціональної діяльності системи, і їхні основні характеристики; побудований алгоритм фіксації даних і вибране середовище розробки. Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує глибоких знань у галузі

криптографії від кінцевого користувача. При створенні програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних веб-систем та забезпечує модульність коду. Програма реалізована на мові високого рівня Python. Дана мова програмування та її бібліотеки дозволяють найбільш ефективно обробляти криптографічні операції та взаємодіяти з веб-інтерфейсом. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як наслідок, зменшити витрати на його створення. Запропоноване програмне забезпечення ділиться на серверну частину (Backend), що відповідає за логіку блокчейну, та клієнтську частину (Frontend), що реалізує відображення даних. Програма призначена для виконання під управлінням операційних систем сімейства Linux (Ubuntu/Debian) або Windows 10/11. Даються необхідні рекомендації з установки та налаштування розробленого програмного забезпечення на серверному обладнанні. Для підвищення рівня безпеки та забезпечення автентичності транзакцій запропоновано застосовувати алгоритм цифрового підпису на еліптичних кривих (ECDSA). В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення, масштабування та застосування у різних галузях електронної комерції та логістики. Проведено маркетингове та економічне обґрунтування ІТ-проєкту, що дозволило визначити ключові фактори успіху даного проекту та підтвердити його інвестиційну привабливість.

СПИСОК ЛІТЕРАТУРИ

1. Антонопулос А. «Mastering Bitcoin: Unlocking Digital Cryptocurrencies». O'Reilly Media, 2017. – 416 p.

2. Накамото С. «Bitcoin: A Peer-to-Peer Electronic Cash System». Whitepaper, 2008. – 9 p.

3. Закон України «Про віртуальні активи» від 17.02.2022 № 2074-IX // Відомості Верховної Ради України. – 2022.

4. Grinberg M. «Flask Web Development: Developing Web Applications with Python». O'Reilly Media, 2018. – 325 p.

5. Гнатюк С.О., Кінзєрський В.М. Перспективи впровадження технології блокчейн у системи електронної комерції // Сучасний захист інформації. – 2021. – № 2 (46). – С. 34-41.

6. Swan M. «Blockchain: Blueprint for a New Economy». O'Reilly Media, 2015. – 152 p.

7. Лисенко В.С., Бобровнікова К.Ю. Аналіз методів захисту веб-додатків на мові Python // Збірник праць молодих науковців ЦНТУ. – Вип. 15. – Кропивницький: ЦНТУ, 2025.

8. Lutz M. «Learning Python», 5th Edition. O'Reilly Media, 2013. – 1600 p.

9. Narayanan A., Bonneau J., Felten E. «Bitcoin and Cryptocurrency Technologies: A Comprehensive Introduction». Princeton University Press, 2016. – 336 p.

10. Коваленко А.І. Використання мікросервісної архітектури у високонавантажених E-commerce системах // Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка. – 2023. – № 18. – С. 12-19.

11. Buterin V. «A Next-Generation Smart Contract and Decentralized Application Platform». Ethereum Whitepaper, 2014.

12. Tapscott D., Tapscott A. «Blockchain Revolution: How the Technology

Behind Bitcoin Is Changing Money, Business, and the World». Penguin, 2016. – 368 p.

13. Закон України «Про електронну комерцію» від 03.09.2015 № 675-VIII // Відомості Верховної Ради України.

14. Смірнов О.А., Смірнова Т.Р. «Проектування захищених інформаційних систем». Кропивницький: ЦНТУ, 2024. – 210 с.

15. Kleppmann M. «Designing Data-Intensive Applications: The Big Ideas Behind Reliable, Scalable, and Maintainable Systems». O'Reilly Media, 2017. – 616 p.

16. ISO/IEC 27001:2022. «Information security, cybersecurity and privacy protection — Information security management systems — Requirements». International Organization for Standardization, 2022.

17. Потапова К.О. Порівняльний аналіз SQL та NoSQL баз даних для зберігання фінансових транзакцій.

18. Richardson L., Amundsen M. «RESTful Web APIs». O'Reilly Media, 2013. – 404 p.

19. Python Software Foundation. «Python 3.12.1 Documentation». [Електронний ресурс]. – Режим доступу: https://docs.python.org/3/

20. Pallets Projects. «Flask Documentation (Version 3.0.x)». [Електронний ресурс]. – Режим доступу: https://flask.palletsprojects.com/

21. Шевченко І.В. Механізми консенсусу в розподілених мережах: від Proof-of-Work до Proof-of-Stake // Наукові праці ВНТУ. – 2022. – № 3. – С. 55-62.

22. Beazley D. «Python Distilled». Addison-Wesley Professional, 2021. – 352 p.

23. Mougayar W. «The Business Blockchain: Promise, Practice, and Application of the Next Internet Technology». Wiley, 2016. – 208 p.

24. PostgreSQL Global Development Group. «PostgreSQL 16.1 Documentation». [Електронний ресурс]. – Режим доступу: https://www.postgresql.org/docs/

25. Мельник П.В. Оптимізація запитів у реляційних базах даних для веб-застосунків // Системний аналіз та інформаційні технології: Матеріали XXV Міжнародної наукової конференції (Київ, 20-22 травня 2024 р.). – Київ: ННК

«ІПСА», 2024. – С. 112-115.

26. NIST Special Publication 800-63B. «Digital Identity Guidelines: Authentication and Lifecycle Management». National Institute of Standards and Technology, 2017.

27. Bashir I. «Mastering Blockchain: Distributed ledger technology, decentralization, and smart contracts explained». Packt Publishing, 2018. – 542 p.

28. Ткачук В.М. «Технології веб-програмування: навчальний посібник». Івано-Франківськ: Прикарпатський національний університет ім. В. Стефаника, 2021. – 240 с.

29. MDN Web Docs. «HTTP response status codes». [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/

30. Савченко В.М., Петренко О.С. Реалізація гібридної архітектури зберігання даних у блокчейн-системах // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2024.

31. Van Rossum G., Drake F.L. «The Python Language Reference». Python Software Foundation, 2023.

32. SQLAlchemy developers. «SQLAlchemy 2.0 Documentation». [Електронний ресурс]. – Режим доступу: https://www.sqlalchemy.org/

33. Кучерявий А.П. Інтеграція платіжних шлюзів у веб-додатки на базі Python // Інженерія програмного забезпечення. – 2023. – № 1. – С. 23-29.

34. W3C Recommendation. «HTML5.2: A vocabulary and associated APIs for HTML and XHTML». World Wide Web Consortium, 2017.

35. Bootstrap Team. «Bootstrap 5.3 Documentation». [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/

36. Гаврилюк М.С. Забезпечення цілісності даних у системах електронної комерції // Кібербезпека: освіта, наука, техніка. – 2022. – Т. 4, № 16. – С. 88-96.

37. Drescher D. «Blockchain Basics: A Non-Technical Introduction in 25 Steps». Apress, 2017. – 255 p.

38. Закон України «Про захист персональних даних» від 01.06.2010 № 2297-

VI // Відомості Верховної Ради України.

39. Miguel Grinberg «The Flask Mega-Tutorial». [Електронний ресурс]. – Режим доступу: https://blog.miguelgrinberg.com/

40. Василенко М.Ю. «Алгоритми та структури даних: Python». Київ: Видавничий дім «Києво-Могилянська академія», 2020. – 312 с.

41. Percival H. «Test-Driven Development with Python». O'Reilly Media, 2017. – 600 p.

42. Jinja2 Team. «Jinja2 Documentation (Designer Documentation)». [Електронний ресурс]. – Режим доступу: https://jinja.palletsprojects.com/

43. Бойко І.І. Методи протидії атакам типу SQL Injection та XSS у веб-середовищі

44. Franco P. «Understanding Bitcoin: Cryptography, Engineering and Economics». Wiley, 2014. – 288 p.

45. Docker Inc. «Docker Documentation». [Електронний ресурс]. – Режим доступу: https://docs.docker.com/

46. Романюк О.В. Архітектурні патерни проектування веб-застосунків MVC та MVT: порівняльний аналіз

47. OWASP Foundation. «OWASP Top 10: 2021 The Ten Most Critical Web Application Security Risks». [Електронний ресурс]. – Режим доступу: https://owasp.org/Top10/

48. Харченко В.С. «Безпека інформаційних технологій та кібербезпека». Харків: Нац. аерокосм. ун-т ім. М. Є. Жуковського «ХАІ», 2020. – 450 с.

49. Github Inc. «GitHub Actions Documentation». [Електронний ресурс]. – Режим доступу: https://docs.github.com/

50. Черновол В.Є. Дослідження та програмна реалізація веб-орієнтованої системи управління онлайн-замовленнями з використанням інтегрованих блокчейн-механізмів захисту транзакцій // Збірник праць молодих науковців ЦНТУ. – Вип. 15. – Кропивницький: ЦНТУ, 2025.