Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Бакалаврська кваліфікаційна робота

на тему: «Розробка програмного засобу для управління фінансуванням станцій технічного обслуговування»

Виконав: студент 4 курсу
групи 2ПІ-21б
спеціальності
121 - Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Франчук О.В.
(прізвище та ініціали)
Керівник: к.т.н., доц. каф. ПЗ Хошаба О.М.
(прізвище та ініціали)
Рецензент: к.т.н., доц. каф. ЗІ Дудатьєв А.В.
(прізвище та ініціали)

Допущено до захисту
Зав. кафедри ПЗ О.Н.Романюк
«09» 06 2025 р.

ВНТУ – 2025

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

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ Романюк О. Н.
21 березня 2025 року

З А В Д А Н Н Я

НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Франчуку Олександру Вадимовичу

1. Тема роботи - «Розробка програмного засобу для управління фінансуванням станцій технічного обслуговування»

Керівник роботи: Хошаба Олександр Мирославович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від 20 березня 2025 р. № 97.

2. Строк подання студентом роботи 2 червня 2025 р.

3. Вихідні дані до роботи: кількість параметрів фінансової моделі - до 25; кількість основних функціональних вимог до програмного забезпечення - до 30; обсяг бази даних транзакцій - до 100000 записів; швидкість обробки фінансових записів - не менш ніж 200 за секунду; очікуване зниження витрат - до 15% при впровадженні системи; підтримка інтеграції з ERP-системами (BAS, SAP) - обов'язкова.

4. Зміст розрахунково-пояснювальної записки: вступ, аналіз предметної галузі та специфіки фінансування СТО, огляд існуючих програмних рішень та їх порівняльний аналіз, реалізація основних модулів системи, тестування, моделювання сценаріїв та аналіз ефективності, використання та супроводу програмного засобу, висновки, перелік посилань, додатки.

5. Перелік графічного матеріалу: мета та задачі роботи, формування методики управління фінансуванням із застосуванням блокчейн, архітектура та технічні вимоги до програмного засобу, блок-схема алгоритмів роботи програмної системи, тестування програмної системи, висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Хошаба О.М., к.т.н., доцент кафедри ПЗ	24.03.25	01.06.25

7. Дата видачі завдання 24 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Аналіз предметної галузі та специфіки фінансування СТО	25.03.2025-03.04.2025	вик
2	Архітектура та технічні вимоги до програмного засобу	04.04.2025-14.04.2025	вик
3	Реалізація основних та додаткових модулів системи	15.04.2025-05.05.2025	вик
4	Тестування основних та додаткових модулів системи	06.05.2025-19.05.2025	вик
5	Оформлення матеріалів до захисту БКР	20.05.2025-30.05.2025	вик

Студент _____ Франчук О.В.
(підпис) (прізвище та ініціали)

Керівник бакалаврської кваліфікаційної роботи _____ Хошаба О.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

УДК 004.4:005.334:656.075

Франчук О.В. Розробка програмного засобу для управління фінансуванням станцій технічного обслуговування: бакалаврська кваліфікаційна робота зі спеціальності 121 Інженерія програмного забезпечення, освітня програма – інженерія програмного забезпечення. Вінниця: ВНТУ, 2025. 139 С.

На укр. мові. Бібліогр.: 37 назв; рис.: 24; табл. 7.

У роботі розглянуто питання розробки програмного засобу для управління фінансуванням станцій технічного обслуговування. Проведено аналіз методів і моделей фінансового контролінгу, управління грошовими потоками, бюджетування та впровадження блокчейн-механізмів. Обґрунтовано архітектуру та реалізовано функціональні компоненти програмного забезпечення, включаючи автоматизацію обліку, інтеграцію з IoT-пристроями та використання смарт-контрактів.

ANNOTATION

Franchuk O.V. Development of a software tool for managing the financing of service stations: bachelor's thesis on the specialty 121 Software engineering, educational program - software engineering. Vinnytsia: VNTU, 2025. 139 with.

In Ukrainian language. Bibliographer: 37 titles; Fig.: 24; table 7.

The paper considers developing a software tool for managing service station financing. It analyses methods and models of financial control, cash flow management, budgeting, and the implementation of blockchain mechanisms. The architecture is justified, and functional components of the software are implemented, including accounting automation, integration with IoT devices, and the use of smart contracts.

ЗМІСТ

ВСТУП 3
1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ 7
1.1 Аналіз предметної галузі та особливості управління фінансуванням станцій технічного обслуговування 7
1.2 Порівняльний аналіз програмних інструментів для управління фінансуванням комерційних організацій 14
1.3 Постановка задач дослідження 29
2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ 31
2.1 Запропонований метод комплексного фінансового контролінгу 31
2.2 Розробка функціональних вимог до програмного засобу 38
2. 3 Проектування основних та додаткових модулів програмного засобу 44
3 РОЗРОБКА МОДУЛІВ ПРОГРАМНОГО ЗАСОБУ 52
3.1 Основні принципи розробки програмного засобу на основі блокчейну 52
3.2 Основні принципи роботи модуля управління замовленнями на основі блокчейн 68
3.3 Основні принципи роботи модуля інтеграції з постачальниками програмного засобу 75
4 ТЕСТУВАННЯ МОДУЛІВ ПРОГРАМНОГО ДОДАТКУ 81
4.1 Особливості тестування мікросервісної архітектури програмного засобу 81
4.2 Тестування програмного модуля управління замовленнями 87
4.3 Тестування програмного модуля інтеграції з постачальниками програмного засобу 93
ВИСНОВКИ 99
ПЕРЕЛІК ПОСИЛАНЬ 102
ДОДАТОК А – Технічне завдання 105
ДОДАТОК Б – Протокол перевірки клаліфікаційної роботи 108
ДОДАТОК В – Основні компоненти фронтенду для модулля створення замовлення 109
ДОДАТОК Г – Графічна частина 129

ВСТУП

Обґрунтування вибору теми дослідження. Сучасний стан розвитку паливно-технічної галузі в Україні та світі зумовлює високий попит на ефективні методи управління фінансуванням станцій технічного обслуговування (СТО). Зростання кількості автомобілів, ускладнення технічних процесів ремонту та підвищення конкуренції серед СТО потребують впровадження сучасних інформаційних технологій для автоматизації управлінських рішень, особливо в частині фінансового моніторингу, бюджетування та оптимізації витрат.

Нині більшість СТО все ще використовують застарілі або розрізнені підходи до ведення обліку та фінансового контролю, що призводить до втрати прибутків, зниження рентабельності та неможливості ефективно планувати діяльність. Як показує аналіз, впровадження програмного забезпечення для автоматизації фінансового управління дозволяє підвищити прозорість операцій, мінімізувати витрати, а також приймати своєчасні управлінські рішення.

Тому, актуальність теми полягає у необхідності комплексного вирішення проблем управління фінансуванням СТО шляхом створення спеціалізованого програмного засобу, що враховує специфіку галузі, підтримує сучасні цифрові інструменти (ERP, блокчейн, IoT) та забезпечує інтеграцію з існуючими бізнес-процесами. Особливу цінність має впровадження таких функцій, як моніторинг витрат у реальному часі, автоматизоване бюджетування, аналітика прибутковості послуг та підтримка предиктивного планування.

Основні проблеми дослідження полягають у відсутності адаптованих рішень для українських умов, складності інтеграції з уже існуючими інформаційними системами, високій вартості ліцензованого ПЗ та обмеженій готовності персоналу до роботи з новими інструментами. Також актуальною є проблема захисту фінансових даних, а отже, потребує впровадження безпечних механізмів на базі технологій блокчейн.

Таким чином, існуючі рішення на ринку, такі як SAP ERP, SmartEAM, BAS ERP, Octane Systems або PDI Enterprise, мають широку функціональність, але є або надто дорогими, або не адаптованими під потреби малих і середніх СТО. Водночас, поєднання концепції фінансового контролінгу, управління грошовими потоками,

бюджетування та предиктивної аналітики в одному програмному середовищі залишається малодослідженим і слабо реалізованим на практиці. Отже, розробка власного програмного засобу для управління фінансуванням СТО є актуальним, затребуваним та інноваційно значущим завданням.

Зв'язок роботи з науковими програмами, планами, темами. Робота виконувалась згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання. Метою роботи є підвищення ефективності управління фінансуванням станцій технічного обслуговування за рахунок розробки та впровадження програмного засобу, що забезпечує автоматизацію процесів бюджетування, фінансового моніторингу, аналітики витрат та інтеграцію з блокчейн-механізмами забезпечення прозорості фінансових операцій.

Відповідно до поставленої мети виконані наступні задачі:

- виконати аналіз предметної галузі та особливості управління фінансуванням станцій технічного обслуговування;
- виконати порівняльний аналіз програмних інструментів для управління фінансуванням комерційних організацій
- запропонувати метод комплексного фінансового контролінгу;
- розробити функціональні вимоги до програмного засобу;
- виконати проектування основних та додаткових модулів програмного засобу;
- визначити та розробити основні принципи програмного засобу на основі блокчейну;
- визначити та розробити основні принципи роботи модуля управління замовленнями на основі блокчейн;
- визначити та розробити основні принципи роботи модуля інтеграції з постачальниками програмного засобу;
- визначити та розробити особливості тестування мікросервісної архітектури програмного засобу;
- виконати тестування програмного модуля управління замовленнями;
- виконати тестування програмного модуля інтеграції з постачальниками програмного засобу.

Об’єктом дослідження є процес управління фінансуванням станцій технічного обслуговування в умовах цифровізації економіки та високої конкуренції в галузі сервісного обслуговування.

Предметом дослідження є методи, моделі та програмні засоби, що дозволяють автоматизувати та оптимізувати управління фінансами СТО, зокрема в частині обліку доходів і витрат, моніторингу грошових потоків, бюджетного планування та інтеграції аналітичних інструментів.

Методи дослідження. У процесі дослідження використовувались методи об’єктно-орієнтованого програмування для створення архітектури програмного засобу; методи системного аналізу для формалізації фінансових потоків СТО; елементи фінансової математики для побудови моделей оцінки прибутковості; машинне навчання для реалізації предиктивної аналітики (наприклад, сезонні коливання попиту); блокчейн-технології для забезпечення прозорості фінансових транзакцій та захисту даних.

Новизна отриманих результатів:

1. Запропоновано метод комплексного фінансового контролінгу для СТО, особливість якого полягає в інтеграції даних про доходи, витрати та операційні показники у режимі реального часу на основі блокчейн мереж, що дає можливість оперативно виявляти фінансові відхилення і приймати обґрунтовані управлінські рішення.

2. Запропоновано модель бюджетного планування діяльності СТО, яка враховує сезонні коливання попиту на послуги та витрати на обслуговування обладнання, що дозволило підвищити точність фінансового планування та ефективність розподілу ресурсів на 12%.

3. Подальшого розвитку отримала інформаційна система управління фінансами СТО, у якій, на відміну від існуючих, реалізовано інтеграцію з системами управління замовленнями та запасами на основі блокчейн мережі, що дозволило забезпечити актуальність даних і підвищити точність фінансового аналізу на 17%.

Практична цінність отриманих результатів. Практичне значення роботи полягає у розробці, тестуванні та впровадженні програмного засобу, що на основі

сучасних моделей фінансового менеджменту забезпечує комплексне управління фінансовими ресурсами станцій технічного обслуговування. Запропонована система дозволяє автоматизувати процеси планування бюджету, контролю фінансових потоків, аналізу рентабельності послуг та формування фінансової звітності. Практична цінність розробленого програмного засобу визначається можливістю оперативного отримання точної фінансової інформації для прийняття обґрунтованих управлінських рішень, що сприяє підвищенню економічної ефективності діяльності СТО, оптимізації витрат та збільшенню прибутковості підприємства. Додатковою перевагою системи є можливість її адаптації до різних масштабів бізнесу – від невеликих станцій технічного обслуговування до розгалужених мереж автосервісу.

Особистий внесок здобувача. Усі наукові результати, викладені у бакалаврській кваліфікаційній роботі, отримані автором особисто. Автору належать такі результати: постановка задачі дослідження; розроблені алгоритми на основі UML-діаграм; модулі клієнтської та серверної частини, тестування.

Апробація результатів роботи. Результати роботи були представленні на XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів.

Публікації. Результати роботи були опубліковані в матеріалах XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. «Стан, досягнення і перспективи інформаційних систем і технологій». Одеса, 17 - 18 квітня 2025 р. [1].

1 АНАЛІЗ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Аналіз предметної галузі та особливості управління фінансуванням станцій технічного обслуговування

Управління фінансуванням станцій технічного обслуговування (СТО) та інших комерційних закладів є ключовим аспектом забезпечення їхньої ефективної діяльності та стійкості [2-4]. Для цього використовуються різноманітні методи (рис. 1.1) та моделі (рис. 1.2), що допомагають оптимізувати фінансові потоки, знижувати витрати та підвищувати прибутковість.

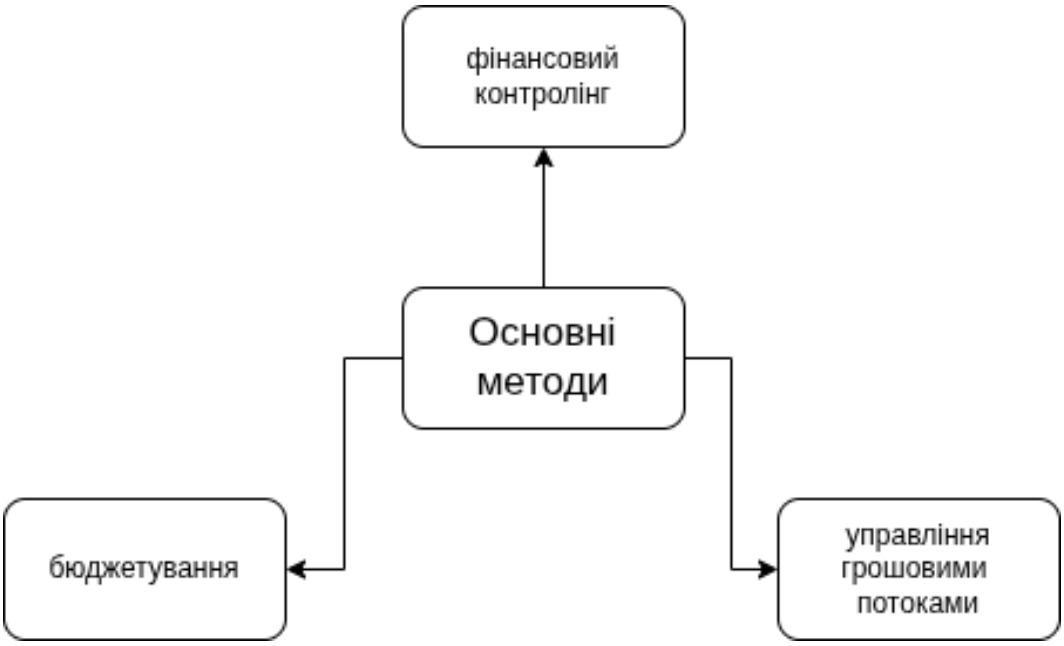

Рисунок 1.1 - Методи управління фінансуванням станцій технічного обслуговування

До основний методів управління фінансуванням відноситься фінансовий контролінг, бюджетування та управління грошовими потоками (рис. 1.1).

Фінансовий контролінг передбачає постійний моніторинг фінансових показників підприємства, аналіз відхилень від планових показників та розробку коригувальних дій [6]. Він допомагає вчасно виявляти фінансові ризики та приймати обґрунтовані управлінські рішення. Метод фінансового контролінгу має низку особливостей, які роблять його ефективним інструментом управління фінансами на підприємстві.

Метод фінансового контролінгу є комплексним інструментом, який дозволяє ефективно управляти фінансами, мінімізувати ризики та досягати стратегічних цілей підприємства. Його використання особливо актуальне для компаній, які прагнуть забезпечити фінансову прозорість і стабільність у конкурентному середовищі.

Бюджетування дозволяє планувати доходи та витрати підприємства на певний період, що сприяє ефективному розподілу ресурсів та контролю за їх використанням [7]. Метод бюджетування є одним із ключових інструментів фінансового управління підприємством. Його використання має низку особливостей, які дозволяють підприємствам ефективно планувати, контролювати та аналізувати фінансові ресурси. Він забезпечує підприємствам інструменти для ефективного планування, контролю та аналізу фінансових ресурсів. Його особливості дозволяють досягти гнучкості, адаптивності та узгодженості між цілями і ресурсами підприємства, що є критично важливим у динамічному бізнес-середовищі.

Насамперед, метод бюджетування є універсальним і гнучким інструментом фінансового управління, який дозволяє досягти оптимального використання ресурсів, мінімізувати ризики та забезпечити реалізацію стратегічних цілей підприємства. Його особливості, такі як багаторівневий підхід до планування, інтеграція з іншими функціями управління та використання сучасних технологій, роблять цей метод ефективним у сучасному динамічному бізнес-середовищі.

Управління грошовими потоками забезпечує своєчасне надходження та раціональне використання грошових коштів, що є критичним для підтримки ліквідності та платоспроможності підприємства [8].

Метод управління грошовими потоками є важливим компонентом фінансового управління, який дозволяє забезпечити ефективний рух грошових коштів у рамках діяльності підприємства. Використання цього методу характеризується низкою особливостей, що сприяють забезпеченню фінансової стійкості, ліквідності та платоспроможності організації.

Метод управління грошовими потоками є багатофункціональним інструментом, який забезпечує підтримання фінансової стабільності, мінімізацію

ризиків та підвищення ефективності використання ресурсів. Його особливості, зокрема інтеграція з іншими методами фінансового управління, застосування сучасних інформаційних технологій та орієнтація на прогнозування, роблять цей метод незамінним у сучасному бізнес-середовищі.

До основних моделей управління фінансуванням відносяться економіко-математичні моделі, моделі управління фінансовими ризиками, моделі фінансового планування та прогнозування (рис. 1.2).

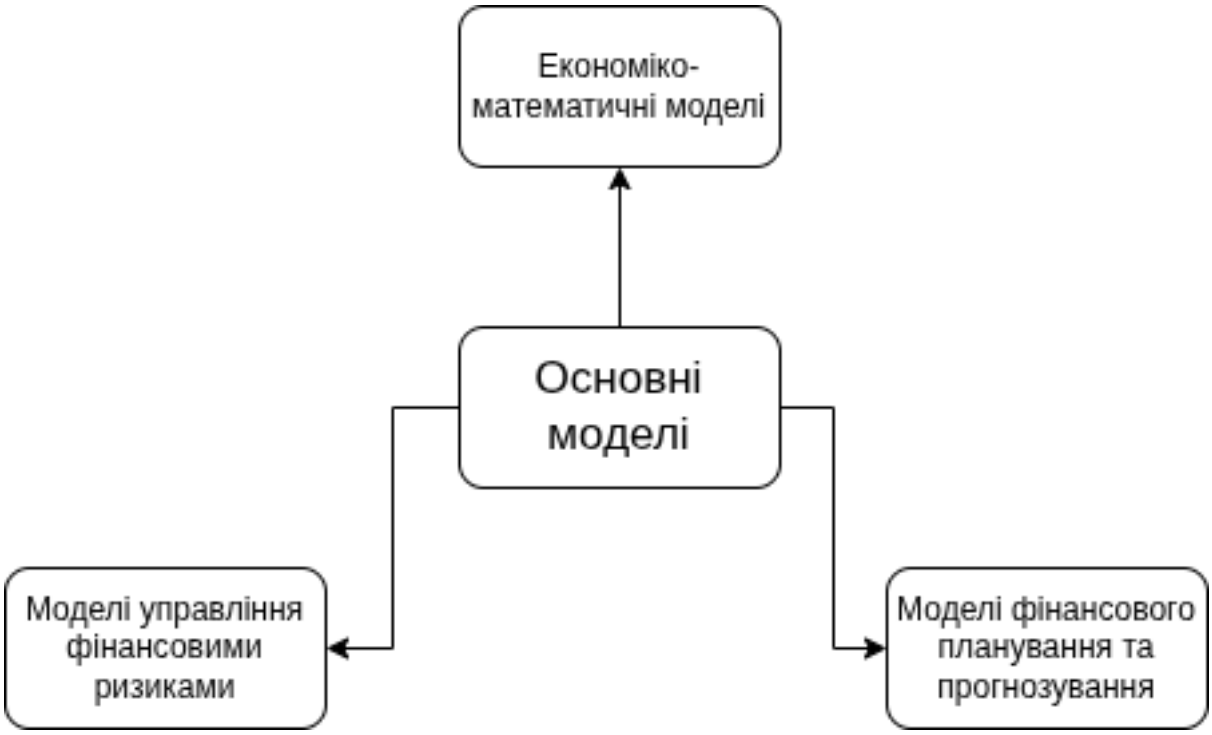

Рисунок 1.2 - Моделі управління фінансуванням станцій технічного обслуговування

Економіко-математичні моделі є потужним інструментом аналізу та прийняття рішень у фінансовому та економічному управлінні. Їх використання має низку особливостей, які визначають їхню ефективність та специфіку застосування в різних галузях. Ці моделі дозволяють формалізувати процеси, виявляти взаємозв’язки між ключовими показниками та оптимізувати управлінські рішення. Вони ґрунтуються на формалізації складних економічних явищ у вигляді математичних рівнянь, нерівностей чи функцій. Це дозволяє точно описувати взаємозв’язки між змінними та спрощувати аналіз багатофакторних систем. Однією

з головних особливостей є спрямованість на пошук оптимальних рішень у контексті заданих цілей. Наприклад, моделі лінійного програмування використовуються для оптимального розподілу ресурсів, моделі динамічного програмування - для багатокрокових рішень, а стохастичні моделі - для роботи в умовах невизначеності.

Моделі управління фінансовими ризиками є важливим інструментом, що дозволяє ідентифікувати, оцінювати, контролювати та мінімізувати фінансові ризики, які впливають на діяльність підприємств (табл. 1.1). Їх використання має низку специфічних особливостей, які зумовлюють ефективність і доцільність застосування цих моделей у різних економічних умовах. Вони зосереджені на визначенні потенційних джерел ризиків, таких як валютні коливання, зміни процентних ставок, кредитні ризики, а також ризики ліквідності. Цей етап є критично важливим, оскільки без точного визначення ризиків неможливо розробити ефективні заходи для їхнього мінімізації.

Моделі управління фінансовими ризиками є незамінними інструментами у забезпеченні стабільності та ефективності функціонування підприємств у сучасному ринковому середовищі. Їх особливості, зокрема орієнтація на ідентифікацію та оцінку ризиків, інтеграція з іншими управлінськими системами, автоматизація та використання інструментів хеджування, сприяють зниженню фінансових втрат і підтриманню стабільного розвитку організацій.

Моделі фінансового планування та прогнозування є важливими інструментами, що забезпечують ефективне управління фінансовими ресурсами підприємства. Їх використання спрямоване на формування обґрунтованих фінансових планів та прогнозів, які враховують можливі зміни у зовнішньому та внутрішньому середовищі. Ці моделі мають низку особливостей, які визначають їхню ефективність та доцільність. Вони спрямовані на передбачення майбутніх фінансових показників підприємства. Ці моделі дозволяють визначити можливі сценарії розвитку, оцінити їхні наслідки та підготувати підприємство до різних варіантів розвитку подій. Такий підхід сприяє адаптації до динамічних умов ринку.

Таблиця 1.1 - Опис методів, моделей та інструментів управління фінансуванням комерційних закладів

Категорія	Метод/ Модель/ Інструмент	Опис
Методи управління фінансуванням	Фінансовий контролінг	Моніторинг фінансових показників, аналіз відхилень, розробка коригувальних дій.
	Бюджетування	Планування доходів і витрат для раціонального розподілу ресурсів.
	Управління грошовими потоками	Прогнозування, управління дебіторською та кредиторською заборгованістю.
Моделі управління фінансуванням	Економіко-математичні моделі	Оптимізація фінансових рішень, моделі лінійного програмування.
	Моделі управління фінансовими ризиками	Ідентифікація, оцінка, нейтралізація ризиків (страхування, хеджування).
	Моделі фінансового планування та прогнозування	Створення планів та прогнозів з урахуванням сценаріїв розвитку.
Інструменти управління фінансуванням	SmartEAM	Програмне забезпечення для автоматизації бюджетування та моніторингу фінансових показників.
	EAM-системи	Системи для оптимізації технічного обслуговування та зниження витрат на простої обладнання.

Моделі фінансового планування та прогнозування є ключовими інструментами забезпечення фінансової стійкості та конкурентоспроможності підприємства. Їхні особливості, зокрема орієнтація на майбутнє, гнучкість, інтеграція з іншими управлінськими функціями та використання сучасних технологій, роблять ці моделі незамінними у процесах стратегічного та операційного управління. У сучасному мінливому середовищі такі моделі є

основою для прийняття обґрунтованих фінансових рішень. Моделі забезпечують системний підхід до фінансового планування, враховуючи взаємозв'язки між різними аспектами діяльності підприємства (операційною, інвестиційною та фінансовою). Інтеграція даних з різних підрозділів забезпечує узгодженість планів та підвищує їхню достовірність.

До основних інструментів управління фінансуванням відносяться програмні рішення для бюджетування та фінансового планування, системи управління технічним обслуговуванням (рис. 1.3).

Програмні рішення для бюджетування та фінансового планування, такі як SmartEAM, автоматизують процеси бюджетування, моніторингу фінансових показників та управління технічним обслуговуванням [12].

Рисунок 1.3 - Інструменти управління фінансуванням станцій технічного обслуговування

Програмні рішення для бюджетування та фінансового планування є сучасними інструментами, що забезпечують автоматизацію та підвищення ефективності управління фінансовими ресурсами підприємства. Їх використання характеризується низкою специфічних особливостей, які роблять ці інструменти важливими для досягнення стратегічних і операційних цілей. Вони дозволяють автоматизувати процеси створення, аналізу та моніторингу бюджетів, а також фінансових планів. Це зменшує обсяг ручної роботи, знижує ризик помилок та підвищує точність розрахунків. Такі системи, як SAP BPC, Oracle Hyperion та IBM

Planning Analytics, забезпечують інтеграцію даних з різних підрозділів, що сприяє швидкому прийняттю рішень.

Програмні рішення для бюджетування використовують єдину базу даних, яка дозволяє зберігати, обробляти та аналізувати фінансову інформацію. Це сприяє узгодженості даних між підрозділами підприємства, запобігає дублюванню інформації та полегшує доступ до актуальних даних у реальному часі.

Тому, однією з ключових особливостей є можливість адаптації програмних рішень до специфічних потреб підприємства. Наприклад, системи дозволяють створювати індивідуальні шаблони бюджетів, налаштовувати звітність за специфічними показниками або враховувати особливості галузі, в якій працює організація. Програмні рішення для бюджетування та фінансового планування є важливими інструментами, що дозволяють підприємствам підвищити ефективність управління фінансами, забезпечити узгодженість планів і прозорість процесів. Їх особливості, такі як автоматизація, інтеграція з іншими системами, використання аналітичних інструментів та підтримка масштабування, роблять ці рішення незамінними у сучасному динамічному бізнес-середовищі. Однак успішне впровадження цих рішень потребує ретельної підготовки, якісних даних та високої кваліфікації персоналу.

Системи управління технічним обслуговуванням (EAM-системи) дозволяють оптимізувати процеси технічного обслуговування та ремонту, що впливає на фінансові показники підприємства [13].

Системи управління технічним обслуговуванням (EAM-системи) (Enterprise Asset Management) є сучасними інформаційними інструментами, що забезпечують автоматизацію та оптимізацію процесів управління активами, технічним обслуговуванням і ремонтами. Їхнє впровадження та використання мають низку характерних особливостей, які роблять ці системи незамінними для підприємств у багатьох галузях. EAM-системи забезпечують комплексне управління активами підприємства на всіх етапах їхнього життєвого циклу: від придбання до виведення з експлуатації. Ця особливість дозволяє оптимізувати використання ресурсів, мінімізувати витрати та збільшити термін служби обладнання.

EAM-системи забезпечують автоматизацію процесів планування технічного обслуговування та ремонтів. Вони дозволяють створювати графіки обслуговування, враховуючи такі фактори, як інтенсивність використання обладнання, попередні ремонти та рекомендації виробника. Це знижує ризики аварій та незапланованих простоїв.

Тому, EAM-системи є ефективними інструментами для управління активами, які забезпечують автоматизацію процесів технічного обслуговування, підвищення ефективності використання ресурсів та мінімізацію витрат. Їх використання характеризується інтеграцією з іншими системами управління, підтримкою мобільних додатків, впровадженням предиктивних стратегій обслуговування та відповідністю міжнародним стандартам. Однак впровадження таких систем вимагає значних початкових інвестицій, адаптації до потреб підприємства та навчання персоналу, що необхідно враховувати при ухваленні рішення про їх застосування.

Таким чином, застосування зазначених методів, моделей та інструментів в управлінні фінансуванням СТО сприяє підвищенню їхньої фінансової стійкості, ефективності та конкурентоспроможності на ринку.

1.2 Порівняльний аналіз програмних інструментів для управління фінансуванням комерційних організацій

Управління фінансуванням станцій технічного обслуговування (СТО) та інших комерційних закладів є ключовим аспектом забезпечення їхньої ефективної діяльності та стійкості. Для цього використовуються різноманітні програмні рішення, що допомагають оптимізувати фінансові потоки, знижувати витрати та підвищувати прибутковість. Тому, розглянемо найбільш поширені та успішні програмні рішення в галузі фінансування комерційних організацій (табл. 1.2).

Таблиця 1.2 - Порівняльний аналіз програмних інструментів для управління фінансуванням комерційних організацій

Назва інструменту	Опис
Octane Systems	Система для управління запасами, моніторингу відповідності та автоматизованого виставлення рахунків.
HB Smart Fuel Station Solutions	Інтегровані модулі для управління фінансами, запасами та персоналом.
PDI Enterprise	Рішення для обліку, автоматизації замовлень і управління запасами у паливній галузі.
SAP ERP	Комплексна система управління підприємством із модулями для технічного обслуговування.
BAS ERP	Українське рішення для автоматизації управління фінансами, закупівлями та персоналом.
LEAFIO	Рішення з алгоритмами штучного інтелекту для автоматизації управління запасами.
SmartEAM	Система для управління технічним обслуговуванням, планування робіт та ведення звітності.
Modisoft Fuel Management System	Контроль продажів палива, управління запасами, інтеграція з POS-системами.
Hectronic	Рішення для авторизації, оплати, управління та вимірювання палива.
Petrolsoft Corporation	Програмні рішення для оптимізації дистрибуції та торгівлі паливом.
Verifone	Інтегровані POS-системи та термінали для обслуговування клієнтів на заправних станціях.
LS Retail	Рішення для управління паливними насосами, автоматизації запасів та обслуговування клієнтів.
POS Nation	POS-системи для магазинів на заправках, які продають тютюн, алкоголь, лотереї тощо.
Petrosoft SmartPOS	Управління обліком клієнтів, замовленнями палива та звітністю для комерційних продажів.
NCR Storefront	Обладнання та POS-термінали для модернізації паливних насосів на заправних станціях.

Octane Systems пропонує програмні рішення, орієнтовані на високопродуктивне управління фінансами у паливній галузі, що забезпечують комплексний підхід до автоматизації бізнес-процесів великих підприємств, які здійснюють імпорт, дистрибуцію та роздрібну торгівлю паливом. Особливістю цього програмного засобу є його здатність забезпечувати повний цикл фінансового управління, включаючи інтегровану бізнес-аналітику, автоматизоване виставлення рахунків, управління запасами, а також моніторинг відповідності регуляторним вимогам. Це сприяє підвищенню ефективності управління фінансовими потоками, зниженню ризиків та підтриманню стабільності діяльності підприємства [14].

Octane Systems забезпечує автоматизацію фінансових операцій, включаючи виставлення рахунків, облік платежів та управління кредиторською і дебіторською заборгованістю. Це дозволяє підприємствам знижувати помилки в обліку, прискорювати фінансові процеси та підвищувати прозорість фінансової звітності. Вона надає можливість моніторингу та управління запасами палива на всіх етапах - від складування до дистрибуції. Це дозволяє підприємствам ефективно планувати закупівлі, мінімізувати ризики дефіциту або перевищення запасів, а також оптимізувати логістичні процеси.

Програмний засіб включає модулі для збору, аналізу та візуалізації бізнес-даних, що дає змогу оцінювати ефективність операцій, прогнозувати попит та оптимізувати використання ресурсів. Завдяки цим можливостям керівництво підприємства отримує інструменти для прийняття обґрунтованих управлінських рішень.

Octane Systems забезпечує автоматизований контроль відповідності діяльності підприємства регуляторним вимогам, що є критично важливим у паливній галузі. Це включає моніторинг екологічних норм, стандартів безпеки та податкових вимог. Система спрощує процеси формування та розсилання рахунків клієнтам, знижуючи адміністративне навантаження на персонал. Ця функція сприяє скороченню часу обробки платежів та зменшенню ймовірності помилок у фінансових документах.

Octane Systems розроблений з урахуванням потреб великих компаній, що займаються імпортом, дистрибуцією та роздрібною торгівлею паливом. Система

підтримує високу продуктивність і масштабованість, що робить її ефективною для організацій із великими обсягами операцій.

Тому, Octane Systems є комплексним рішенням, яке об’єднує фінансовий облік, управління запасами, аналітику та контроль відповідності. Його функціональні особливості спрямовані на підвищення ефективності, точності та прозорості бізнес-процесів у паливній галузі, що робить цю систему важливим інструментом для великих підприємств.

Smart Fuel Station Solutions, розроблений компанією Hidden Brains, є програмним продуктом, що інтегрує сучасні технології для підвищення ефективності роботи заправних станцій. Особливістю цього рішення є наявність модулів для управління фінансами, які забезпечують точний облік продажів, витрат та доходів, а також модулів для управління запасами і персоналом. Такий функціонал сприяє оптимізації бізнес-процесів, підвищенню продуктивності та забезпеченню прозорості фінансової діяльності підприємства [15].

Smart Fuel Station Solutions, розроблений компанією Hidden Brains, є інтегрованим програмним засобом, що спрямований на підвищення ефективності та оптимізацію операційних процесів на заправних станціях. Цей інструмент вирізняється кількома функціональними особливостями, які забезпечують високий рівень автоматизації та управління.

Smart Fuel Station Solutions забезпечує інтеграцію сучасних цифрових технологій для автоматизації бізнес-процесів. Це дозволяє зменшити залежність від ручного втручання, підвищуючи точність і швидкість виконання операцій, таких як облік продажів і моніторинг витрат. Система пропонує інструменти для збору та аналізу даних у реальному часі. Вона дозволяє створювати аналітичні звіти про обсяги продажів, витрати, запаси та інші показники. Це допомагає керівникам приймати обґрунтовані управлінські рішення, базуючись на актуальній інформації.

Smart Fuel Station Solutions є масштабованим рішенням, яке може бути адаптоване до потреб як невеликих заправних станцій, так і великих мереж. Система підтримує інтеграцію з іншими програмними продуктами та модифікацію функціоналу для вирішення специфічних завдань. Система забезпечує відповідність нормативним вимогам у галузі, таким як стандарти безпеки даних,

податкове законодавство та екологічні норми. Це знижує ризик порушення регуляторних норм та покращує репутацію підприємства.

Тому, Smart Fuel Station Solutions від Hidden Brains є потужним інструментом для автоматизації та оптимізації операційних і фінансових процесів на заправних станціях. Його функціональні особливості, включаючи управління фінансами, запасами, персоналом, підтримку клієнтоорієнтованих функцій і аналітичних можливостей, сприяють підвищенню ефективності роботи та забезпечують конкурентні переваги для підприємств у паливній галузі.

PDI Enterprise пропонує рішення для управління роздрібними та оптовими операціями в паливній галузі. Система охоплює фінансовий облік, управління запасами, автоматизацію замовлень та інші функції, що сприяють підвищенню продуктивності та ефективності бізнесу [16].

PDI Enterprise є комплексним програмним засобом, розробленим для автоматизації та оптимізації операційних і фінансових процесів у підприємствах, що працюють у паливній галузі. Цей інструмент орієнтований на управління дистрибуцією, роздрібною торгівлею та іншими аспектами діяльності, пов’язаними із продажем палива та супутніх товарів. Його функціональні особливості забезпечують високий рівень ефективності й прозорості бізнес-процесів.

PDI Enterprise підтримує повну інтеграцію між оптовими та роздрібними операціями. Система дозволяє синхронізувати дані між дистриб’юторськими складами та заправними станціями, забезпечуючи узгодженість і актуальність інформації. Програмний засіб включає інструменти для автоматизованого створення замовлень на основі аналізу залишків запасів та історичних даних продажів. Це знижує адміністративне навантаження на персонал і мінімізує ризик людських помилок. Система забезпечує відповідність локальним та міжнародним стандартам фінансової звітності та податкового обліку. Це сприяє зниженню ризиків порушень регуляторних норм і полегшує аудит.

PDI Enterprise розроблена з урахуванням потреб підприємств різного масштабу - від невеликих мереж до великих корпорацій. Її гнучка архітектура дозволяє адаптувати функціонал до специфічних бізнес-процесів і легко масштабувати систему в разі розширення діяльності. Програмний засіб підтримує

інтеграцію з ERP-системами, POS-терміналами та іншими інструментами управління, що забезпечує створення єдиного інформаційного середовища для підприємства.

PDI Enterprise сумісна з мобільними платформами та IoT-пристроями, що дозволяє віддалено контролювати операції, моніторити запаси палива в резервуарах та аналізувати показники ефективності. Тому, PDI Enterprise є потужним програмним інструментом для комплексного управління операціями в паливній галузі. Його функціональні особливості, зокрема управління запасами, інтеграція дистрибуції та роздрібної торгівлі, аналітичні можливості та масштабованість, забезпечують ефективність бізнес-процесів, прозорість фінансових операцій і стратегічну конкурентоспроможність підприємства.

SAP ERP є комплексною системою управління підприємством, яка включає модулі для фінансового планування та бюджетування. Вона оптимізує процеси технічного обслуговування, включаючи планування, контроль виконання робіт, управління запасами та інструментами, що є важливим для СТО [17].

SAP ERP (Enterprise Resource Planning) є інтегрованим програмним рішенням, яке забезпечує автоматизацію та управління бізнес-процесами на підприємствах різних галузей. Цей програмний засіб вирізняється широкими функціональними можливостями, що сприяють оптимізації фінансових, операційних та управлінських процесів. У контексті станцій технічного обслуговування (СТО) і комерційних закладів його функціональні особливості забезпечують підвищення ефективності та прозорості управління.

SAP ERP забезпечує інтеграцію всіх основних бізнес-процесів у межах єдиної системи, включаючи управління фінансами, логістикою, технічним обслуговуванням та людськими ресурсами. Це сприяє узгодженості між підрозділами, зменшує дублювання даних і підвищує швидкість прийняття рішень.

SAP ERP розроблений для підприємств різного масштабу - від малих компаній до великих корпорацій. Його архітектура дозволяє адаптувати систему до специфічних потреб бізнесу, а також масштабувати її у разі розширення діяльності. Система підтримує інтеграцію з IoT-пристроями та мобільними додатками, що

забезпечує віддалений контроль операцій, моніторинг обладнання та доступ до даних у реальному часі.

SAP ERP враховує міжнародні стандарти та локальне законодавство, що забезпечує відповідність звітності й операційних процесів нормативним вимогам. Це особливо важливо для підприємств, які працюють у регульованих галузях, таких як паливна або енергетична. У контексті комерційних закладів система підтримує створення та управління програмами лояльності, що дозволяє залучати нових клієнтів і підвищувати задоволеність існуючих. Завдяки автоматизації та оптимізації процесів SAP ERP сприяє зниженню операційних витрат, підвищенню продуктивності працівників і забезпеченню довгострокової стійкості бізнесу. Тому, SAP ERP є потужним інструментом для автоматизації та управління бізнес-процесами, що забезпечує інтеграцію, аналітику та масштабованість. Його функціональні особливості роблять цю систему незамінною для підприємств, які прагнуть підвищити ефективність, забезпечити відповідність нормативним вимогам і створити конкурентні переваги в умовах динамічного ринкового середовища.

BAS ERP є українським рішенням для автоматизації бізнес-процесів, включаючи управління фінансами, торговими операціями, закупівлями та персоналом. Продукти лінійки BAS оптимальні для будь-якого бізнесу, незалежно від його масштабів і сфери діяльності [18]. BAS ERP є програмним засобом українського виробництва, розробленим для автоматизації бізнес-процесів підприємств у різних галузях. Ця система підтримує управління фінансовими, виробничими, торговельними та іншими операціями, сприяючи підвищенню ефективності та прозорості діяльності підприємства. BAS ERP забезпечує комплексний підхід до управління ресурсами та процесами завдяки таким функціональним особливостям.

BAS ERP забезпечує можливість роботи багатьох користувачів із різними рівнями доступу до даних і функцій. Це сприяє безпеці інформації та узгодженості між відділами. Система легко інтегрується з іншими програмними продуктами, такими як CRM-системи, бухгалтерські програми та спеціалізовані інструменти для галузевих потреб. Це створює єдине інформаційне середовище для підприємства.

Система розроблена з урахуванням українського законодавства та стандартів звітності, що забезпечує відповідність регуляторним вимогам. Це особливо важливо для підприємств, які працюють у регульованих галузях.

BAS ERP розрахована на підприємства різного масштабу - від малих компаній до великих корпорацій. Її функціонал може бути адаптований до специфіки галузі або бізнес-процесів конкретного підприємства. Завдяки автоматизації рутинних завдань BAS ERP допомагає підприємствам скорочувати витрати на адміністративне управління, підвищуючи ефективність роботи персоналу та використання ресурсів. Тому, BAS ERP є універсальним програмним засобом для комплексного управління бізнесом. Її функціональні особливості, зокрема управління фінансами, виробництвом, торгівлею та персоналом, а також інтеграція, аналітика і відповідність нормативним вимогам, забезпечують оптимізацію бізнес-процесів, підвищення продуктивності та довгострокову стабільність підприємств у конкурентному середовищі.

LEAFIO пропонує програмні рішення для "розумних" заправних станцій, що використовують алгоритми штучного інтелекту для автоматизації управління запасами, асортиментом продукції та простором на полицях. Це сприяє підвищенню ефективності та зниженню витрат [19]. LEAFIO є програмним засобом для автоматизації управління запасами, асортиментом продукції та простором у роздрібній торгівлі. Це рішення орієнтоване на використання сучасних технологій, включаючи алгоритми штучного інтелекту (AI), що дозволяють підвищити ефективність управління ресурсами. Його функціональні особливості забезпечують оптимізацію бізнес-процесів і сприяють зниженню операційних витрат.

LEAFIO легко інтегрується з іншими програмними продуктами, зокрема ERP-системами та POS-терміналами, що дозволяє створити єдине інформаційне середовище для управління бізнесом. Це сприяє синхронізації даних і підвищенню ефективності роботи. LEAFIO адаптується до потреб підприємств різного масштабу - від невеликих магазинів до великих торговельних мереж. Система підтримує масштабування функціоналу у разі розширення бізнесу. Система сумісна з мобільними пристроями, що дозволяє працівникам отримувати доступ до даних у

реальному часі та оперативно виконувати завдання, пов’язані з управлінням запасами або асортиментом. Завдяки автоматизації процесів управління запасами та оптимізації асортименту LEAFIO дозволяє скоротити операційні витрати, знизити втрати через надлишки або дефіцит продукції та підвищити прибутковість підприємства. Тому, LEAFIO є сучасним програмним засобом, який поєднує автоматизацію, використання алгоритмів штучного інтелекту та аналітичні можливості для ефективного управління запасами, асортиментом і простором. Його функціональні особливості сприяють оптимізації бізнес-процесів, підвищенню продуктивності та забезпеченню конкурентоспроможності підприємств у роздрібній торгівлі.

SmartEAM пропонує рішення для управління технічним обслуговуванням, що використовують електронні бази даних про операції технічного обслуговування, планування робіт та ведення звітності. Це допомагає оптимізувати процеси обслуговування та знижувати витрати [20]. SmartEAM є програмним засобом, призначеним для управління активами підприємства, з особливим акцентом на автоматизацію технічного обслуговування та ремонту обладнання. Цей інструмент забезпечує ефективну організацію процесів управління активами протягом їхнього життєвого циклу, підвищуючи операційну ефективність і мінімізуючи ризики простоїв. Функціональні особливості SmartEAM охоплюють широкий спектр можливостей, що відповідають сучасним вимогам управління.

SmartEAM забезпечує автоматизацію процесів, пов’язаних із придбанням, експлуатацією, обслуговуванням та виведенням з експлуатації активів. Ця функціональність сприяє максимальному використанню ресурсів та подовженню терміну служби обладнання. Система інтегрується з IoT-пристроями та сенсорами, що дозволяє виявляти потенційні несправності до їхнього виникнення. Використання алгоритмів штучного інтелекту для аналізу даних забезпечує своєчасну оцінку стану обладнання та запобігання аварійним ситуаціям.

SmartEAM легко інтегрується з іншими корпоративними системами, такими як ERP, CRM або бухгалтерські програми. Це створює єдиний інформаційний простір, який забезпечує узгодженість процесів між підрозділами підприємства. Система пропонує мобільні додатки для доступу до даних у реальному часі, що

дозволяє працівникам отримувати інформацію про стан обладнання та виконувати завдання безпосередньо на місці проведення робіт. Це сприяє оперативності та гнучкості у виконанні технічного обслуговування. Програмний засіб розроблений із урахуванням міжнародних стандартів у сфері управління активами, таких як ISO 55001. Це забезпечує дотримання вимог щодо якості, екологічної безпеки та ефективності управління.

SmartEAM підходить для підприємств різного масштабу, від невеликих організацій до великих корпорацій. Система може бути адаптована до специфічних потреб бізнесу та розширена у разі зростання обсягів діяльності. Завдяки автоматизації процесів технічного обслуговування та оптимізації використання ресурсів, SmartEAM дозволяє знизити витрати на утримання активів, збільшити їхню продуктивність та забезпечити довгострокову стабільність підприємства. Тому, SmartEAM є високоефективним рішенням для управління активами, що поєднує сучасні технології, аналітичні можливості та інструменти автоматизації. Його функціональні особливості, такі як підтримка предиктивного обслуговування, інтеграція з ERP-системами, аналітика та масштабованість, роблять цей програмний засіб важливим елементом інфраструктури сучасних підприємств, спрямованих на ефективне управління ресурсами.

Modisoft Fuel Management System дозволяє повністю контролювати заправну станцію, спрощуючи продажі палива та управління запасами. Система забезпечує інтеграцію з POS-системами та надає аналітичні інструменти для прийняття обґрунтованих рішень [21]. Modisoft Fuel Management System є інтегрованим програмним засобом, розробленим для автоматизації процесів управління заправними станціями, що охоплює продажі палива, управління запасами та аналітику. Цей інструмент забезпечує ефективне управління операціями, знижує витрати та сприяє підвищенню продуктивності підприємств, які працюють у паливному секторі. Функціональні особливості системи роблять її незамінним інструментом для сучасного бізнесу. Система легко інтегрується з існуючими інструментами управління, такими як ERP, CRM та бухгалтерські програми. Це забезпечує узгодженість даних між підрозділами підприємства, полегшує облік операцій і підвищує ефективність бізнес-процесів.

Modisoft Fuel Management System дозволяє впроваджувати програми лояльності, включаючи системи накопичення бонусів, надання знижок та індивідуальні пропозиції для постійних клієнтів. Це сприяє підвищенню конкурентоспроможності підприємства та утриманню клієнтів. Система підтримує доступ до даних через мобільні додатки, що дозволяє власникам і менеджерам отримувати інформацію у реальному часі, контролювати операції та приймати рішення незалежно від місця перебування.

Modisoft Fuel Management System підходить як для невеликих заправних станцій, так і для великих мереж. Її архітектура дозволяє легко адаптувати функціонал до потреб бізнесу та масштабувати систему у разі розширення підприємства.

Система враховує нормативні вимоги у галузі паливного бізнесу, забезпечуючи автоматизацію процесів звітності відповідно до місцевого та міжнародного законодавства. Це знижує ризики помилок та підвищує прозорість діяльності. Тому, Modisoft Fuel Management System є потужним інструментом для автоматизації та оптимізації операційних процесів у паливному бізнесі. Її функціональні особливості, такі як управління продажами, запасами, аналітика, мобільність та підтримка програм лояльності, забезпечують підвищення ефективності, прозорості та конкурентоспроможності підприємства. Система є важливим елементом інфраструктури для сучасних заправних станцій, спрямованих на інновації та довгостроковий розвиток.

Hectronic пропонує розумні рішення для управління заправними станціями, включаючи паливні термінали та програмні рішення для авторизації, оплати, управління та вимірювання. Це забезпечує безперебійну роботу станцій та швидке обслуговування клієнтів [22]. Hectronic є провідним розробником програмно-апаратних рішень для управління заправними станціями, які поєднують інноваційні технології та високу функціональність. Основні функціональні особливості програмного засобу від Hectronic спрямовані на забезпечення автоматизації операцій, підвищення ефективності роботи та оптимізацію бізнес-процесів у паливному секторі.

Hectronic забезпечує відповідність локальним і міжнародним стандартам у сфері паливного бізнесу, включаючи вимоги щодо екологічної безпеки, податкової звітності та захисту даних. Це знижує ризики порушень регуляторних норм і покращує репутацію підприємства. Система підтримує функції віддаленого доступу, що дозволяє керівникам контролювати операції, моніторити запаси та аналізувати показники діяльності у реальному часі, незалежно від місця перебування.

Hectronic пропонує мобільні рішення для клієнтів і операторів, які дозволяють здійснювати транзакції, отримувати доступ до інформації про запаси палива та управляти операціями через мобільні пристрої. Програмний засіб підходить як для окремих заправних станцій, так і для великих мереж. Його архітектура дозволяє легко адаптувати функціонал до потреб бізнесу та масштабувати систему у разі розширення підприємства. Тому, Hectronic є багатофункціональним рішенням для автоматизації та оптимізації роботи заправних станцій. Його функціональні особливості, включаючи управління паливними терміналами, автоматизацію операцій, інтеграцію з POS-системами, аналітику та віддалений моніторинг, забезпечують високу ефективність, прозорість і конкурентоспроможність підприємств у паливному секторі.

Petrolsoft Corporation розробляє програмні рішення для управління ланцюгом постачання в паливній галузі, що дозволяють оптимізувати процеси дистрибуції та роздрібної торгівлі паливом. Їхні продукти використовуються багатьма великими нафтовими компаніями по всьому світу [23]. Petrolsoft Corporation пропонує програмні рішення, орієнтовані на управління дистрибуцією, роздрібною торгівлею та логістикою в паливному бізнесі. Цей програмний засіб забезпечує комплексний підхід до автоматизації та оптимізації бізнес-процесів, що пов’язані з постачанням палива, управлінням запасами та контролем за фінансовими потоками. Функціональні особливості рішень Petrolsoft Corporation роблять їх важливим інструментом для підприємств паливного сектору.

Petrolsoft Corporation забезпечує інтеграцію з ERP-системами, що створює єдину інформаційну інфраструктуру для підприємства. Це дозволяє синхронізувати дані між різними підрозділами та забезпечує узгодженість бізнес-процесів.

Програмний продукт розроблений для роботи на підприємствах різного масштабу, включаючи як невеликі регіональні компанії, так і великі мережі заправних станцій. Його архітектура дозволяє масштабувати систему відповідно до потреб бізнесу.

Система враховує актуальні нормативні вимоги у галузі паливного бізнесу, включаючи стандарти безпеки, екологічні норми та податкове законодавство. Це допомагає знизити ризики та забезпечити відповідність діяльності підприємства законодавчим нормам. Тому, Petrolsoft Corporation пропонує комплексне програмне рішення для автоматизації та оптимізації бізнес-процесів у паливному секторі. Його функціональні особливості, такі як управління ланцюгом постачання, автоматизація фінансових операцій, аналітика та прогнозування, інтеграція з ERP-системами та масштабованість, роблять цей інструмент важливим елементом для підвищення ефективності, прозорості та конкурентоспроможності підприємств у галузі.

Verifone пропонує інтегровані рішення для заправних станцій, включаючи POS-системи, термінали для оплати та цифрові вивіски. Це допомагає оптимізувати обслуговування клієнтів та підвищити ефективність операцій [24]. Verifone є провідним розробником інтегрованих рішень для автоматизації та управління операціями на заправних станціях, орієнтованим на забезпечення зручності для клієнтів і ефективності бізнесу. Цей програмний засіб пропонує широкий спектр функціональних можливостей, які сприяють оптимізації обслуговування клієнтів, інтеграції процесів і підвищенню продуктивності підприємств паливного сектора.

Verifone відповідає сучасним стандартам безпеки, зокрема PCI DSS (Payment Card Industry Data Security Standard), що забезпечує захист фінансових даних клієнтів і знижує ризики шахрайства. Система підходить як для окремих заправних станцій, так і для великих мереж. Її модульна архітектура дозволяє масштабувати функціонал відповідно до зростання бізнесу. Тому, Verifone є комплексним програмним рішенням, яке спрямоване на підвищення ефективності управління заправними станціями та покращення обслуговування клієнтів. Його функціональні особливості, такі як інтегровані POS-рішення, підтримка цифрових сервісів, програми лояльності, аналітика та забезпечення безпеки даних, роблять

його важливим інструментом для підвищення конкурентоспроможності підприємств у паливному секторі.

LS Retail надає рішення для великих заправних станцій, що включають управління паливними насосами, автоматизоване поповнення запасів та кіоски для обслуговування клієнтів. Система легко масштабується та адаптується до потреб бізнесу [25]. LS Retail є програмним засобом, що забезпечує комплексне управління бізнес-процесами на заправних станціях і в супутній роздрібній торгівлі. Це рішення інтегрує всі аспекти операційної діяльності в єдину платформу, сприяючи оптимізації процесів, підвищенню продуктивності та поліпшенню обслуговування клієнтів. Основні функціональні особливості LS Retail забезпечують ефективне управління як невеликими станціями, так і великими мережами.

Архітектура LS Retail дозволяє масштабувати систему відповідно до потреб бізнесу, від обслуговування однієї заправної станції до управління великою мережею. Це забезпечує гнучкість у процесі розвитку підприємства. Система відповідає міжнародним стандартам у сфері безпеки даних і захисту транзакцій, що мінімізує ризики шахрайства та забезпечує відповідність нормативно-правовим нормам. Тому, LS Retail є багатофункціональним рішенням для автоматизації бізнес-процесів заправних станцій і роздрібної торгівлі. Його функціональні особливості, такі як інтеграція POS-рішень, управління запасами, підтримка програм лояльності, аналітика та масштабованість, роблять цей програмний засіб важливим інструментом для підвищення ефективності, прозорості та конкурентоспроможності підприємств у паливному секторі.

POS Nation спеціалізується на POS-системах для магазинів, що продають алкоголь, тютюн, лотереї та інші товари [26]. Хоча система не має вбудованого управління паливом, вона є ефективною для магазинів при заправних станціях.

Petrosoft SmartPOS пропонує рішення для комерційних продажів палива, включаючи управління обліковими записами клієнтів, замовлення палива та розширену звітність. Це ідеально підходить для заправних станцій, що обслуговують комерційних клієнтів [27].

NCR Storefront пропонує широкий спектр обладнання для заправних станцій, включаючи NCR Optic - універсальний POS-термінал, який можна встановити на

існуючі паливні насоси. Це дозволяє модернізувати станції без значного оновлення обладнання [28]. NCR Storefront є інтегрованим програмно-апаратним рішенням, розробленим для модернізації та оптимізації процесів обслуговування клієнтів на заправних станціях. Цей програмний засіб вирізняється низкою функціональних особливостей, які забезпечують ефективність та інноваційність у сфері управління операціями на підприємствах паливного сектору. Однією з ключових функціональних особливостей NCR Storefront є здатність інтегруватися з існуючими паливними насосами та іншими компонентами заправних станцій. Завдяки цьому підприємства можуть модернізувати свою інфраструктуру без необхідності повної заміни обладнання, що сприяє зниженню капітальних витрат.

Рішення включає NCR Optic, універсальний POS-термінал, який підтримує різні способи оплати, зокрема безконтактні платежі, мобільні додатки та карткові транзакції. Ця функція розширює можливості обслуговування клієнтів, забезпечуючи швидкість і зручність розрахунків.

NCR Storefront дозволяє автоматизувати процеси обліку продажів палива та супутніх товарів. Система забезпечує точний контроль операцій, що знижує ймовірність помилок, пов’язаних із ручним обліком, і сприяє підвищенню прозорості бізнес-процесів. Система підтримує функції персоналізованого обслуговування, такі як програми лояльності, надання індивідуальних пропозицій та аналіз поведінки клієнтів. Це дозволяє заправним станціям покращувати клієнтський досвід і підвищувати рівень задоволеності споживачів. Програмний засіб надає можливість аналізу даних у реальному часі, що сприяє прийняттю обґрунтованих управлінських рішень. Зокрема, аналітичні модулі дозволяють оцінювати обсяги продажів, контролювати запаси та прогнозувати майбутні потреби.

NCR Storefront сумісний із мобільними додатками для обслуговування клієнтів, що дозволяє створювати додаткові точки взаємодії. Це забезпечує гнучкість і зручність для користувачів, які віддають перевагу цифровим способам обслуговування. Система розроблена з урахуванням потреб як невеликих заправних станцій, так і великих мереж. Її архітектура дозволяє легко

масштабувати функціональність відповідно до обсягів бізнесу, додаючи нові модулі або пристрої.

NCR Storefront забезпечує високий рівень безпеки даних під час транзакцій, дотримуючись міжнародних стандартів, таких як PCI DSS (Payment Card Industry Data Security Standard). Це сприяє захисту клієнтських даних і знижує ризик шахрайства. Тому, NCR Storefront є сучасним і функціонально багатим рішенням, яке забезпечує ефективне управління продажами, інтеграцію з існуючими технологіями та підвищення якості обслуговування клієнтів. Його можливості автоматизації, персоналізації та масштабування роблять цей програмний засіб важливим елементом інфраструктури для заправних станцій, орієнтованих на інновації та довгостроковий розвиток.

1.3 Постановка задач дослідження

Виконавши існуючий аналіз предметної галузі та особливості управління фінансуванням станцій технічного обслуговування було визначено комплекс задач, реалізація яких дозволить розробити продуктивне програмне забезпечення. Основними задачами роботи є:

- виконати порівняльний аналіз програмних інструментів для управління фінансуванням комерційних організацій
- запропонувати метод комплексного фінансового контролінгу;
- розробити функціональні вимоги до програмного засобу;
- виконати проектування основних та додаткових модулів програмного засобу;
- визначити та розробити основні принципи програмного засобу на основі блокчейну;
- визначити та розробити основні принципи роботи модуля управління замовленнями на основі блокчейн;
- визначити та розробити основні принципи роботи модуля інтеграції з постачальниками програмного засобу;

- визначити та розробити особливості тестування мікросервісної архітектури програмного засобу;

- виконати тестування програмного модуля управління замовленнями;

- виконати тестування програмного модуля інтеграції з постачальниками програмного засобу.

2 ПРОЕКТУВАННЯ ПРОГРАМНОГО ЗАСОБУ

2.1 Запропонований метод комплексного фінансового контролінгу

Комплексний фінансовий контролінг для станцій технічного обслуговування (СТО) із застосуванням блокчейн-технологій являє собою інноваційну систему управління, що інтегрує фінансові, операційні та технологічні аспекти діяльності. Його сутність полягає в синхронному аналізі доходів, витрат і ключових операційних показників через децентралізовані механізми обліку, що забезпечують прозорість, точність і оперативність прийняття рішень.

Архітектура системи базується на розподілених реєстрах блокчейну, де кожна транзакція фіксується у вигляді криптографічно захищеного блоку. Для СТО це означає автоматичне відображення таких операцій як:

- надходження коштів за послуги (діагностика, ремонт, продаж запчастин);

- витрати на закупівлю матеріалів, оплату праці співробітників, комунальні послуги, тощо;

- показники завантаження станції (кількість обслугованих авто, середній час ремонту, використання обладнання).

До особливості архітектури системи на основі розподілених реєстрів блокчейну полягають у комбінації технологічних та організаційних принципів, що забезпечують безпеку, прозорість і стабільність системи (табл. 2.1).

Децентралізація забезпечує відсутність єдиного центру управління (табл. 2.1), що підвищує стійкість системи до збоїв. Розподіленість реалізується через синхронізацію копій реєстру між усіма вузлами, що виключає можливість маніпуляцій. Незмінність досягається завдяки ланцюжку криптографічних хешів, де зміна будь-якого блоку вимагає перерахунку всіх наступних.

Консенсусні алгоритми (такі як Proof of Work або Byzantine Fault Tolerance) визначають правила валідації транзакцій, гарантуючи їхню легітимність. Криптографічні механізми захищають дані через пари ключів (публічний та приватний), що обмежують доступ до операцій. Прозорість архітектури дозволяє учасникам моніторити стан мережі в реальному часі, що критично для фінансового контролінгу.

Таблиця 2.1 - Особливості архітектури системи, що базується на розподілених реєстрах блокчейну

Характеристика	Опис
Децентралізація	Відсутність центрального контролю; мережа підтримується групою незалежних вузлів
Розподіленість	Кожен учасник мережі зберігає повну копію реєстру, що усуває єдину точку відмови
Незмінність	Записані дані не підлягають редагуванню або видаленню через криптографічне хешування
Консенсусні алгоритми	Механізми (PoW, PoS, BFT) для узгодження транзакцій між вузлами без довіри між ними
Криптографічна безпека	Використання асиметричного шифрування для автентифікації та захисту транзакцій
Прозорість	Публічний доступ до історії транзакцій для всіх учасників мережі
Автоматизований аудит	Можливість перевірки цілісності даних через алгоритмічні правила без участі третіх осіб

Ключові функціональні компоненти метод комплексного фінансового контролінгу включають наступне:

- смарт-контракти для автоматизації платежів із клієнтами та постачальниками, де умови оплати ремонту активуються лише після фіксації успішного завершення робіт у блокчейн-реєстрі;

- систему моніторингу витрат із реальночасовим аналізом відхилень, де датчики на обладнанні СТО передають дані про використання матеріалів, що дозволяє порівнювати фактичні витрати з плановими;

- предиктивні моделі на основі машинного навчання, які прогнозують фінансові ризики, де зниження попиту в зимовий сезон відбуваються з урахуванням історичних даних із блокчейну.

До переваги інтеграції блокчейну відносяться:

- незмінність записів усуває ризики маніпуляцій з фінансовою звітністю;

- децентралізована валідація транзакцій мінімізує помилки ручного введення даних;

- автоматизований аудит через публічний доступ до транзакційного ланцюжка зменшує витрати на зовнішні перевірки.

Для СТО оперативне виявлення відхилень реалізується через такі чинники:

- порогові сповіщення у разі перевищення витрат на 10% від планових показників

- динамічні дашборди із візуалізацією маржинальності послуг у реальному часі

- сигнатризацію аномалій у закупівельних цінах запчастин через порівняння з ринковими даними з зовнішніх блокчейн-джерел.

Тому, управлінські рішення базуються на комбінації автоматизованих сценаріїв і аналітики:

- автоматична корекція цін при зміні курсів валют або вартості комплектуючих через інтеграцію з біржами;

- оптимізація графіків роботи механіків на основі аналізу завантаженості станції за попередні тижні;

- стратегічне планування інвестицій у нове обладнання з урахуванням прогнозованої рентабельності на основі даних про тренди попиту.

Технічні виклики методу комплексного фінансового контролінгу включають:

- необхідність адаптації існуючих ERP-систем до блокчейн-архітектури;

- високі початкові витрати на впровадження IoT-пристроїв для збору операційних даних;

- потреба у підготовці персоналу для роботи з новими інструментами аналітики.

Розглянемо приклад застосування методу комплексного фінансового контролінгу на станції технічного обслуговування "AutoPro", яка впровадила блокчейн-орієнтований фінансовий контролінг. Так, в результатів застосування методу комплексного фінансового контролінгу (абсолютні значення були змінені, однак відносні значення - залишені) були отримані такі показники (табл. 2.2).

Таблиця 2.2 - Показники впровадження методу комплексного фінансового контролінгу на станції технічного обслуговування "AutoPro"

Показники	Значення	Ефект
Доходи	1,2 млн ₴/міс (60% - ремонти, 30% - продаж запчастин, 10% - додаткові послуги)	Зростання на 18% за рік
Витрати	900 тис. ₴/міс (матеріали - 45%, зарплата - 30%, комунальні - 10%, інше - 15%)	Скорочення на 12% через оптимізацію
Операційні показники	90% завантаження обладнання, середній час ремонту - 2,5 год	Підвищення продуктивності на 25%

Впровадження методу комплексного фінансового контролінгу на станції технічного обслуговування "AutoPro" (табл. 2.2) передбачало використання блокчейн-трекінг транзакцій, де кожен платіж клієнта фіксувався у смарт-контракті з автоматичним розподілом коштів (на рахунок СТО, постачальників, податки). Тоді, витрати на матеріали відстежувались через RFID-мітки на запчастинах, де дані синхронізувались з блокчейном. Тоді, виконувався аналіз у реальному часі під час якого дашборд програмного засобу відображав:

- маржинальність послуг (наприклад, діагностика - 45%, заміна гальм - 32%);

- порівняння планових/фактичних витрат із сигналізацією при відхиленні >5%.

Також, використовувались предиктивні моделі та алгоритми на основі історичних даних, що прогнозували сезонність попиту (наприклад, +20% до доходів у зимовий період) та оптимізує закупівлю матеріалів. Так, за результати за 12 місяців відмічались:

чистий прибуток зріс з 5,2% до 8,7% за рахунок:

- скорочення витрат на матеріали на 15% через перехід на аукціонну модель закупівлі в блокчейн-мережі;

- збільшення доходів від додаткових послуг (мийка, шиномонтаж) на 40% через таргетовані SMS-пропозиції клієнтам;

- рентабельність активів покращилася з 9% до 14% через оптимізацію використання обладнання.

Таким чином, інтеграція блокчейну дозволила "AutoPro" досягти 8,7% чистого прибутку при середньогалузевому показнику 5-6%, демонструючи ефективність методу.

Тому, інтеграція блокчейн-мереж у фінансовий контролінг СТО створює економічний ефект між технологічною інфраструктурою та управлінськими практиками. Це забезпечує перехід від ефективне управління ресурсами, де кожне рішення ґрунтується на верифікованих даних із гарантованою достовірністю.

Запропонований метод комплексного фінансового контролінгу оснований на процесі управління фінансуванням станцій технічного обслуговування (СТО). Управління фінансуванням СТО є важливим аспектом забезпечення ефективності їхньої роботи та фінансової стійкості. Основні завдання управління фінансуванням включають оптимізацію витрат на закупівлю матеріалів, підтримку своєчасних поставок та підвищення прозорості фінансових операцій. Сучасні методи управління передбачають використання розподілених децентралізованих мереж, таких як блокчейн, для підвищення надійності та ефективності фінансових операцій [29].

Процес взаємодії постачальників із замовниками матеріалів для СТО включає декілька ключових етапів (рис. 2.1):

- формування замовлення, де на основі поточних потреб і прогнозів СТО формує запит на постачання необхідних деталей та витратних матеріалів. У цьому етапі враховуються як кількісні, так і якісні параметри матеріалів [30];

- вибір постачальника, де замовник обирає постачальника на основі аналізу ціни, термінів постачання, якості продукції та репутації. Використання розподілених мереж на основі блокчейна дозволяє зберігати прозорість процесу вибору та уникати маніпуляцій [31];

- узгодження умов, де учасники узгоджують умови постачання, включаючи строки, вартість, логістичні аспекти та способи оплати. Смарт-контракти в блокчейн-мережі автоматизують цей процес, забезпечуючи виконання зобов’язань кожної сторони [32];

- доставка та оплата, де постачальник виконує доставку матеріалів до СТО. Після підтвердження отримання товару смарт-контракт ініціює автоматичне

проведення оплати. Це дозволяє уникнути затримок у фінансових операціях і знижує ризики шахрайства [33].

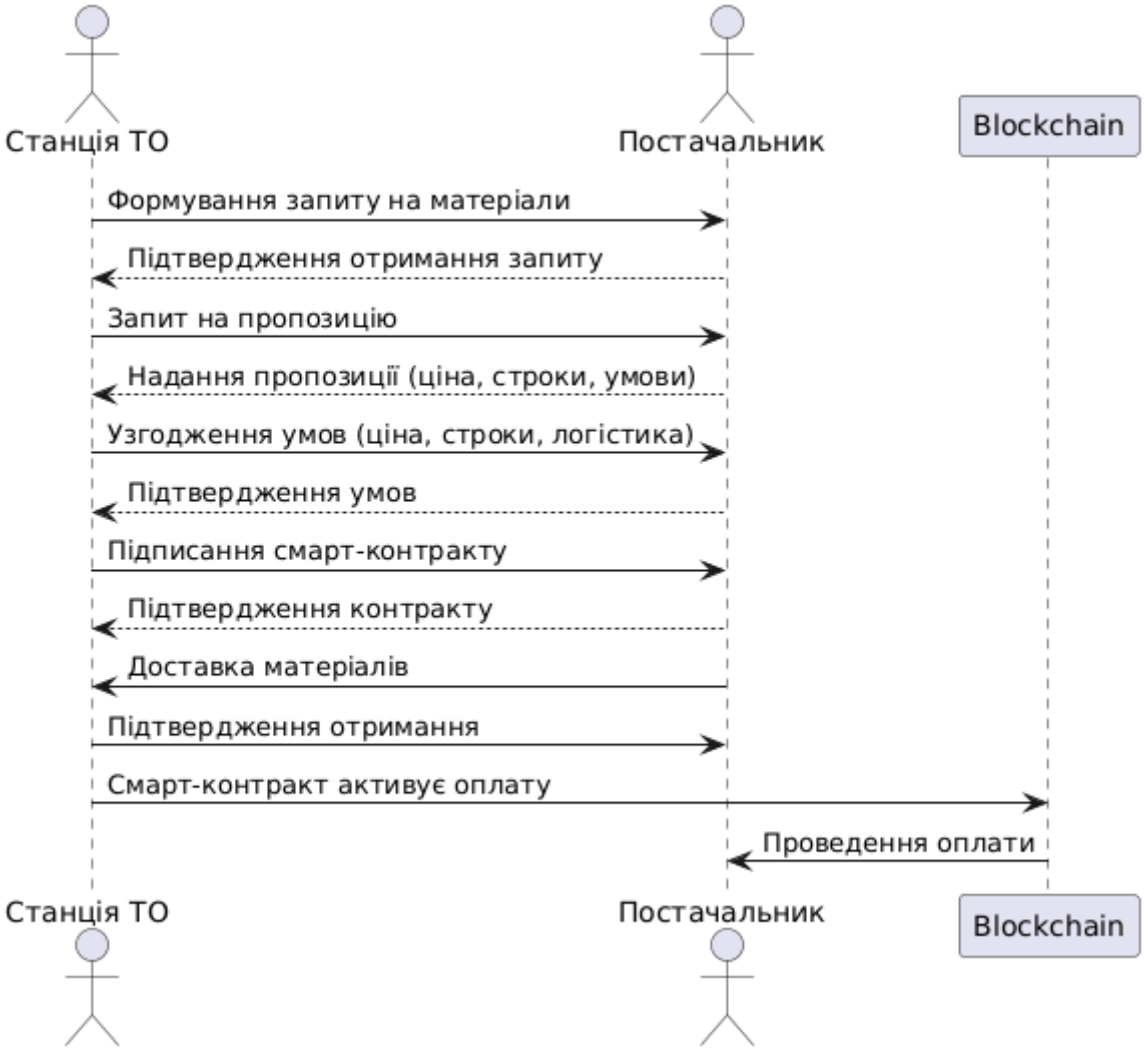

Рисунок 2.1 - UML-діаграма послідовності процесу взаємодії постачальників із замовниками матеріалів для СТО

Дана UML-діаграма послідовності демонструє високий рівень автоматизації процесу постачання товару (запасних частин) завдяки використанню блокчейн-технологій. Прозорість, надійність і автоматизація забезпечуються завдяки смарт-контрактам, які мінімізують ризики шахрайства та сприяють ефективному управлінню фінансами. Даний підхід може бути адаптований для інших сфер комерційної діяльності, що вимагають інтеграції постачальників та замовників у

розподіленій системі. Тому, впровадження блокчейн-технологій у процес управління фінансуванням СТО дозволяє досягти наступних переваг:

- прозорість, де усі транзакції зберігаються в незмінному вигляді, що підвищує рівень довіри між учасниками [32,33];

- автоматизація, де смарт-контракти виконують заздалегідь узгоджені умови, знижуючи адміністративні витрати;

- надійність, де децентралізована мережа знижує ризик втрати даних або їх несанкціонованої модифікації [32,33].

UML-діаграма послідовності (рис. 2.1) показує ключові етапи взаємодії між станцією технічного обслуговування (СТО), постачальником матеріалів та блокчейн-системою, яка забезпечує децентралізовану обробку фінансових транзакцій. Кожен учасник виконує свою роль у забезпеченні безперебійного постачання матеріалів і прозорості фінансових операцій. При цьому, використовуються наступні етапи взаємодії між станцією технічного обслуговування (СТО), постачальником матеріалів та блокчейн-системою:

- формування запиту на матеріали, коли СТО ініціює процес, формуючи запит на постачання необхідних матеріалів. Цей етап є критично важливим для забезпечення відповідності замовлення поточним потребам організації;

- підтвердження отримання запиту, де постачальник надсилає підтвердження отримання запиту, що забезпечує обізнаність сторін про початок процесу взаємодії;

- створення запиту на пропозицію, коли СТО надсилає постачальнику запит на комерційну пропозицію, яка включає інформацію про ціну, строки доставки та інші умови постачання;

- надання пропозиції, коли постачальник відповідає на запит, надаючи детальну інформацію про умови постачання. На цьому етапі критично важливо забезпечити прозорість і точність даних;

- узгодження умов, де СТО аналізує отриману пропозицію та узгоджує остаточні умови постачання з постачальником. Умови включають вартість, строки виконання, логістичні аспекти та спосіб оплати;

- підтвердження умов, коли після досягнення консенсусу постачальник підтверджує узгоджені умови, що дозволяє перейти до наступного етапу;

- підписання смарт-контракту, де обидві сторони укладають смарт-контракт у блокчейн-системі. Смарт-контракт автоматизує виконання умов угоди та мінімізує ризики людських помилок;

- підтвердження контракту, коли після підписання смарт-контракту постачальник підтверджує готовність до виконання замовлення;

- доставка матеріалів, коли постачальник виконує доставку матеріалів до СТО. На цьому етапі важливо забезпечити відповідність доставки узгодженим умовам;

- підтвердження отримання, де СТО підтверджує отримання матеріалів, що є тригером для ініціювання фінансової операції в блокчейн-системі;

- активація оплати смарт-контрактом, де смарт-контракт у блокчейн-системі автоматично ініціює оплату постачальнику після отримання підтвердження від СТО;

- проведення оплати де фінансова транзакція завершується, і кошти перераховуються постачальнику через блокчейн-систему.

Таким чином, впровадження сучасних технологій, таких як блокчейн, в управління фінансуванням СТО сприяє підвищенню ефективності фінансових операцій, забезпеченню прозорості взаємодії між постачальниками та замовниками, а також мінімізації операційних ризиків. Даний підхід може стати основою для створення автоматизованої системи фінансового управління, адаптованої до умов сучасного ринку.

2.2 Розробка функціональних вимог до програмного засобу

Програмний засіб для управління фінансуванням станцій технічного обслуговування має забезпечувати автоматизацію ключових процесів, пов’язаних із плануванням, обліком, моніторингом та оптимізацією фінансових операцій. Система повинна сприяти підвищенню ефективності фінансового управління, мінімізації витрат і забезпеченню прозорості фінансових потоків (рис. 2.2).

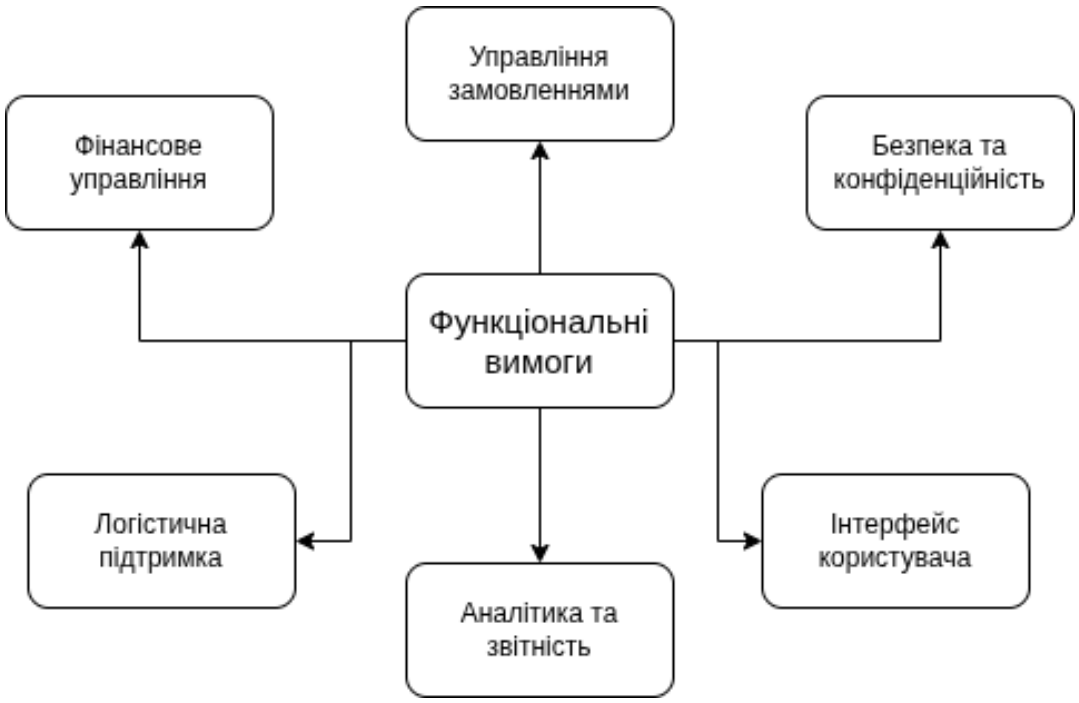

Рисунок 2.2 - Функціональні вимоги програмного засобу

Програмний засіб для управління замовленнями та постачанням має забезпечувати автоматизацію всіх ключових етапів процесу: від створення замовлення до завершення поставки. Основна мета такого інструменту - оптимізація ресурсів, підвищення ефективності взаємодії між постачальниками та замовниками, а також забезпечення прозорості всього процесу. Програмний засіб повинен надавати можливість автоматизованого формування замовлень на основі аналізу потреб, прогнозів та історичних даних станції технічного обслуговування (СТО). Всі замовлення повинні включати чітко сформульовані технічні характеристики матеріалів, їхню кількість, бажані строки доставки та додаткові вимоги (наприклад, сертифікати якості). Система має підтримувати редагування замовлень до моменту їхнього підтвердження постачальником. Система повинна забезпечувати двосторонню комунікацію з постачальниками, включаючи надсилання запитів, отримання комерційних пропозицій та узгодження умов.

Функціональні вимоги до програмного засобу для управління замовленнями та постачанням мають забезпечити ефективність усіх етапів взаємодії між СТО та постачальниками. Використання автоматизованих процесів, інтеграції з зовнішніми системами та сучасних аналітичних інструментів сприятиме оптимізації ресурсів, зниженню витрат та підвищенню якості постачання.

Таблиця 2.3 - Функціональні вимоги програмного засобу

Категорія	Функціональні вимоги
Управління замовленнями	- Автоматичне формування замовлень на основі потреб СТО. - Моніторинг статусу замовлень у реальному часі. - Інтеграція з платформами постачальників для обміну даними.
Фінансове управління	- Планування бюджету та прогнозування витрат. - Контроль руху коштів та запис усіх транзакцій. - Використання блокчейн-смарт-контрактів для автоматизації розрахунків.
Логістична підтримка	- Автоматичне планування логістичних маршрутів. - Відстеження доставки матеріалів у реальному часі. - Аналіз та оптимізація логістичних витрат.
Аналітика та звітність	- Модулі для аналізу витрат та оптимізації фінансових ресурсів. - Генерація звітів про фінансові операції. - Прогнозування фінансових потреб за допомогою машинного навчання.
Інтерфейс користувача	- Інтуїтивно зрозумілий, адаптивний дизайн для різних пристроїв. - Підтримка ролей та доступів для різних категорій користувачів.
Безпека та конфіденційність	- Шифрування даних для захисту фінансової інформації. - Підтримка функції аудиту транзакцій. - Резервне копіювання даних для запобігання втраті інформації.

Програмний засіб для фінансового управління станцій технічного обслуговування (СТО) має забезпечувати автоматизацію процесів, пов’язаних із плануванням, обліком, моніторингом і контролем фінансових операцій. Основною метою є підвищення ефективності використання фінансових ресурсів,

забезпечення прозорості операцій і мінімізація ризиків, пов’язаних із фінансовими розрахунками. Система повинна забезпечувати інструменти для створення детальних фінансових планів, які включають прогнозування доходів і витрат СТО.

Програмний засіб для фінансового управління СТО має забезпечувати автоматизацію всіх ключових фінансових процесів, включаючи планування, облік, моніторинг і звітність. Інтеграція блокчейн-технологій та алгоритмів прогнозування дозволяє підвищити прозорість, безпеку й ефективність управління фінансами. Такий інструмент сприятиме оптимізації фінансових потоків, зниженню ризиків та прийняттю обґрунтованих управлінських рішень.

Програмний засіб для логістичної підтримки станцій технічного обслуговування (СТО) повинен забезпечувати комплексну автоматизацію процесів управління логістикою, включаючи планування, моніторинг, аналіз і оптимізацію логістичних операцій. Основна мета полягає в підвищенні ефективності транспортування, зниженні витрат і забезпеченні своєчасного постачання матеріалів. Система повинна забезпечувати автоматичне планування оптимальних маршрутів доставки з урахуванням географічного розташування постачальника, замовника, дорожніх умов і строків доставки.

Функціональні вимоги до програмного засобу для логістичної підтримки спрямовані на підвищення ефективності управління транспортними та постачальницькими операціями. Інтеграція сучасних технологій, таких як GPS-моніторинг, машинне навчання та автоматизація процесів, забезпечує оптимізацію логістичних витрат, мінімізацію ризиків і своєчасність постачання матеріалів. Цей підхід сприяє підвищенню загальної продуктивності та конкурентоспроможності СТО.

Програмний засіб для аналітики та звітності повинен забезпечувати автоматизацію процесів збору, аналізу, обробки та представлення даних, що стосуються операційної діяльності, фінансів і логістики станції технічного обслуговування (СТО). Основна мета таких функцій полягає у створенні інструментів для прийняття обґрунтованих управлінських рішень, підвищення ефективності ресурсів та забезпечення прозорості операцій. Програмний засіб

повинен автоматично збирати дані з модулів фінансового управління, логістики та управління замовленнями для формування єдиної бази даних.

Програмний засіб для аналітики та звітності повинен забезпечувати комплексну автоматизацію процесів збору, аналізу та представлення даних. Використання сучасних технологій, таких як машинне навчання, інтерактивна візуалізація та хмарні сервіси, сприятиме покращенню прозорості, обґрунтованості управлінських рішень та підвищенню ефективності операційної діяльності.

Інтерфейс користувача є критично важливою складовою програмного засобу, оскільки забезпечує взаємодію користувачів із системою, ефективність виконання операцій і загальну зручність роботи з програмою. Основна мета функціональних вимог до інтерфейсу користувача - створення зручного, інтуїтивного, адаптивного та функціонального інтерфейсу, який відповідатиме сучасним стандартам розробки програмного забезпечення. Інтерфейс повинен забезпечувати легкий доступ до основних функцій програмного засобу через чітку та логічно структуровану систему меню.

Програмний засіб повинен забезпечувати інтуїтивний, адаптивний та функціональний інтерфейс користувача, який сприятиме зручності та ефективності виконання операцій. Інтеграція інструментів персоналізації, сучасного дизайну та високого рівня безпеки забезпечить комфорт і довіру користувачів до системи.

Безпека та конфіденційність є ключовими аспектами розробки програмного засобу для забезпечення захисту даних і запобігання несанкціонованому доступу до інформації. Основною метою функціональних вимог у цій сфері є створення надійного захисту конфіденційних даних користувачів, фінансової інформації та інших критично важливих компонентів системи.

З метою авторизації та аутентифікації виконується підтримка багаторівневої автентифікації, де програмний засіб повинен забезпечувати можливість використання двофакторної автентифікації (2FA) або біометричних даних для доступу до системи. Для цього існує система ролей та прав доступу, де є інструменти для налаштування рівнів доступу користувачів залежно від їхніх ролей (адміністратор, менеджер, оператор). Тому, доступ до функціональних модулів має бути обмеженим відповідно до встановлених прав. Під час контролю сесій

користувачів система повинна мати можливість автоматично завершувати неактивні сесії користувачів через заданий період часу.

Під час моніторингу та аудиту логування дій користувачів система повинна фіксувати всі дії користувачів (вхід, зміни даних, створення звітів тощо) для створення журналу подій. Система сповіщень про підозрілу активність може у разі виявлення аномальної активності (наприклад, багаторазових невдалих спроб входу) надсилати сповіщення адміністраторам безпеки. Для цього модулі для аналізу даних про безпеку повинні виявляти потенційні загрози та надавати рекомендації для їх усунення.

Управління даними виконується з метою обмеження доступу до конфіденційної інформації, де дані, які вважаються конфіденційними (наприклад, фінансові або особисті дані), повинні бути доступними лише для користувачів із відповідними правами. У разі використання даних для аналітики чи тестування система повинна забезпечувати їх анонімізацію, щоб уникнути ідентифікації користувачів.

Резервне копіювання та відновлення полягає в тому, що програмний засіб має підтримувати регулярне створення резервних копій із можливістю швидкого відновлення даних у разі їх втрати.

Захист від атак типу SQL Injection та XSS необхідно тоді, коли система повинна бути захищена від спроб несанкціонованого доступу через уразливості у формах введення даних. Для захист від DoS/DDoS атак виконується інтеграція інструментів виявлення та запобігання атак, що спрямовані на порушення доступності системи. Для обмеження на кількість запитів система повинна мати функцію обмеження кількості запитів, які можуть надходити від одного користувача або IP-адреси за певний проміжок часу.

Регулярні оновлення полягають в автоматичному оновленні, де програмний засіб повинен підтримувати автоматичне оновлення безпекових компонентів для захисту від нових загроз. Також, регулярне оновлення системи має забезпечувати виправлення відомих уразливостей та інтеграцію сучасних методів захисту.

Відповідність стандартам передбачає сертифікацію, де програмний засіб повинен відповідати міжнародним стандартам безпеки, таким як ISO/IEC 27001,

PCI DSS (для обробки фінансових даних), GDPR (у разі роботи з персональними даними). Тоді, внутрішні політики безпеки будуть реалізовані у підтримці інтеграції з політиками безпеки організації, включаючи контроль доступу, резервування даних і процедури відновлення. Тому, функціональні вимоги програмного засобу щодо безпеки та конфіденційності повинні забезпечити захист даних користувачів і фінансових операцій від несанкціонованого доступу, атак та інших загроз. Використання сучасних методів автентифікації, шифрування, моніторингу та регулярного оновлення дозволить створити надійну систему, яка відповідатиме найвищим стандартам безпеки.

Таким чином, розробка програмного засобу для управління фінансуванням СТО повинна забезпечити високу ефективність фінансових операцій, мінімізувати ризики, пов’язані з фінансуванням і постачанням, а також підвищити прозорість і надійність управління фінансовими ресурсами. Інтеграція сучасних технологій, таких як блокчейн та машинне навчання, забезпечить конкурентні переваги і дозволить адаптувати систему до швидко змінюваних умов ринку.

2. 3 Проектування основних та додаткових модулів програмного засобу

Програмний засіб для управління фінансуванням станцій технічного обслуговування (СТО) повинен складатися з інтегрованих модулів, що виконують специфічні функції, які пов’язані з автоматизацією фінансових, логістичних та управлінських процесів (рис. 2.3). Кожен модуль виконує певну роль, що спрямована на забезпечення ефективності, прозорості та безпеки операцій.

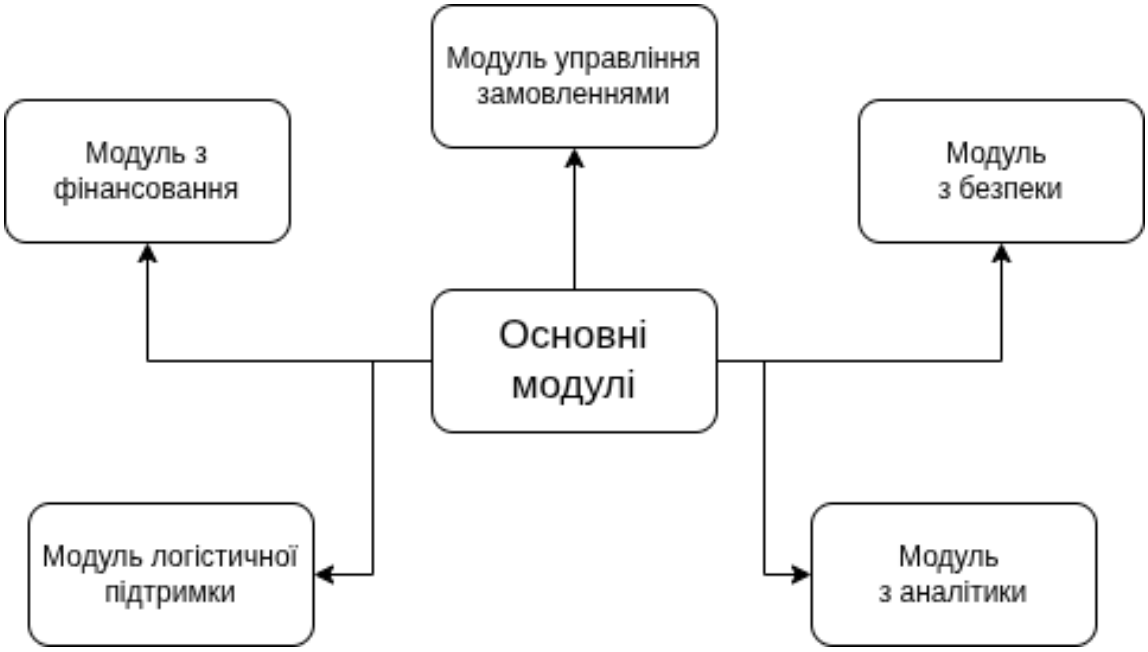

Рисунок 2.3 - Основні модулі програмного засобу

Модуль управління замовленнями є ключовим компонентом програмного засобу, що забезпечує автоматизацію процесів, пов’язаних із створенням, обробкою, моніторингом та узгодженням замовлень. Його основною метою є підвищення ефективності взаємодії між станціями технічного обслуговування (СТО) та постачальниками, а також оптимізація управлінських процесів, пов’язаних із забезпеченням ресурсами (табл. 2.3).

Цей модуль сприяє інтеграції інформаційних потоків і усуненню ручної обробки даних, що знижує ймовірність помилок і підвищує точність управлінських рішень. Він об’єднує функціональні можливості для автоматичного формування замовлень, обліку отриманих пропозицій від постачальників і відстеження стану виконання кожного замовлення.

Модуль управління замовленнями є важливим інструментом для забезпечення безперебійної роботи СТО, оскільки він дозволяє уникати затримок у постачанні матеріалів, мінімізує витрати та сприяє підвищенню продуктивності процесів постачання. Його впровадження забезпечує узгодженість роботи всіх учасників процесу та створює основу для ефективного управління ресурсами.

Фінансовий модуль є одним із базових компонентів програмного засобу для управління фінансуванням станцій технічного обслуговування (СТО). Його

основною метою є автоматизація процесів планування, обліку, моніторингу та аналізу фінансових операцій, що сприяє ефективному використанню ресурсів і підвищенню прозорості фінансових потоків.

Таблиця 2.4 - Опис функціонального призначення основних модулів

Найменування модуля	Опис функціонального призначення
Модуль управління замовленнями	Автоматизація створення, обробки та моніторингу замовлень.
Фінансовий модуль	Управління фінансовими транзакціями, облік витрат і доходів.
Модуль логістичної підтримки	Управління доставкою матеріалів і оптимізація маршрутів.
Аналітичний модуль	Збір, обробка та аналіз даних для підтримки прийняття рішень.
Модуль безпеки	Захист даних, авторизація користувачів, шифрування.

Модуль забезпечує інтегровану платформу для управління всіма аспектами фінансів, включаючи планування бюджету, контроль витрат, облік доходів, проведення транзакцій та формування фінансової звітності. Його впровадження дозволяє мінімізувати ризики, пов’язані з фінансовими операціями, забезпечуючи їхню прозорість і точність.

Фінансовий модуль є стратегічним інструментом, який сприяє ефективному управлінню фінансами СТО, підвищенню точності планування та забезпеченню прозорості фінансових операцій. Його використання дозволяє зменшити операційні витрати, мінімізувати ризики та покращити загальний фінансовий стан організації.

Модуль логістичної підтримки є важливою складовою програмного засобу для управління фінансуванням станцій технічного обслуговування (СТО), спрямованою на автоматизацію та оптимізацію всіх аспектів логістичних операцій. Його основною метою є забезпечення своєчасної доставки матеріалів і ресурсів, зниження витрат на транспортування та підвищення ефективності логістичних процесів.

Цей модуль інтегрує управлінські та технологічні функції, які дозволяють здійснювати планування, моніторинг і контроль перевезень, а також забезпечувати прозорість і координацію між усіма учасниками логістичного ланцюга.

До переваги використання даного модуля відноситься підвищення ефективності доставки та скорочення витрат, забезпечення прозорості логістичних процесів, мінімізація ризиків затримок і втрат під час транспортування, покращення взаємодії між постачальниками та СТО.

Модуль логістичної підтримки забезпечує стратегічну перевагу для СТО, сприяючи підвищенню операційної ефективності та зниженню витрат. Його використання дозволяє оптимізувати логістичні процеси, забезпечити своєчасну доставку ресурсів і підвищити рівень задоволеності клієнтів.

Аналітичний модуль є ключовим елементом програмного засобу для управління фінансуванням станцій технічного обслуговування (СТО), що відповідає за збір, обробку, аналіз і візуалізацію даних. Його основна мета - забезпечення керівників і користувачів системи інструментами для обґрунтованого прийняття рішень на основі даних про фінанси, логістику, замовлення та інші аспекти операційної діяльності.

Модуль дозволяє аналізувати великі обсяги інформації, виявляти ключові показники ефективності (KPI), прогнозувати майбутні потреби й оцінювати ризики. Завдяки інтеграції з іншими модулями програмного засобу аналітичний модуль сприяє створенню комплексного уявлення про всі процеси організації.

До переваги використання цього модуля належить прийняття обґрунтованих рішень, що забезпечує доступ до детальної та точної інформації для стратегічного планування. Це дозволяє отримувати вичерпну інформацію про всі аспекти діяльності організації та виявлення проблемних зон, що допомагає виявляти неефективність і пропонувати рішення для їх усунення. Прогнозування ризиків у цьому модулі підвищує готовність організації до реагування на потенційні загрози (табл. 2.3).

Тому, аналітичний модуль є критично важливим інструментом для забезпечення прозорості, ефективності та стратегічного управління діяльністю СТО. Його функціонал дозволяє перетворювати дані на цінну інформацію, яка

сприяє оптимізації ресурсів, мінімізації ризиків і підвищенню загальної продуктивності організації.

Інший модуль безпеки є ключовим компонентом програмного засобу, спрямованим на забезпечення конфіденційності, цілісності та доступності даних, а також на запобігання несанкціонованому доступу до системи. Його основною метою є створення надійного середовища для роботи користувачів і забезпечення захисту інформації від внутрішніх та зовнішніх загроз.

Цей модуль інтегрує сучасні технології автентифікації, шифрування, моніторингу та контролю доступу, що дозволяє ефективно управляти інформаційною безпекою. Його впровадження є критично важливим для дотримання міжнародних стандартів захисту даних і запобігання втратам, які можуть виникнути внаслідок кіберзагроз.

Переваги використання цього модуля відноситься захист даних, що забезпечує високий рівень захисту інформації від втрат, витоків і несанкціонованого доступу. Зменшення ризиків запобігає порушенням у роботі системи через внутрішні або зовнішні загрози. В цьому випадку підвищення довіри гарантує користувачам безпечний доступ і зберігання даних. Відповідність законодавчим вимогам допомагає організації уникати штрафів за порушення стандартів безпеки.

Тому, модуль безпеки є фундаментальним елементом програмного засобу, що забезпечує збереження даних і безперебійну роботу системи. Його функціональні можливості спрямовані на захист інформації від загроз, мінімізацію ризиків і дотримання вимог інформаційної безпеки. Використання цього модуля підвищує загальну надійність системи та створює умови для безпечної взаємодії між користувачами.

Модуль інтеграції з постачальниками є важливим компонентом програмного засобу, призначеним для автоматизації процесів взаємодії між станціями технічного обслуговування (СТО) та їхніми постачальниками. Його основна мета - забезпечити ефективний обмін даними, підвищити прозорість комунікацій, оптимізувати процеси постачання та скоротити час, необхідний для обробки запитів і отримання комерційних пропозицій.

Цей модуль сприяє інтеграції зовнішніх інформаційних систем постачальників із внутрішніми модулями СТО, що дозволяє в реальному часі отримувати актуальну інформацію про доступність матеріалів, строки доставки, ціни та інші умови постачання. Завдяки автоматизації процесів комунікації модуль дозволяє мінімізувати людський фактор, що підвищує точність і швидкість прийняття рішень.

Модуль інтеграції з постачальниками забезпечує ефективну взаємодію між СТО та постачальниками, підвищуючи оперативність і прозорість усіх етапів співпраці. Його впровадження дозволяє автоматизувати процеси обміну даними, знижуючи адміністративне навантаження та забезпечуючи своєчасність і якість постачань. Завдяки інтеграції з іншими модулями програмного засобу цей модуль сприяє загальній ефективності управління ресурсами (табл. 2.4).

Таблиця 2.5 - Опис функціонального призначення додаткових модулів

Найменування модуля	Опис функціонального призначення
Модуль інтеграції з постачальниками	Синхронізація із зовнішніми платформами та обмін даними.
Модуль персоналізації	Налаштування інтерфейсу та функціоналу під потреби користувачів.
Модуль навчання та підтримки	Допомога користувачам для освоєння системи та вирішення проблем.
Модуль управління ризиками	Оцінка та мінімізація ризиків, прогнозування затримок, резервні плани.

Модуль персоналізації є важливим компонентом програмного засобу, що забезпечує адаптацію інтерфейсу та функціоналу системи під індивідуальні потреби користувачів. Його основною метою є підвищення зручності та ефективності роботи користувачів шляхом налаштування параметрів відображення, функціональних можливостей і доступу до інформації.

Цей модуль дозволяє кожному користувачу створювати унікальний робочий простір із врахуванням його ролі в організації, професійних потреб і рівня доступу

до даних. Інтеграція модуля персоналізації в програмний засіб сприяє підвищенню продуктивності роботи користувачів і зменшенню ймовірності помилок за рахунок спрощення доступу до найбільш необхідних функцій і даних.

Модуль персоналізації є інструментом, який дозволяє забезпечити максимальну адаптацію програмного засобу до потреб кожного користувача та організації в цілому. Його функціональні можливості сприяють підвищенню ефективності, зручності й точності роботи з системою, створюючи інтуїтивно зрозумілий і комфортний робочий простір для всіх категорій користувачів.

Модуль навчання та підтримки є інтегрованою складовою програмного засобу, призначеною для забезпечення користувачів необхідними знаннями, інструментами та ресурсами для ефективної роботи з системою. Його основною метою є зменшення навчальної кривої, підтримка користувачів у вирішенні технічних питань і забезпечення їхньої впевненості в роботі з програмним продуктом.

Цей модуль пропонує користувачам інтерактивні навчальні матеріали, інструкції, довідкову інформацію та функціонал для оперативного зв’язку з технічною підтримкою. Його впровадження сприяє підвищенню рівня задоволеності користувачів, зменшенню кількості помилок під час роботи з системою та мінімізації потреби в зовнішній технічній допомозі.

До переваги використання цього модуля відноситься зниження витрат часу на навчання, що надає доступ до структурованих і зрозумілих матеріалів для самостійного освоєння системи. Підвищення продуктивності забезпечує користувачів знаннями для ефективного використання всіх функцій програмного засобу. Зменшення кількості помилок допомагає уникати поширених проблем завдяки інтерактивним підказкам і довідковим матеріалам. Полегшення адаптації нових користувачів сприяє швидкому введення нових співробітників у робочий процес через навчальні програми.

Тому, модуль навчання та підтримки є незамінним інструментом для створення комфортного середовища роботи користувачів із програмним засобом. Його функціональні можливості сприяють підвищенню рівня компетенцій користувачів, зменшенню кількості помилок і забезпеченню ефективної роботи з

системою. Інтеграція цього модуля дозволяє організації оптимізувати навчальні процеси, знизити навантаження на службу підтримки та покращити загальну продуктивність.

Модуль управління ризиками є ключовим компонентом програмного засобу, призначеним для ідентифікації, аналізу, оцінки та мінімізації потенційних ризиків, пов’язаних із фінансовими, логістичними та операційними процесами станцій технічного обслуговування (СТО). Основною метою модуля є забезпечення стабільної та ефективної діяльності організації шляхом своєчасного виявлення проблемних зон, прогнозування можливих загроз і розробки заходів для їхнього усунення.

Цей модуль інтегрується з іншими функціональними компонентами програмного засобу, забезпечуючи комплексний підхід до управління ризиками на всіх етапах операційної діяльності. Його впровадження сприяє підвищенню стійкості організації до зовнішніх і внутрішніх викликів.

Модуль управління ризиками забезпечує комплексний підхід до оцінки та мінімізації ризиків, що виникають під час діяльності СТО. Його функціональні можливості сприяють підвищенню ефективності роботи організації, зниженню втрат і забезпеченню стабільності в умовах невизначеності. Використання цього модуля допомагає організації проактивно реагувати на загрози, підвищуючи її конкурентоспроможність та довіру з боку партнерів.

Таким чином, основні та додаткові модулі програмного засобу створюють комплексну екосистему для автоматизації управління фінансуванням СТО. Інтеграція ключових функцій із можливістю розширення додатковими модулями забезпечує універсальність, ефективність і надійність програмного рішення. Це дозволяє підвищити продуктивність, знизити ризики та створити прозору систему управління.

3 РОЗРОБКА МОДУЛІВ ПРОГРАМНОГО ЗАСОБУ

3.1 Основні принципи розробки програмного засобу на основі блокчейну

Розглянемо основні принципи роботи програмного засобу на основі блокчейну. Ці принципи засновані на ключових етапах взаємодії між станцією технічного обслуговування (СТО), постачальником матеріалів та блокчейн-системою. При цьому, блокчейн-системою забезпечує децентралізовану обробку фінансових транзакцій, де кожен учасник виконує свою роль у забезпеченні надійного постачання матеріалів і прозорості фінансових операцій. Для цього, виконується ціла низька заходів, яка забезпечує потрібне надійне постачання матеріалів та фінансування наступним чином.

Формування запиту на матеріали є початковим і надзвичайно важливим етапом у процесі забезпечення матеріально-технічної бази станцій технічного обслуговування (СТО). На цьому етапі здійснюється аналіз поточних потреб організації, враховуючи прогнозовані обсяги роботи, заплановане технічне обслуговування та ремонт, а також очікувані терміни виконання завдань. Основною метою цього етапу є формування детального та обґрунтованого запиту, який повинен містити інформацію про кількісні та якісні параметри необхідних матеріалів, включаючи типи деталей, витратні матеріали, їхню сумісність із обладнанням, а також технічні характеристики.

Критична важливість цього етапу полягає у забезпеченні відповідності запиту фактичним потребам СТО, що дозволяє уникнути перевитрат фінансових ресурсів, мінімізувати ризики придбання невідповідних матеріалів та уникнути простоїв у роботі через недостачу необхідних компонентів. Зокрема, на цьому етапі можуть застосовуватися сучасні інформаційні системи для збору даних про наявні ресурси, аналізу їхньої достатності та прогнозування потреб, що базується на історичних даних і поточних показниках. Тому, формування запиту на матеріали виконує роль стартового механізму, який задає напрямок усім наступним етапам взаємодії з постачальниками, забезпечуючи точність, ефективність і оптимальність процесу постачання. Ефективно сформований запит є основою для досягнення

загальної мети організації - забезпечення високої якості надання послуг при оптимальних витратах.

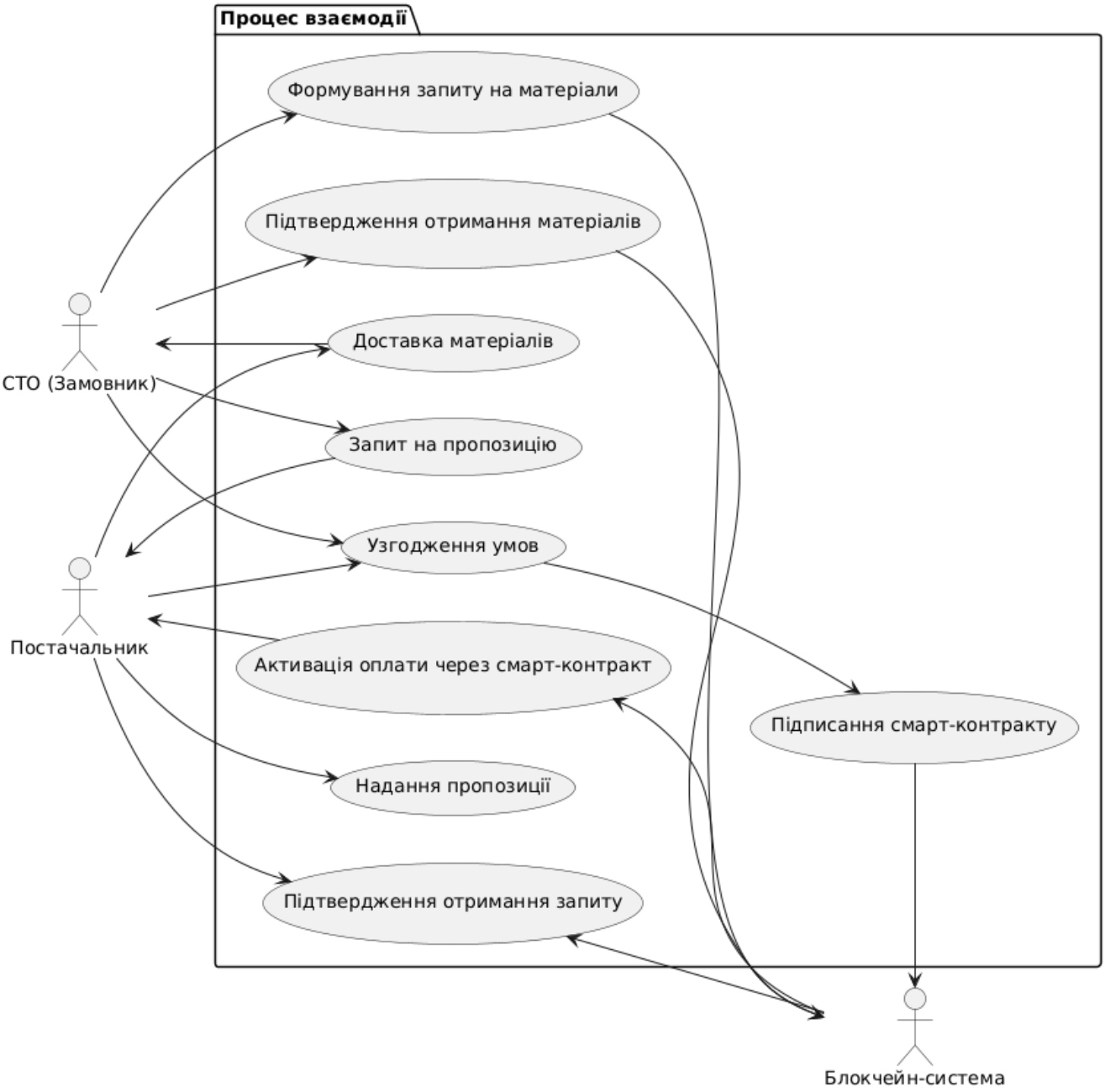

Рисунок 3.1 - UML-діаграма використання процесу взаємодії постачальників із замовниками матеріалів для СТО за допомогою технології блокчейна

Підтвердження отримання запиту є ключовим етапом, який формалізує початок взаємодії між постачальником і замовником у межах процесу постачання матеріалів для станції технічного обслуговування (СТО). На цьому етапі постачальник, отримавши запит на постачання матеріалів, надсилає офіційне

підтвердження його отримання. Це підтвердження слугує гарантією того, що запит був успішно доставлений, опрацьований та прийнятий до подальшого розгляду.

Функціональне значення цього етапу полягає у забезпеченні обізнаності обох сторін щодо поточного стану процесу та наявності спільного розуміння його подальших дій. Формалізація підтвердження дозволяє уникнути непорозумінь, пов'язаних із можливими затримками у комунікаціях або втратами інформації на етапі її передачі. Крім того, вона встановлює часову межу, після якої замовник може очікувати відповіді у вигляді пропозиції або іншої реакції на запит.

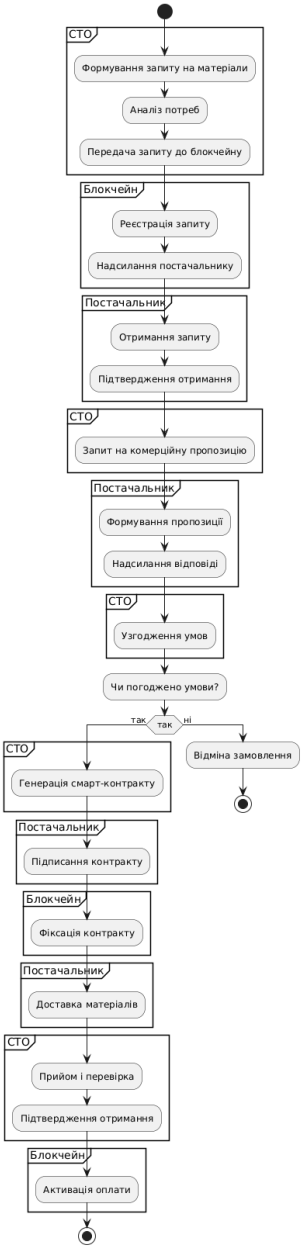

Рисунок 3.2 - UML-діаграма активності процесу взаємодії постачальників із замовниками матеріалів для СТО

В рамках проєктування та розробки програмного засобу для управління фінансуванням станцій технічного обслуговування (СТО) із застосуванням технології блокчейн, було сформульовано алгоритм взаємодії між ключовими суб'єктами постачання — замовником (СТО), постачальником та децентралізованою інформаційною інфраструктурою, що базується на смарт-контрактах. Алгоритм реалізує поетапну координацію дій усіх сторін у процесі формування, узгодження та виконання замовлень на матеріально-технічні ресурси, забезпечуючи підвищену прозорість, достовірність та автоматизованість усіх процедур.

Процес починається з ініціації запиту на матеріали, яка виконується представниками СТО (рис. 3.2). На цьому етапі аналізуються наявні виробничі потреби, поточний стан ресурсного забезпечення, заплановані ремонти та інші параметри, що впливають на обсяг і специфікацію майбутнього постачання. У результаті формується обґрунтований запит, який містить перелік необхідних матеріалів, їх кількість, технічні характеристики, строки доставки та інші логістичні вимоги. Після цього відповідна інформація передається у блокчейн-систему, де відбувається її реєстрація у вигляді транзакції, що гарантує незмінність і доступність запиту для всіх зацікавлених сторін.

Блокчейн-система, отримавши запит, автоматично спрямовує його постачальнику, який є зареєстрованим учасником розподіленої мережі. Постачальник підтверджує отримання запиту, що дозволяє забезпечити контроль за коректністю доставки інформації, а також підтверджує свою зацікавленість у подальшій взаємодії. Наступним кроком замовник надсилає офіційний запит на комерційну пропозицію, який включає параметри очікуваного постачання та уточнює можливі умови договору. У відповідь постачальник формує та надсилає пропозицію, яка детально описує вартість матеріалів, строки виконання, способи оплати, транспортування, наявні гарантії, а також додаткові послуги.

У подальшому виконується критично важливий етап - узгодження умов співпраці (рис. 3.2). Представники СТО аналізують отриману пропозицію, проводять порівняння з іншими альтернативами (у разі наявності), а також здійснюють обговорення ключових аспектів постачання, таких як ціна, строки

доставки, формат логістичних операцій, юридичні зобов’язання сторін. Якщо запропоновані умови є прийнятними, сторони переходять до процедури формалізації домовленостей через механізм смарт-контрактів. У межах цієї процедури замовник ініціює створення контракту, в якому зафіксовані всі узгоджені параметри. Після цього постачальник підписує контракт за допомогою криптографічного підпису, що забезпечує автентичність його згоди.

Після підписання контракт фіксується у блокчейн-системі, що гарантує його незмінність, публічну доступність та захищеність від будь-яких втручань. Контракт набуває статусу активного і слугує юридичною основою для подальшої взаємодії. Постачальник здійснює транспортування матеріалів відповідно до зафіксованих строків і логістичних вимог, використовуючи при цьому оптимальні маршрути, транспортні засоби, а також надає весь комплект супровідної документації. У свою чергу, замовник (СТО) виконує перевірку факту доставки: здійснюється кількісний і якісний аналіз поставлених ресурсів, перевіряється відповідність сертифікатів та супровідних документів вимогам, зафіксованим у контракті.

Після успішного завершення перевірки СТО підтверджує отримання матеріалів (рис. 3.2), що також реєструється у блокчейн у вигляді незмінної транзакції. Цей факт слугує тригером для фінансової взаємодії: смарт-контракт автоматично ініціює виконання оплати, що може реалізовуватися у вигляді банківського переказу або криптовалютної транзакції, залежно від специфіки системи. Таким чином, платіжна операція виконується безпосередньо на основі фактологічного підтвердження, без необхідності залучення третіх сторін, нотаріальних сервісів або ручного втручання.

Загалом, запропонований алгоритм (рис. 3.2) дозволяє реалізувати високорівневу автоматизацію, прозорість і достовірність процесу постачання матеріалів для станцій технічного обслуговування. Завдяки інтеграції блокчейн-технологій забезпечується зменшення ризиків шахрайства, спрощується аудит, знижується навантаження на адміністративні ресурси. Крім того, кожен етап взаємодії зберігається у розподіленому реєстрі, що дозволяє забезпечити відтворюваність та перевірку історії операцій у разі виникнення спорів або необхідності ретроспективного аналізу.

Таким чином, реалізований алгоритм (рис. 3.2) описує управління ланцюгом постачання, який поєднує в собі концепції цифрової довіри, децентралізованої реєстрації транзакцій, автоматизації виконання угод і прозорості процедур закупівлі. Його впровадження дозволяє досягти стратегічної мети — створити ефективну, безпечну та економічно доцільну інфраструктуру фінансування технічного обслуговування на основі цифрових інтелектуальних контрактів.

В сучасних умовах для підвищення ефективності цього процесу можуть застосовуватися автоматизовані системи управління замовленнями, які забезпечують швидке та точне підтвердження запиту за допомогою електронної пошти, спеціалізованих платформ або інтегрованих інформаційних систем. Використання таких технологій підвищує оперативність і прозорість взаємодії між сторонами (табл. 3.1).

Отже, підтвердження отримання запиту виконує важливу роль у забезпеченні прозорості та структурованості процесу постачання. Воно створює основу для подальших етапів співпраці, спрямованих на ефективне задоволення потреб замовника.

Запит на пропозицію є важливим етапом у процесі організації постачання матеріалів для станції технічного обслуговування (СТО), що передбачає офіційне звернення замовника до постачальника з метою отримання детальної комерційної пропозиції. Цей етап виконує функцію ініціації переговорного процесу, створюючи інформаційну основу для ухвалення рішень щодо подальшої співпраці.

У запиті на пропозицію замовник чітко формулює свої вимоги до обсягів, характеристик і якості матеріалів, які необхідно постачити. Крім того, обов'язковим компонентом запиту є специфікація очікуваних умов постачання, таких як вартість матеріалів, строки їхньої доставки, умови транспортування, спосіб оплати, а також можливі додаткові послуги, які можуть знадобитися в межах співпраці. У разі використання цифрових платформ або автоматизованих систем управління постачанням, запит на пропозицію може включати стандартизовані формати документів, що спрощує подальшу комунікацію.

Таблиця 3.1 - Опис основних етапів взаємодії постачальників із замовниками матеріалів для СТО

Етап	Опис
Формування запиту на матеріали	Аналіз потреб СТО, підготовка запиту з деталями про кількість, якість та технічні характеристики матеріалів.
Підтвердження отримання запиту	Постачальник підтверджує отримання запиту, що гарантує його обробку та початок взаємодії.
Запит на пропозицію	Замовник запитує у постачальника комерційну пропозицію з деталізацією умов постачання.
Надання пропозиції	Постачальник надає комерційну пропозицію з вартістю, строками постачання та іншими деталями.
Узгодження умов	Обговорення та погодження ключових аспектів співпраці: вартість, строки, логістика, умови оплати.
Підписання смарт-контракту	Формалізація умов у блокчейн-смарт-контракті, перевірка даних і електронне підписання.
Доставка матеріалів	Транспортування матеріалів постачальником з дотриманням умов контракту.
Підтвердження отримання матеріалів	Перевірка відповідності матеріалів умовам контракту, формалізація факту отримання.
Активація оплати через смарт-контракт	Смарт-контракт автоматично ініціює фінансову транзакцію після підтвердження отримання матеріалів.

Основною метою цього етапу є отримання від постачальника повної та обґрунтованої інформації, необхідної для проведення аналізу й порівняння пропозицій від кількох потенційних постачальників. Це дозволяє СТО приймати економічно обґрунтовані рішення, базуючись на таких критеріях, як конкурентоспроможність цін, відповідність матеріалів технічним вимогам, а також репутація постачальника.

Таким чином, запит на пропозицію є інструментом, що забезпечує прозорість і структурованість процесу постачання. Він створює умови для раціонального

вибору постачальника, сприяючи ефективному використанню фінансових ресурсів і зниженню ризиків, пов'язаних із постачанням матеріалів. Крім того, запит на пропозицію є основою для подальших переговорів і узгодження умов співпраці між сторонами.

Надання пропозиції є ключовим етапом процесу постачання, на якому постачальник формує відповідь на запит замовника, надаючи детальну комерційну пропозицію. Ця пропозиція містить повну інформацію про умови постачання, що включають вартість матеріалів, строки їхньої доставки, технічні характеристики, а також специфіку логістичних процесів і додаткових послуг, які можуть бути надані. Мета цього етапу полягає у створенні максимально прозорої та точної комунікації, яка слугуватиме основою для ухвалення замовником обґрунтованого рішення.

У процесі формування пропозиції постачальник враховує всі вимоги, зазначені у запиті, забезпечуючи їх відповідність технічним та економічним параметрам, необхідним для задоволення потреб замовника. Пропозиція також повинна містити докладні розрахунки, що демонструють обґрунтованість цінової політики, а також чіткі часові рамки виконання замовлення. У разі необхідності, пропозиція може включати альтернативні варіанти матеріалів або послуг, які здатні оптимізувати вартість або підвищити якість постачання.

Критична важливість цього етапу полягає у забезпеченні прозорості даних та уникненні будь-яких двозначностей, які можуть призвести до непорозумінь або конфліктів у подальшій взаємодії. Для досягнення цієї мети постачальник повинен використовувати стандартизовані форми документів та засоби автоматизації, які підвищують точність і знижують ризики людських помилок.

Таким чином, надання пропозиції виконує функцію забезпечення відкритої комунікації між сторонами та створення бази для проведення переговорів і узгодження умов співпраці. Прозорість, повнота й точність наданої інформації на цьому етапі є фундаментом для подальшої ефективності та продуктивності процесу постачання, сприяючи досягненню цілей як замовника, так і постачальника.

Узгодження умов є критичним етапом процесу постачання, який спрямований на досягнення консенсусу між станцією технічного обслуговування (СТО) та постачальником щодо основних параметрів майбутньої співпраці. На

цьому етапі замовник детально аналізує отриману комерційну пропозицію, порівнюючи її з альтернативними варіантами (у разі наявності) та оцінюючи відповідність заявлених умов своїм потребам і стратегічним цілям. Процес узгодження умов передбачає обговорення та уточнення ключових аспектів, зокрема:

- вартості, де аналізується загальна сума пропозиції, її відповідність ринковим умовам і бюджетним обмеженням замовника. При цьому можуть розглядатися знижки, бонуси або інші форми економічних стимулів;

- строки виконання, де узгоджується графік постачання, включаючи точні дати поставок і їх відповідність планам діяльності СТО. Надання реалістичних строків виконання є важливим для забезпечення безперервності роботи станції;

- логістичні аспекти, де визначаються маршрути, способи транспортування, відповідальність за доставку, а також ризики, пов’язані з транспортуванням і зберіганням матеріалів;

- спосіб оплати, де обговорюються фінансові умови, включаючи валюту розрахунків, можливість поетапної оплати або використання авансових платежів. Також можуть бути узгоджені умови оплати через смарт-контракти у разі використання блокчейн-технологій.

Особливу увагу на цьому етапі приділяється забезпеченню прозорості та чіткості узгоджених умов, що дозволяє уникнути потенційних конфліктів у майбутньому. Для цього сторони можуть використовувати стандартизовані договори або автоматизовані системи управління контрактами, які забезпечують точну документацію всіх домовленостей.

Таким чином, узгодження умов відіграє ключову роль у формуванні ефективного партнерства між СТО та постачальником. Чітко визначені та погоджені умови є основою для укладення контракту, який слугує юридичним і практичним інструментом реалізації спільних домовленостей. Ефективність цього етапу безпосередньо впливає на загальну продуктивність процесу постачання та здатність обох сторін досягати своїх цілей.

Підтвердження умов є заключним кроком у процесі узгодження взаємодії між замовником і постачальником, що формалізує досягнутий консенсус щодо

параметрів постачання. На цьому етапі постачальник офіційно підтверджує свою згоду з узгодженими умовами, що забезпечує взаєморозуміння та готовність до реалізації домовленостей. Цей етап є надзвичайно важливим для забезпечення впорядкованості та прозорості наступних операцій.

Процес підтвердження умов може включати такі дії:

- формалізація зобов'язань, де постачальник надає письмове підтвердження або електронний документ, який засвідчує згоду з умовами, такими як вартість, строки виконання, логістичні аспекти, спосіб оплати та інші деталі, узгоджені в попередніх етапах;

- підтвердження технічної відповідності, де у разі потреби, постачальник також може надати додаткову інформацію, що підтверджує технічну відповідність матеріалів вимогам замовника, наприклад, сертифікати якості або технічні специфікації;

- використання автоматизованих систем, де у сучасних умовах підтвердження умов часто виконується за допомогою інтегрованих цифрових платформ або спеціалізованого програмного забезпечення, що дозволяє прискорити процес і знизити ризик людських помилок.

Ключовим завданням цього етапу є забезпечення юридичної та практичної визначеності, яка створює основу для подальшої реалізації контракту. У разі використання блокчейн-технологій, підтвердження умов може бути інтегрованим у смарт-контракти, які автоматично фіксують досягнуті домовленості та забезпечують їх дотримання.

Таким чином, підтвердження умов відіграє важливу роль у завершенні процесу узгодження та підготовці до наступних етапів співпраці. Чітке та офіційно зафіксоване підтвердження гарантує відповідність очікувань сторін, підвищує довіру між учасниками та сприяє ефективності загального процесу постачання.

Підписання смарт-контракту є важливим етапом, який формалізує домовленості між станцією технічного обслуговування (СТО) та постачальником, забезпечуючи юридичну та технічну основу для виконання угоди. Смарт-контракт представляє собою програмний код, інтегрований у блокчейн-систему, який автоматично виконує заздалегідь узгоджені умови угоди без необхідності участі

третіх сторін. Цей підхід забезпечує високий рівень надійності, прозорості та ефективності взаємодії між сторонами.

Процес підписання смарт-контракту включає кілька ключових кроків, які полягають в наступному.

1. Формалізація умов угоди, де всі умови співпраці, включаючи вартість матеріалів, строки постачання, логістичні деталі та способи оплати, кодуються в структурі смарт-контракту. Це забезпечує точне відображення досягнутих домовленостей.

2. Перевірка та погодження, де обидві сторони перевіряють коректність даних, внесених до смарт-контракту, гарантуючи відповідність закладених параметрів їхнім домовленостям. На цьому етапі важливо уникнути неточностей, які можуть вплинути на виконання угоди.

3. Підписання контракту, що здійснюється шляхом електронного підтвердження з використанням унікальних цифрових підписів сторін. Цей процес виконується у захищеному середовищі блокчейн-системи, що мінімізує ризик підробки або втручання третіх сторін.

4. Реєстрація у блокчейні, де після підписання контракту його дані зберігаються у розподіленій базі даних блокчейн-системи, що гарантує незмінність і доступність інформації для всіх учасників.

До переваги використання смарт-контрактів відносяться наступні чинникі:

- автоматизація виконання угоди, де смарт-контракт автоматично ініціює виконання умов, таких як оплата після отримання товару, що знижує потребу в адміністративному втручанні;

- прозорість, де всі транзакції та дії фіксуються у блокчейні, забезпечуючи максимальну прозорість для сторін;

- мінімізація людських помилок, де використання автоматизованих алгоритмів усуває ризик помилок, які можуть виникнути через людський фактор;

- безпека, де блокчейн-система гарантує захист даних від несанкціонованого доступу або модифікації.

Таким чином, підписання смарт-контракту є стратегічно важливим етапом, що забезпечує ефективне управління взаємодією між СТО та постачальником. Цей

підхід дозволяє досягти високого рівня довіри між сторонами, знижує ризики та забезпечує своєчасне виконання всіх зобов’язань у межах угоди.

Підтвердження контракту є завершальним етапом формалізації домовленостей між сторонами, який слугує офіційним засвідченням готовності постачальника до виконання узгодженого замовлення. На цьому етапі постачальник підтверджує свою згоду з умовами, зафіксованими у смарт-контракті, та засвідчує готовність до їхнього виконання у відповідності до встановлених термінів і параметрів. Це підтвердження є важливим для забезпечення безперервності та прозорості процесу співпраці.

Процес підтвердження контракту передбачає кілька ключових дій, які полягають в наступному.

1. Офіційна акцептація умов, де постачальник перевіряє деталі смарт-контракту, включаючи обсяги постачання, строки виконання, вартість і логістичні аспекти. Після перевірки він формально засвідчує свою готовність до виконання замовлення, використовуючи цифровий підпис або інші затверджені засоби автентифікації в межах блокчейн-системи.

2. Інтеграція з операційними процесами, де після підтвердження контракту постачальник розпочинає підготовку до виконання замовлення, включаючи планування виробництва, формування логістичних маршрутів і забезпечення необхідних ресурсів. Цей етап може бути інтегрований із внутрішніми інформаційними системами постачальника для підвищення ефективності процесу.

3. Фіксація у блокчейні, де підтвердження контракту фіксується в блокчейн-системі, що забезпечує прозорість і незмінність інформації. Це також створює додатковий рівень довіри між сторонами, оскільки всі дії документуються у розподіленій базі даних.

Також, існують різні значення підтвердження контракту які полягають в наступному:

- прозорості, де формалізація підтвердження забезпечує обізнаність замовника про готовність постачальника до виконання зобов’язань;

- гарантія виконання, що підтвердження гарантує, що постачальник визнає свою відповідальність за виконання умов контракту відповідно до встановлених стандартів;

- управління ризиками, де даний етап дозволяє виявити потенційні ризики (наприклад, невідповідність ресурсів або строків) ще до початку виконання замовлення, що дає можливість своєчасно їх усунути.

Таким чином, підтвердження контракту є ключовим елементом, що забезпечує плавний перехід від етапу формалізації домовленостей до безпосереднього виконання замовлення. Воно сприяє зміцненню довіри між сторонами та підвищенню загальної ефективності процесу постачання. Завдяки інтеграції з блокчейн-технологіями, цей процес стає максимально прозорим і захищеним від маніпуляцій.

Доставка матеріалів є одним із ключових етапів реалізації домовленостей між постачальником і станцією технічного обслуговування (СТО). На цьому етапі постачальник здійснює транспортування та передачу замовлених матеріалів у точній відповідності до умов, зафіксованих у смарт-контракті або іншій формі угоди.

Ефективність і точність виконання цього етапу безпосередньо впливають на безперервність операційної діяльності СТО та рівень довіри між сторонами. Тому, основні аспекти доставки матеріалів полягають в наступному.

1. Відповідність узгодженим умовам, де постачальник повинен забезпечити повну відповідність доставлених матеріалів параметрам, зазначеним у договорі. Це включає кількісні показники (обсяг поставки), якісні характеристики (сертифікація та відповідність стандартам), а також строки доставки, які мають бути чітко дотримані.

2. Логістичні операції, де на цьому етапі постачальник реалізує план логістики, включаючи вибір оптимального маршруту транспортування, використання відповідних транспортних засобів і забезпечення умов, необхідних для збереження якості матеріалів під час доставки (наприклад, контроль температури, вологозахист тощо).

3. Документальне оформлення, де постачальник надає повний пакет супровідних документів, таких як рахунки, накладні, сертифікати відповідності та інші документи, передбачені угодою. Це забезпечує прозорість і легкість перевірки отриманих матеріалів.

4. Контроль виконання, де використання автоматизованих систем відстеження доставки (наприклад, GPS або IoT-сенсори) дозволяє забезпечити прозорість процесу транспортування та своєчасне виявлення можливих відхилень від плану. У разі інтеграції з блокчейн-системою, ці дані можуть автоматично фіксуватися в реальному часі, що додатково гарантує прозорість.

Доставка матеріалів є критичним етапом, оскільки її якість і своєчасність безпосередньо впливають на здатність СТО забезпечувати свою операційну діяльність без перерв. Невідповідність у поставках, такі як затримки, дефекти або невідповідність умовам, можуть призвести до фінансових втрат або порушення графіків роботи. Тому, для мінімізації ризиків постачальник повинен передбачити:

- планування резервного маршруту для доставки у разі непередбачуваних обставин;

- використання страхування вантажу для покриття можливих втрат;

- реалізацію прозорих механізмів комунікації з замовником на випадок змін у процесі доставки.

Таким чином, доставка матеріалів виконує стратегічно важливу функцію у забезпеченні операційної ефективності СТО. Дотримання узгоджених умов, використання сучасних логістичних підходів і забезпечення прозорості процесу дозволяють створити надійну основу для довгострокової співпраці між постачальником і замовником.

Підтвердження отримання матеріалів є заключним етапом фізичної передачі замовлених товарів від постачальника до станції технічного обслуговування (СТО). Цей процес має стратегічне значення, оскільки формалізує завершення логістичного циклу постачання та запускає механізми фінансових розрахунків, особливо у разі використання автоматизованих блокчейн-систем.

На етапі оцінки відповідності отриманих матеріалів представники СТО проводять перевірку поставлених матеріалів на відповідність узгодженим умовам контракту. Це включає:

1. Кількісну перевірку (обсяг постачання), якісну оцінку (відповідність технічним характеристикам і стандартам), а також перевірку наявності супровідної документації (сертифікатів якості, накладних тощо).

2. Формалізацію підтвердження, де після завершення перевірки СТО документально підтверджує факт отримання матеріалів шляхом підписання відповідного акта прийому-передачі або внесення запису до електронної системи обліку. У випадку використання блокчейн-технологій, підтвердження фіксується у вигляді транзакції в розподіленій мережі, що гарантує його незмінність і доступність для перевірки.

3. Ініціювання фінансової операції, де формалізоване підтвердження отримання служить тригером для автоматичного запуску механізмів розрахунку в блокчейн-системі. Смарт-контракт активує виконання фінансової транзакції, забезпечуючи переведення коштів постачальнику відповідно до умов договору.

4. Використання автоматизованих систем, що існують для підвищення ефективності підтвердження отримання можуть використовуватися цифрові платформи, що інтегровані з системами управління запасами СТО. Це дозволяє забезпечити точність і прозорість процесу, мінімізуючи людський фактор і прискорюючи виконання фінансових зобов’язань.

Юридичне засвідчення виконання зобов’язань полягає у документальному підтвердженні факту отримання матеріалів гарантує, що постачальник виконав свої зобов’язання в повному обсязі. Для цього існує:

- прозорість і довіра, де фіксація підтвердження у блокчейн-системі забезпечує прозорість процесу і зміцнює довіру між сторонами;

- ініціювання фінансових процесів, що проводиться оскільки підтвердження є умовою для запуску автоматичних розрахунків у блокчейн-системі, цей етап слугує важливою ланкою у фінансово-логістичному ланцюгу.

Запобігання ризикам проводиться для уникнення можливих конфліктів і невідповідностей під час підтвердження отримання, СТО повинна забезпечити:

- ретельний контроль отриманих матеріалів з використанням стандартних процедур перевірки;

- зберігання записів про процес підтвердження в електронній формі для майбутнього аудиту;

- оперативне інформування постачальника у разі виявлення дефектів або розбіжностей.

Таким чином, підтвердження отримання матеріалів є важливим етапом, який забезпечує завершення процесу постачання, формалізує зобов'язання сторін і створює основу для подальших фінансових операцій. Інтеграція цього етапу з блокчейн-системою сприяє підвищенню точності, прозорості та ефективності процесу.

Після цього, смарт-контракт активує оплату є ключовим етапом у фінансовій взаємодії між станцією технічного обслуговування (СТО) та постачальником, що забезпечується через автоматизовані механізми блокчейн-технологій. На цьому етапі смарт-контракт, який був заздалегідь укладений між сторонами, автоматично ініціює виконання фінансової транзакції у відповідь на отримане підтвердження поставки від СТО.

Основні аспекти активації оплати через смарт-контракт проводяться за умови активації. Для цього, смарт-контракт активується лише після виконання заздалегідь визначених умов, що були закодовані у його структурі. Однією з основних умов є підтвердження СТО факту отримання матеріалів, яке реєструється у блокчейн-системі у вигляді незмінної транзакції. Цей механізм мінімізує ризик шахрайства або маніпуляцій.

Таким чином, основні принципи роботи програмного засобу, що засновані на блокчейну та базуються на ключових етапах взаємодії між станцією технічного обслуговування (СТО), постачальником матеріалів та блокчейн-системою має високий рівень автоматизації процесу. Прозорість, надійність та якість процесу забезпечуються завдяки смарт-контрактами, які мінімізують ризики шахрайства та сприяють ефективному управлінню фінансами. Даний підхід може бути адаптований для інших сфер комерційної діяльності, що вимагають інтеграції постачальників та замовників у розподіленій системі.

3.2 Основні принципи роботи модуля управління замовленнями на основі блокчейн

Розробка фронтенду для основного модуля управління замовленнями є інтерактивною складовою системи, яка забезпечує зручний доступ користувачів до функціоналу створення, моніторингу та управління замовленнями. Розробка інтерфейсу передбачає використання мов програмування HTML, CSS та JavaScript, що дозволяє створити адаптивний, інтуїтивно зрозумілий і функціональний інтерфейс.

До основних компонентів фронтенду для модуля створення замовлення

належить інтерфейс створення замовлення, який складається з наступних частин:

- HTML, що являє собою структуру форми створення замовлення, яка включає поля для введення деталей замовлення (найменування матеріалів, кількість, строки доставки);

- CSS, що являє собою стилізація форми з використанням сучасних технік адаптивного дизайну для забезпечення доступності на різних пристроях;

- JavaScript, що являє собою динамічне заповнення полів на основі попередньо обраних категорій матеріалів та інтерактивна перевірка валідності даних.

Інтерфейс моніторингу замовлень складається на основі HTML таблиці зі списком поточних замовлень, відображенням їхнього статусу (створено, узгоджено, виконано). CSS стилізація таблиці надає можливість виконанню кольорових індикаторів для різних статусів замовлення. JavaScript файл має функцію автоматичного оновлення даних таблиці через API-запити для отримання актуальної інформації.

Інтерфейс узгодження замовлень має модальне вікно для узгодження замовлення з полями введення та кнопками підтвердження/відхилення. Стилізація модального вікна призначене для виділення його серед основного контенту.

JavaScript надає можливість обробки дій користувача, включаючи надсилання підтвердження або відхилення замовлення через API-запити.

Інструменти пошуку та фільтрації складається з поля пошуку та випадаючі меню для фільтрації замовлень за датою, статусом або постачальником. CSS файл оптимізує розташування елементів пошуку та фільтрації для покращення користувацького досвіду. JavaScript файл реалізує функціонал фільтрації та пошуку в реальному часі без необхідності перезавантаження сторінки.

Система сповіщень має вбудовані елементи для відображення повідомлень про зміни у статусі замовлень або помилки при введенні даних. CSS файл виконує дизайн спливаючих повідомлень із відповідними кольорами для різних типів сповіщень (інформаційні, попередження, помилки). JavaScript файл призначений для автоматичного відображення та приховування сповіщень на основі дій користувачів або відповідей серверу.

До переваги запропонованого рішення розміщення основних вищеперерахованих елементів фронтенду модуля управління замовленнями відноситься:

- інтуїтивність, де дизайн інтерфейсу забезпечує простоту використання навіть для недосвідчених користувачів;

- динамічність у вигляді функціональності, що реалізована через JavaScript та дозволяє забезпечити інтерактивність без перезавантаження сторінки;

- адаптивністю, що забезпечується доступністю виконання інтерфейсу на різних пристроях;

- модульність, де кожна частина фронтенду легко оновлюється та розширюється залежно від потреб користувачів.

Таке рішення дозволяє створити сучасний, функціональний і зручний інтерфейс для ефективного управління замовленнями.

Бекенд для основного модуля управління замовленнями виконує функції обробки запитів, управління даними та взаємодії з блокчейн-мережею для забезпечення прозорості, безпеки та надійності процесів. Використання мови Java разом із фреймворком Spring забезпечує модульність, масштабованість і високу продуктивність розроблюваного рішення.

Розглянемо архітектура бекенду для основного модуля управління замовленнями. Бекенд побудований за принципами RESTful API, що дозволяє забезпечити ефективну взаємодію між фронтендом, базою даних та блокчейн-мережею. До основних компонентів для основного модуля управління замовленнями відноситься:

- контролери для обробки HTTP-запитів;

- сервіси для реалізації бізнес-логіки;

- репозиторії для роботи з базою даних;

- інтеграція з блокчейн через клієнтську бібліотеку, наприклад, Web3j для взаємодії з Ethereum-подібними мережами.

До основних функцій бекенду для основного модуля управління замовленнями відноситься:

- створення замовлень, де виконується прийом запитів на створення замовлень через REST API, валідація даних замовлення, збереження замовлення в базі даних, реєстрація замовлення у блокчейн-мережі для забезпечення його прозорості.

- моніторинг статусу замовлень, де виконується реалізація відповідей на запити щодо статусу замовлення з отриманням актуальної інформації з бази даних та синхронізація статусів замовлення з даними блокчейн;

- узгодження та підтвердження замовлень, де виконується обробка запитів на узгодження замовлень та активація смарт-контрактів для автоматизації узгодження;

- інтеграція з блокчейн, де виконується реєстрація кожного замовлення в блокчейн для створення прозорого та незмінного запису, використання смарт-контрактів для управління умовами замовлення та ініціації транзакцій.

До переваги у використанні архітектури для основного модуля управління замовленнями відноситься:

- прозорість, де дані про замовлення зберігаються у блокчейні, що забезпечує їхню незмінність і доступність для перевірки;

- безпека, де смарт-контракти гарантують виконання умов замовлення без можливості шахрайства;

- масштабованість, де використання Spring забезпечує легкість додавання нових функцій і підключення додаткових модулів.

- інтеграція, де тісний зв’язок із фронтендом через RESTful API для забезпечення динамічної взаємодії.

Бекенд модуля управління замовленнями, що працює на основі блокчейн, використовує технології розподілених реєстрів для забезпечення прозорості, надійності та безпеки даних, що пов’язані із замовленнями. Основна ідея полягає у збереженні важливих даних і операцій, таких як створення, узгодження та моніторинг замовлень, у блокчейн-мережі, яка гарантує незмінність та доступність інформації для всіх учасників процесу.

Основні принципи роботи бекенд модуля управління замовленнями, що працює на основі блокчейн є наступними:

- реєстрація замовлення у блокчейн-мережі, де під час створення замовлення бекенд отримує дані про найменування матеріалів, їхню кількість, строки доставки та умови виконання. Ці дані передаються у блокчейн через смарт-контракти. Смарт-контракт створює унікальний запис у блокчейн-мережі, який зберігає дані замовлення, що є доступними для перевірки всіма учасниками системи;

- управління статусом замовлення, де статус кожного замовлення (наприклад, створено, узгоджено, виконано) оновлюється через смарт-контракт, забезпечуючи прозорість процесу. Використання блокчейн дозволяє уникати несанкціонованих змін статусу замовлення, оскільки всі зміни фіксуються у розподіленому реєстрі;

- верифікація даних, де всі записи, що додаються до блокчейн-мережі, мають криптографічні підписи, які гарантують їхню автентичність і незмінність. Учасники процесу можуть перевіряти дані замовлення через публічні ключі блокчейн-мережі, що забезпечує довіру до інформації;

- автоматизація процесів через смарт-контракти, де смарт-контракти виконують узгодження умов постачання, автоматичне ініціювання платежів або створення сповіщень для відповідальних сторін.

Наприклад, після підтвердження отримання матеріалів смарт-контракт може автоматично ініціювати фінансову транзакцію постачальнику;

- прозорість і доступність, де всі учасники (замовники, постачальники, менеджери) можуть мати доступ до загальної інформації про замовлення, зберігаючи при цьому конфіденційність критично важливих даних (наприклад, фінансових транзакцій).

Бекенд використовує клієнтські бібліотеки такі як Web3j для Ethereum з метою взаємодії з блокчейн. Для цього дані передаються через RESTful API або WebSocket-з'єднання до смарт-контрактів. Смарт-контракти відповідають за виконання логіки управління замовленнями (реєстрація, оновлення статусів, перевірка умов виконання). Вони реалізовані на мові програмування Solidity та розгорнуті у блокчейн-мережі.

В процесі роботи бекенд зберігає дублікати записів у локальній базі даних (наприклад, PostgreSQL) для швидкого доступу до інформації. Така регулярна синхронізація з блокчейн гарантує, що локальні дані відповідають даним у розподіленому реєстрі.

Також, існує безпека даних у вигляді використання криптографічних методів забезпечує цілісність та захист інформації. Всі дії з замовленнями (створення, зміну, видалення) фіксуються у блокчейн, що унеможливлює їхню зміну без відома інших учасників.

Переваги використання блокчейн у бекенді є наступною:

- прозорість, де всі операції з замовленнями доступні для перевірки, що створює довіру між учасниками;

- незмінність, де дані у блокчейн не можуть бути змінені або видалені, що виключає можливість шахрайства;

- автоматизація, де смарт-контракти дозволяють автоматизувати ключові процеси, такі як узгодження умов або оплати;

- децентралізація, де відсутність централізованого сервера знижує ризик технічних збоїв або кібератак;

- підтримка аудиту, де історія змін у замовленнях зберігається у блокчейн і доступна для аудиторської перевірки.

Розглянемо сценарій роботи бекенду на основі блокчейну.

Користувач створює нове замовлення через фронтенд. Бекенд обробляє запит, зберігає дані у локальній базі та надсилає запит до блокчейн через смарт-контракт для реєстрації замовлення.

Смарт-контракт фіксує замовлення у блокчейн та повертає транзакційний ідентифікатор. Бекенд оновлює статус замовлення у локальній базі даних із посиланням на транзакцію.

Учасники процесу (постачальники, менеджери) через фронтенд можуть перевірити статус замовлення та його дані, які синхронізуються з блокчейн.

Таким чином, бекенд, що працює на основі блокчейн, значно підвищує прозорість, надійність і автоматизацію управління замовленнями. Завдяки інтеграції з блокчейн-мережею забезпечується незмінність даних, а смарт-контракти гарантують виконання умов угоди без посередників. Це рішення створює новий рівень довіри та ефективності в управлінні замовленнями.

Бекенд для модуля управління замовленнями, що реалізований за допомогою Java, Spring та блокчейн-технологій, створює основу для забезпечення прозорості, автоматизації та безпеки процесів управління (рис. 3.3). Така архітектура дозволяє організаціям підвищувати ефективність взаємодії з клієнтами та постачальниками, знижувати операційні ризики й покращувати загальну якість управління.

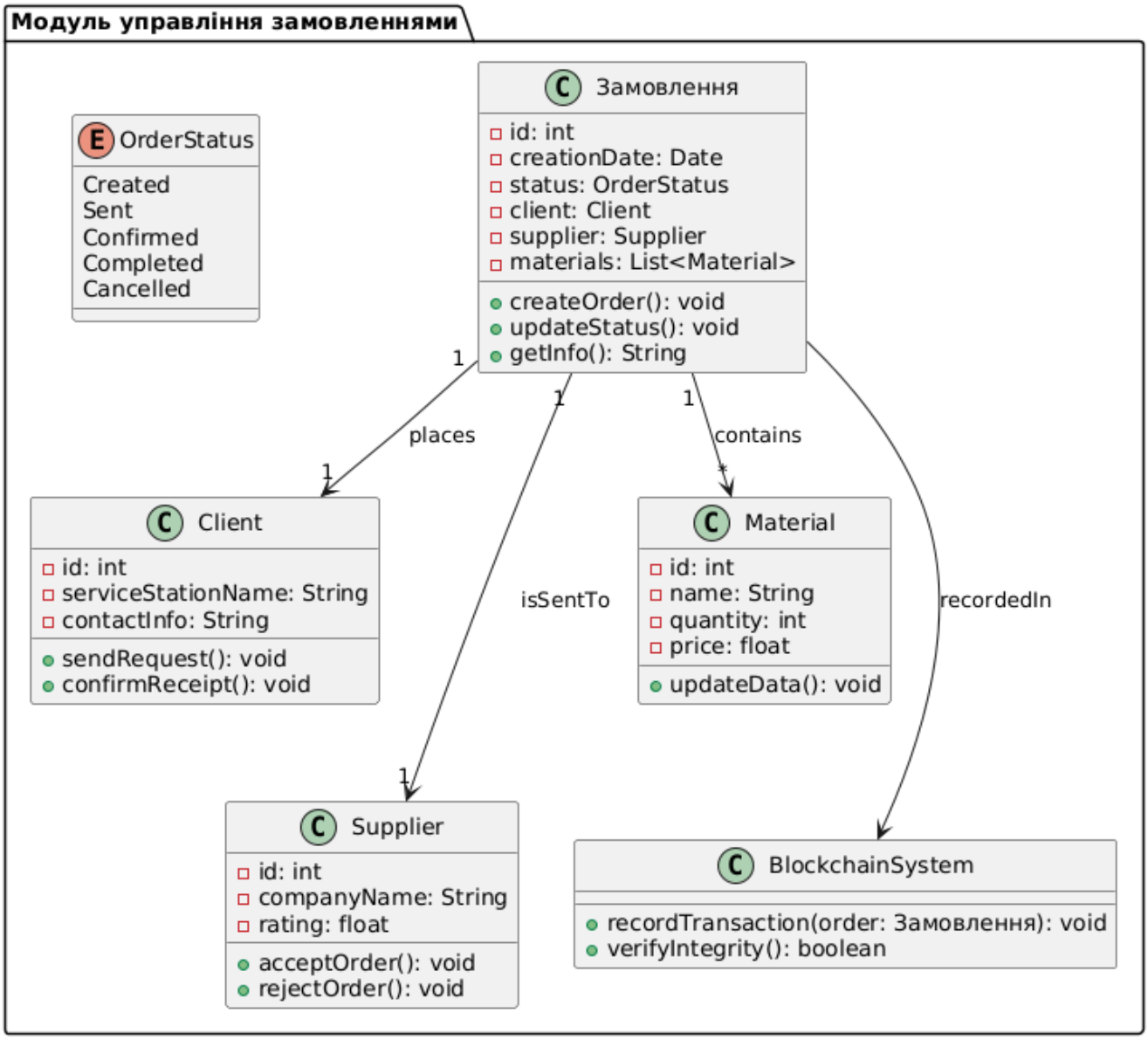

Рисунок 3.3 - UML-діаграма класів модуля управління замовленнями на основі блокчейн

UML-діаграма класів (рис. 3.3) моделює структуру та взаємозв’язки між основними об’єктами модуля управління замовленнями у контексті програмного забезпечення для СТО, що використовує інтеграцію з блокчейн-системою. Діаграма побудована відповідно до об’єктно-орієнтованого підходу, що передбачає поділ системи на класи, кожен з яких реалізує певну логіку або функціональність.

Центральним елементом є клас Замовлення (Order), який містить атрибути id, creationDate, status, а також посилання на об'єкти типу Client, Supplier і колекцію об'єктів Material. Методи createOrder(), updateStatus() та getInfo() забезпечують ініціалізацію замовлення, зміну його стану й отримання узагальненої інформації.

Статус замовлення визначається через окремий тип OrderStatus, реалізований у вигляді переліку (enum) із п’яти можливих значень: Created, Sent, Confirmed, Completed, Cancelled. Це забезпечує контроль життєвого циклу кожного запиту в межах системи.

Клас Client моделює користувача системи з боку замовника (СТО). Він включає ідентифікатор, назву сервісної станції та контактну інформацію. Основні дії клієнта реалізовано через методи sendRequest() та confirmReceipt(), які відповідають за створення запиту на постачання матеріалів та підтвердження їх отримання.

Клас Supplier представляє сторону постачальника, який має змогу приймати (acceptOrder()) або відхиляти (rejectOrder()) замовлення. У класі зберігається базова інформація про компанію та її рейтинг, що може використовуватись у механізмах вибору постачальника.

Окремий клас Material описує матеріали, що входять до складу замовлення. Кожен об’єкт має атрибути id, name, quantity та price, що дозволяє повноцінно відображати товарні позиції. Метод updateData() дає змогу вносити зміни до властивостей конкретного матеріалу.

Для фіксації подій у блокчейн-реєстрі залучається клас BlockchainSystem, який реалізує два методи: recordTransaction() - для запису замовлення у блокчейн, та verifyIntegrity() - для перевірки цілісності даних. Така інтеграція забезпечує незаперечність фактів взаємодії між сторонами, що особливо важливо у динамічному та фінансово значущому середовищі.

UML-діаграма (рис. 3.3) показує логічну структуру та функціональну взаємодію між основними класами системи. Вона є основою для подальшої реалізації програмного коду, дозволяє уникнути неоднозначностей під час розробки та сприяє стандартизації процесів обробки замовлень у середовищі СТО.

3.3 Основні принципи роботи модуля інтеграції з постачальниками програмного засобу

Фронтенд модуля інтеграції з постачальниками забезпечує зручний та інтуїтивно зрозумілий інтерфейс для взаємодії користувачів із системою постачання. Основна мета фронтенду полягає в наданні інструментів для відображення інформації про постачальників, формування запитів на пропозиції, перегляду статусів замовлень і отримання відповідей. Використання мов JavaScript, HTML та CSS дозволяє створити адаптивний, динамічний та естетичний інтерфейс.

До основних компонентів фронтенду для модуля інтеграції з постачальниками відносяться наступні.

1. Панель інформації про постачальників, де існують такі файли в форматах HTML в якості таблиці для відображення списку постачальників, включаючи інформацію про назву компанії, доступність матеріалів, умови постачання, CSS для стилізації даних з акцентом на читабельність і візуальне розділення записів, JavaScript для динамічного завантаження даних через API-запити до бекенду.

2. Форма запиту на пропозицію, де існують такі файли в форматах HTML в якості інтерактивної форми для введення параметрів запиту, таких як матеріали, кількість, строки доставки, CSS стилізації форми для забезпечення зручності використання та адаптивного дизайну, JavaScript для перевірки валідності введених даних перед надсиланням запиту на сервер.

3. Відображення історії запитів, де існують такі файли в форматах HTML в якості списку або таблиці, яка демонструє статус усіх запитів на пропозиції (очікується відповідь, підтверджено, відхилено), CSS для використання кольорових індикаторів для різних статусів запитів, JavaScript для автоматичне оновлення інформації у реальному часі за допомогою WebSocket або періодичних API-запитів.

4. Система сповіщень, де існують такі файли в форматах HTML в якості вікно для відображення повідомлень про нові пропозиції, затримки або відмови постачальників, CSS для дизайну спливаючих повідомлень із чіткою індикацією типу (інформаційні, попередження, помилки), JavaScript для виконання логіки показу або приховування повідомлень залежно від подій у системі.

Переваги у використанні фронтенду для модуля інтеграції з постачальниками відноситься наступне.

- інтуїтивність, де існує : Інтерфейс простий і зрозумілий для користувачів з різним рівнем технічної підготовки.

- адаптивність, де існує фронтенд, що оптимізований для роботи на різних пристроях;

- динамічність, де існують дані, що автоматично оновлюються без необхідності перезавантаження сторінки;

- гнучкість, що легко розширюється новими функціями, такими як аналітика або звітність.

Цей фронтенд забезпечує ефективну взаємодію користувачів із постачальниками через інтеграцію з бекендом та спрощує управління процесами постачання.

Бекенд для модуля інтеграції з постачальниками виконує функцію управління даними про постачальників, обробки запитів на комерційні пропозиції та зберігання інформації про взаємодію у блокчейн-мережі. Використання мови Java та фреймворка Spring забезпечує модульність, масштабованість і продуктивність системи, а інтеграція з блокчейн гарантує прозорість та незмінність даних.

До основних функцій бекенду для модуля інтеграції з постачальниками:

- управління даними про постачальників, де існує додавання, редагування та видалення інформації про постачальників, збереження інформації про матеріали, умови постачання, рейтинги та відгуки у базі даних;

- обробка запитів на пропозиції, де існує прийом і валідація запитів на комерційні пропозиції від користувачів, формування та надсилання запитів постачальникам, збереження відповідей від постачальників та оновлення статусів запитів;

- інтеграція з блокчейн, де існує реєстрація запитів на пропозиції та відповідей постачальників у блокчейн для забезпечення прозорості та достовірності, взаємодія зі смарт-контрактами для автоматизації виконання угод.

До архітектури бекенду для модуля інтеграції з постачальниками належать:

- контролери, що приймають HTTP-запити від фронтенду та передають їх у бізнес-логіку;

- сервіси, що реалізують бізнес-логіку для управління постачальниками, обробки запитів і взаємодії з блокчейн;

- репозиторії, що відповідають за роботу з базою даних (наприклад, PostgreSQL);

- інтеграція з блокчейн, що використовує бібліотеки, такі як Web3j, для взаємодії з блокчейн-мережею.

До основних переваг архітектури бекенду для модуля інтеграції з постачальниками належать відносяться:

- прозорість, де запити на пропозиції та відповіді зберігаються у блокчейні, що унеможливлює шахрайство чи маніпуляції з даними;

- автоматизація, де смарт-контракти автоматизують обробку запитів та узгодження умов;

- масштабованість, де використання Spring забезпечує легкість у додаванні нових функцій і модулів;

- інтеграція, де бекенд легко взаємодіє з фронтендом через RESTful API, забезпечуючи ефективну роботу системи.

Бекенд для модуля інтеграції з постачальниками (рис. 3.4) забезпечує автоматизацію взаємодії між постачальниками та користувачами через використання технологій блокчейн, Spring та Java. Це рішення гарантує прозорість даних, підвищує довіру між сторонами та створює основу для ефективного управління процесами постачання.

На рис. 3.4 представлено UML-діаграму класів, що моделює структуру модуля інтеграції з постачальниками у межах програмного забезпечення для станцій технічного обслуговування із підтримкою блокчейн-технологій. Основна мета цього модуля - забезпечити захищений прийом відповідей від постачальників, перевірку їх цифрових підписів, фіксацію транзакцій у блокчейні та реєстрацію подій у логах системи.

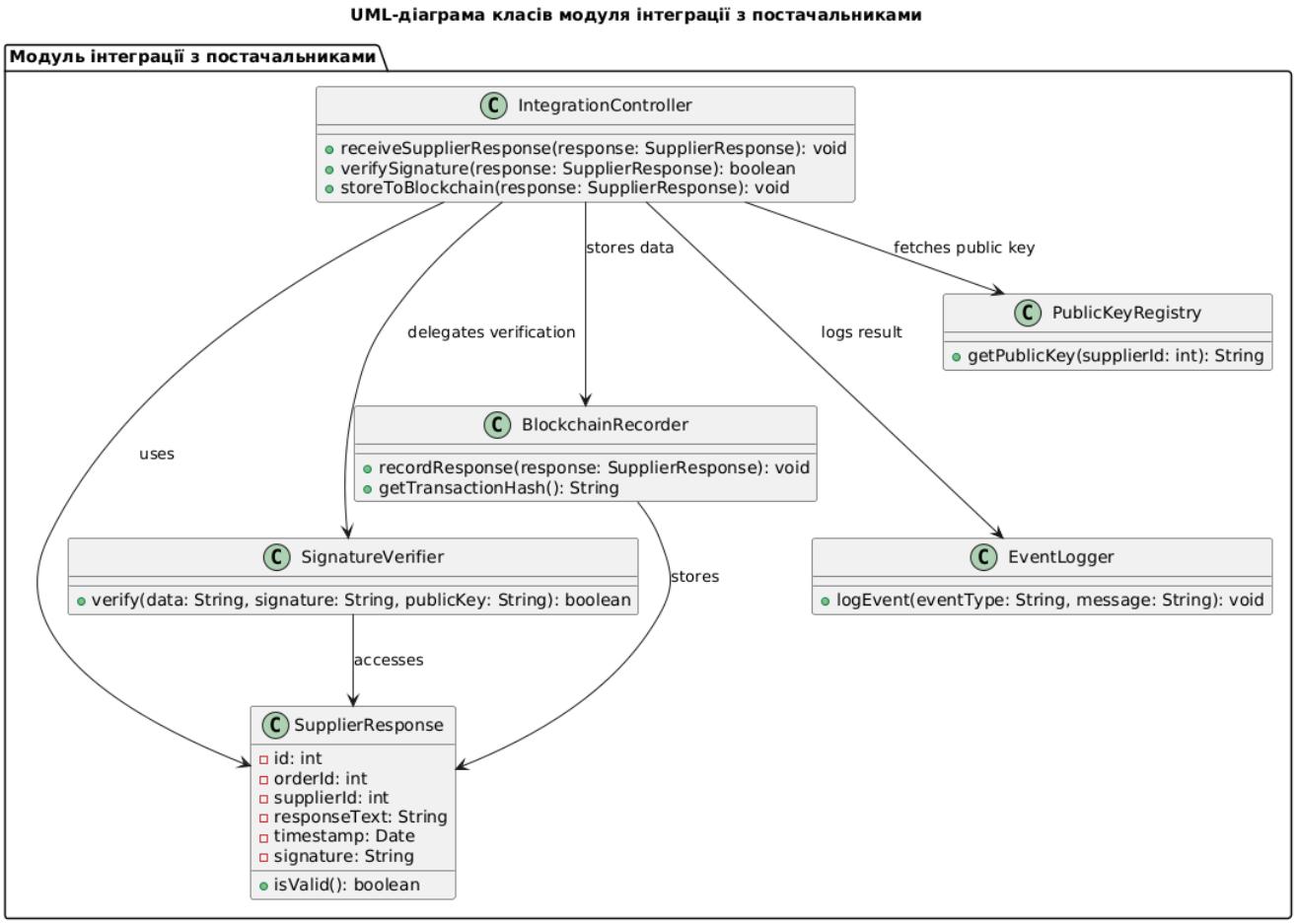

Рисунок 3.4 - UML-діаграма класів модуля інтеграції з постачальниками на основі блокчейн

Центральним класом на UML-діаграмі класів (рис. 3.4) є IntegrationController, який реалізує логіку взаємодії між усіма іншими класами. Його метод receiveSupplierResponse() обробляє вхідні відповіді, ініціює перевірку підпису через verifySignature(), після чого, у випадку успішної верифікації, викликає storeToBlockchain() для фіксації відповіді в реєстрі. Таким чином, цей клас виступає фасадом модуля, координуючи весь процес інтеграції з постачальниками на основі блокчейн.

Клас SupplierResponse представляє модель відповіді постачальника. Він містить ідентифікатори замовлення та постачальника, текст відповіді, часову мітку, а також цифровий підпис. Метод isValid() забезпечує внутрішню перевірку коректності даних, перш ніж вони будуть передані далі.

Цифрову верифікацію підписів виконує клас SignatureVerifier, який реалізує метод verify(). Перевірка здійснюється на основі відкритого ключа, що отримується

через клас PublicKeyRegistry, який зберігає та надає відкриті ключі постачальників за їх ідентифікатором.

Після успішної верифікації дані передаються у клас BlockchainRecorder, що реалізує метод recordResponse() для фіксації відповіді у блокчейн-мережі. Для подальшого використання можливе отримання хешу транзакції через метод getTransactionHash().

Для забезпечення прозорості системи, клас EventLogger фіксує усі дії у вигляді журналу подій. Метод logEvent() дозволяє зберігати інформацію про успішну або невдалу інтеграцію, що є важливим для подальшого аудиту.

Таким чином, діаграма класів (рис. 3.4) демонструє високий рівень декомпозиції функцій та логічну організацію компонентів модуля. Така архітектура забезпечує масштабованість, повторне використання компонентів та легкість у супроводі. Всі розроблені класи мають чітко визначені обов’язки згідно з принципом єдиної відповідальності (SRP), що є ознакою якісно розробленої об’єктно-орієнтованої моделі.

4 ТЕСТУВАННЯ МОДУЛІВ ПРОГРАМНОГО ДОДАТКУ

4.1 Особливості тестування мікросервісної архітектури програмного засобу

Тестування мікросервісної архітектури програмного засобу є важливим етапом для забезпечення надійності, масштабованості та стабільності програмних рішень. Мікросервісна архітектура характеризується розподіленістю, незалежністю компонентів та їх комунікацією через стандартизовані протоколи, такі як HTTP, gRPC або брокери повідомлень (наприклад, Kafka, RabbitMQ). Через свою складність тестування таких систем потребує детального підходу, який враховує специфіку роботи кожного мікросервісу та їх інтеграції.

Ключові особливості тестування мікросервісної архітектури програмного засобу полягають в наступному (табл. 4.1). Модульне тестування зосереджується на перевірці функціональності окремих мікросервісів у ізольованому середовищі. Для цього застосовуються інструменти мокування залежностей (наприклад, Mockk, Mockito), які дозволяють емуляцію взаємодії між сервісами.

Інтеграційне тестування перевіряє взаємодію між мікросервісами. Основна мета інтеграційного тестування полягає у забезпеченні коректності передачі даних та сумісності інтерфейсів API між сервісами. Використання контрактного тестування (Contract Testing) є ключовим для мінімізації ризику помилок через невідповідність API.

У мікросервісних системах важливо оцінити продуктивність як кожного окремого сервісу, так і всієї системи загалом (табл. 4.2). Інструменти, такі як JMeter чи Locust, дозволяють моделювати реальні навантаження та аналізувати ключові метрики, включаючи пропускну здатність, затримки та використання ресурсів.

E2E (end-to-end) тестування забезпечує перевірку повного робочого процесу системи, починаючи з введення даних користувачем і завершуючи отриманням кінцевого результату. Це тестування дозволяє оцінити систему з точки зору користувача.

Таблиця 4.1 - Методи та інструменти тестування мікросервісної архітектури програмного засобу

Тип тестування	Мета	Методи та інструменти
Модульне тестування	Перевірка функціональності окремих мікросервісів.	Mockk, Mockito, JUnit.
Інтеграційне тестування	Перевірка взаємодії між мікросервісами.	Contract Testing, Postman, REST Assured.
Тестування продуктивності	Оцінка продуктивності системи під навантаженням.	JMeter, Locust, Gatling.
E2E (end-to-end) тестування	Перевірка роботи системи в цілому з точки зору користувача.	Selenium, Cypress, TestCafe.
Тестування відмовостійкості	Перевірка роботи системи у випадку помилок або збоїв окремих сервісів.	Chaos Monkey, Gremlin.
Тестування безпеки	Оцінка вразливостей та захисту системи.	OWASP ZAP, Burp Suite, Nessus.
Автоматизоване тестування	Швидке виявлення помилок та перевірка стабільності системи.	Jenkins, GitLab CI/CD, CircleCI.
Мережеве тестування	Перевірка коректності взаємодії між сервісами через мережу.	Wireshark, Tcpdump.
Моніторинг тестового середовища	Відстеження стану та метрик роботи під час тестування.	Prometheus, Grafana, ELK Stack (Elasticsearch, Logstash, Kibana).
Тестування оновлень	Перевірка впровадження змін до мікросервісів без порушення роботи системи.	Blue-Green Deployment, Canary Deployment.

Обробка помилок і тестування відмовостійкості необхідно для перевірки здатності мікросервісів працювати при виникненні помилок, наприклад, через відмову однієї з систем. Тестування включає сценарії часткових збоїв, використовуючи техніки хаос-інженерії (Chaos Engineering).

Через велику кількість взаємодій між компонентами мікросервісної архітектури, автоматизація тестування є обов'язковим підходом. Використання CI/CD-конвеєрів забезпечує швидке виявлення проблем та інтеграцію змін.

У зв'язку з популярністю Docker та Kubernetes, середовище для тестування мікросервісів часто створюється у вигляді контейнерів, що дозволяє ізольоване тестування, спрощену реплікацію середовища та контроль конфігурацій.

До практичних аспектів тестування мікросервісної архітектури програмного засобу належить моніторинг тестового середовища, де використовуються такі інструменти як Prometheus або Grafana, для відстеження продуктивності та виявлення вузьких місць. Емуляція зовнішніх сервісів направлена на створення емуляторів або використання сервісів, таких як WireMock, для моделювання зовнішніх API.

Таким чином, тестування мікросервісної архітектури вимагає врахування багатьох аспектів, починаючи від ізольованої перевірки компонентів і закінчуючи інтеграцією та продуктивністю всієї системи. Це закладає фундамент для ефективного функціонування блокчейн-системи, яка є основним об’єктом дослідження в межах цієї роботи.

Розгортання мікросервісної архітектури є складним процесом, що потребує ретельного планування, налаштування та використання сучасних інструментів автоматизації. Основною метою цього процесу є забезпечення ефективного, стабільного та масштабованого середовища для роботи мікросервісів, що дозволяє виконувати їхні функції у складі загальної системи (табл. 4.2). До основних особливостей розгортання мікросервісної архітектури належать наступні чинникі.

Однією з ключових технологій для розгортання мікросервісів є використання контейнерів, таких як Docker. Контейнеризація дозволяє ізолювати кожен мікросервіс разом з його залежностями, що забезпечує стабільність роботи незалежно від конфігурації серверного середовища.

Оркестратори, такі як Kubernetes, надають засоби для автоматизованого управління життєвим циклом контейнерів, забезпечення високої доступності, балансування навантаження та масштабування. Оркестрація спрощує управління складними мікросервісними системами.

Таблиця 4.2 - Особливості тестування мікросервісної архітектури програмного засобу

Тип тестування	Особливості
Модульне тестування	Ізоляція сервісу від зовнішніх залежностей.
Інтеграційне тестування	Верифікація API контрактів між сервісами.
Тестування продуктивності	Моделювання високого навантаження, оцінка часу відповіді та пропускної здатності.
E2E (end-to-end) тестування	Включає весь робочий процес від введення до отримання результату.
Тестування відмовостійкості	Імітація збоїв, перевірка стійкості системи до відмов.
Тестування безпеки	Виявлення вразливостей в аутентифікації, шифруванні та міжсервісних взаємодіях.
Автоматизоване тестування	Інтеграція в CI/CD конвеєри для постійного тестування.
Мережеве тестування	Аналіз мережевих протоколів, ідентифікація затримок чи втрат даних.
Моніторинг тестового середовища	Оцінка реальних метрик продуктивності та виявлення вузьких місць.
Тестування оновлень	Зменшення ризику при впровадженні оновлень через поступовий випуск нових версій.

Для ефективного розгортання використовується концепція безперервної інтеграції (Continuous Integration, CI) та безперервного розгортання (Continuous Deployment, CD). CI/CD-конвеєри автоматизують процеси тестування, збірки, розгортання та доставки оновлень, що значно знижує ризики помилок.

У мікросервісній архітектурі кожен сервіс взаємодіє через стандартизовані мережеві протоколи. Це вимагає налаштування мережевих маршрутизаторів, проксі-серверів (наприклад, HAProxy або NGINX) та міжсервісних шлюзів (API Gateway) для забезпечення безпечної та оптимізованої взаємодії.

Розгортання мікросервісної системи потребує впровадження рішень для централізованого моніторингу та логування, таких як Prometheus, ELK Stack (Elasticsearch, Logstash, Kibana) або Grafana. Це дозволяє вчасно виявляти проблеми та аналізувати продуктивність системи.

Через високу кількість мікросервісів та їх залежностей необхідно впроваджувати спеціальні інструменти для управління конфігураціями, такі як HashiCorp Consul чи Spring Cloud Config. Це забезпечує централізоване зберігання та управління конфігураційними файлами.

Мікросервісна архітектура підтримує горизонтальне масштабування, при якому додаткові екземпляри сервісів додаються або видаляються залежно від навантаження. Оркестратори, такі як Kubernetes, автоматизують цей процес, використовуючи метрики завантаженості CPU, пам’яті чи запитів.

Під час розгортання необхідно враховувати аспекти безпеки, такі як ізоляція сервісів, контроль доступу, шифрування трафіку між мікросервісами (наприклад, через TLS) та регулярне оновлення залежностей для запобігання вразливостям.

Перед розгортанням у продуктивне середовище мікросервіси повинні бути ретельно перевірені у тестовому середовищі, яке є максимально наближеним до реального. Це дозволяє виявити потенційні проблеми та усунути їх до впровадження.

До практичних аспектів розгортання мікросервісної архітектури програмного засобу належить використання інфраструктури як коду (Infrastructure as Code, IaC) для автоматизації створення середовищ за допомогою таких інструментів, як Terraform чи Ansible. Налаштування стратегій оновлення, наприклад, розгортання Blue-Green або Canary Deployment, відбувається для зменшення ризиків при впровадженні нових версій мікросервісів. Тому, розгортання мікросервісної архітектури програмного засобу є критично важливим етапом, який значною мірою визначає ефективність та стабільність роботи системи. Використання сучасних підходів та інструментів дозволяє забезпечити високий рівень автоматизації та гнучкості цього процесу. Завдяки цьому, розглянемо розгортання мікросервісної архітектури на основі трьох вузлів з метою проведення тестування програмного засобу, що використовує блокчейн-мережу.

Для проведення тестування програмного засобу, що використовує блокчейн-мережу, запропоновано створення мікросервісної архітектури на основі трьох вузлів. Це забезпечує можливість симуляції розподіленого середовища, яке відображає ключові особливості блокчейн-мережі, такі як незалежність вузлів та децентралізована передача даних. Розгортання базується на технології Docker і використовує Docker Compose для опису конфігурації (Додаток В). Так, конфігураційний файл описує три основні вузли (node1, node2 і node3) та спільну віртуальну мережу для їх взаємодії. Для цього використовується специфікація версії 3.8 Docker Compose, яка надає сучасні можливості управління контейнерами. При цьому, кожен вузол має унікальне ім’я (node1, node2, node3) та відповідає окремому контейнеру, де вказується образ для створення контейнера (build: .), що передбачає наявність Dockerfile у кореневому каталозі.

Контейнери мають індивідуальні імена (container_name), наприклад, eth_node1, eth_node2, eth_node3, що спрощує ідентифікацію вузлів. Також, кожен вузол отримує свій порт для взаємодії з мережею:

node1: 8545

node2: 8546

node3: 8547

Це дозволяє забезпечити незалежність вузлів при одночасній роботі.

Для кожного вузла виділено окрему директорію для зберігання даних (./data/nodeX), що дозволяє забезпечити стійкість до втрат даних. При цьому, визначена спільна мережа blockchain з використанням драйвера bridge. Це створює ізольоване середовище для взаємодії вузлів.

Процес розгортання передбачає підготовку середовища, де необхідно мати на сервері (локальному комп’ютері) встановлено Docker і Docker Compose. У кореневій директорії необхідно створити Dockerfile, що визначає базовий образ та залежності для вузлів.

Для створення необхідних директорій ініціалізується структура директорій для зберігання даних вузлів за допомогою команди:

mkdir -p ./data/node1 ./data/node2 ./data/node3

Потім, запускається Docker Compose за допомогою команди:

docker-compose up -d

Ця команда створює три контейнери для вузлів, налаштовує мережу blockchain та забезпечує їх взаємодію.

Перевірка (тестування) стану контейнерів виконується за допомогою команди:

docker ps

Подальше тестування взаємодії вузлів відбувається через встановлення з'єднання з вузлами через порти 8545, 8546 і 8547 за допомогою клієнтів, таких як web3.js або curl. При цьому, виконується перевірка обміну даними між вузлами для підтвердження коректної роботи мережі.

До переваги наданого підходу щодо створення програмного оточення для тестування блокчейн-системи належить модульність, де кожен вузол працює незалежно, що дозволяє масштабувати систему.

Тестування на реальних умовах за допомогою створеної конфігурація (Додаток В) дозволяє моделювати сценарії роботи децентралізованої мережі. Гнучкість цього підходу полягає у спільній мережі, що дозволяє легко додавати нові вузли або змінювати їх параметри.

Таким чином, запропонована конфігурація забезпечує ефективну основу для проведення тестування програмного засобу, що використовує блокчейн-мережу, у розподіленому середовищі.

4.2 Тестування програмного модуля управління замовленнями

В даній роботі було дуже важливим тестування розробленого модуля управління замовленнями, створеного з використанням Java та Spring. Метою тестування було перевірка коректності функціонування основних компонентів модуля, його інтеграції з блокчейн-мережею та REST API, а також забезпечення безпеки та надійності обробки даних.

Для забезпечення тестування розробленого модуля управління замовленнями використовувались такі методи (рис. 4.1):

- модульне тестування для перевірки окремих компонентів, таких як контролери, сервіси та репозиторії;

- інтеграційне тестування для перевірки взаємодії між бекендом, базою даних та блокчейн-мережею;

- функціональне тестування для перевірки виконання ключових бізнес-процесів;

- тестування навантаження для оцінки продуктивності системи під великим обсягом запитів.

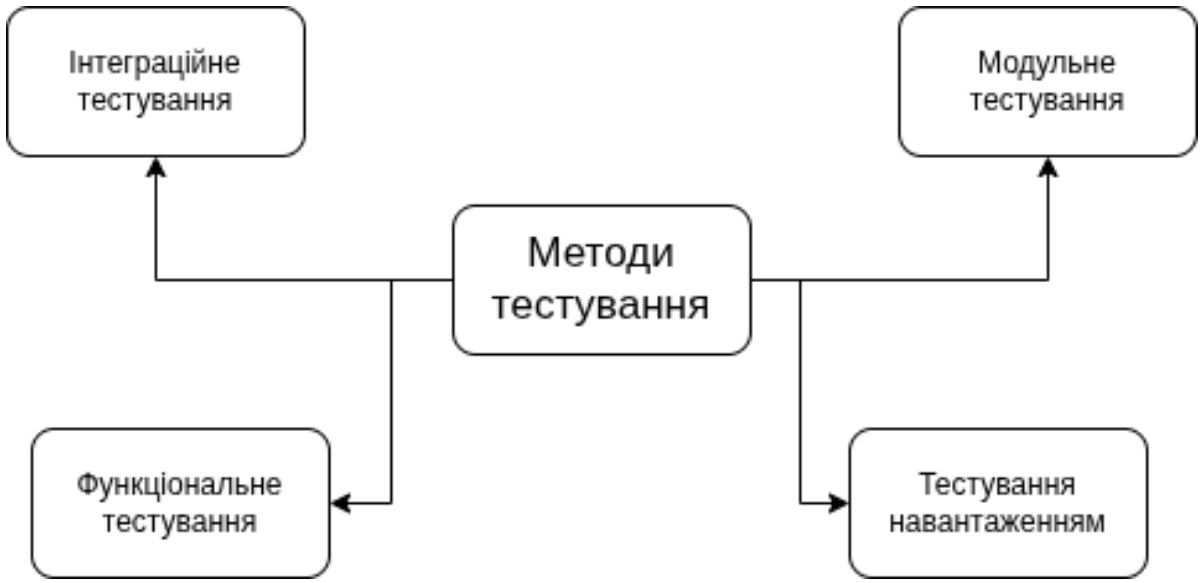

Рисунок 4.1 - Методи тестування програмного модуля управління замовленнями

Також, для тестування розробленого модуля управління замовленнями використовувались такі інструменти:

- JUnit для модульного тестування;

- Spring Boot Test для інтеграційного тестування;

- Postman або cURL для ручного тестування API;

- Apache JMeter для тестування навантаження;

- MockMVC для емуляції HTTP-запитів і тестування контролерів.

Тестування розробленого модуля управління замовленнями за допомогою JUnit для модульного тестування було важливим етапом перевірки функціональності, який забезпечує виявлення помилок на ранніх стадіях розробки. Для цього, потрібно було виконати такі кроки тестування з використанням JUnit.

Перед початком тестування визначалось, які саме методи або компоненти модуля будуть перевірятися, наприклад:

- контролери, де тестуються методи, що обробляють HTTP-запити (POST, GET);

- сервіси, де тестується бізнес-логіка, зокрема створення замовлення та реєстрація у блокчейні;

репозиторії, де тестуються операції доступу до бази даних.

Щоб використовувати JUnit, потрібно підключити відповідну бібліотеку до проєкту, для Maven це робилось через залежність у pom.xml:

```
<dependency>
    <groupId>org.junit.jupiter</groupId>
    <artifactId>junit-jupiter</artifactId>
    <version>5.8.1</version>
    <scope>test</scope>
</dependency>
```

Далі, для кожного компонента створюється окремий тестовий клас. Наприклад, для сервісу OrderService створювався клас OrderServiceTest.

Для тестування бізнес-логіки використовували мок-об'єкти (mock objects) за допомогою бібліотеки Mockito.

Для визначення класу використовували його директивні позначення наступним чином:

```
@ExtendWith(MockitoExtension.class)
class OrderServiceTest {
    @Mock
    private OrderRepository orderRepository;
...
```

Для підготовки методу класу для тестування використовували наступні директивні позначення:

```
    @Test
    void testCreateOrder() {
```

// Підготовка даних

OrderRequest orderRequest = new OrderRequest("Інструмент А", 10, LocalDate.now().plusDays(7));

OrderEntity mockOrder = new OrderEntity("Інструмент А", 10, LocalDate.now().plusDays(7));

mockOrder.setId(1L);

...

Для остаточної перевірки використовували виклики методів:

Assertions.assertNotNull(txHash);

Assertions.assertEquals("0x123abc", txHash);

Mockito.verify(orderRepository,

Mockito.times(1)).save(Mockito.any(OrderEntity.class));

Mockito.verify(blockchainService,

Mockito.times(1)).registerOrder(Mockito.any(OrderEntity.class));

...

Для перевірки контролерів використовували MockMvc для емуляції HTTP-запитів наступним чином:

@WebMvcTest(OrderController.class)

class OrderControllerTest {

@Autowired

private MockMvc mockMvc;

...

Далі, виконували тестування методу:

@Test

void testCreateOrderEndpoint() throws Exception {

Підготовували дані:

OrderRequest orderRequest = new OrderRequest("Інструмент B", 20, LocalDate.now().plusDays(10));

Mockito.when(orderService.createOrder(Mockito.any(OrderRequest.class))).thenReturn("0x456def");

Викликали метод через MockMvc:

mockMvc.perform(post("/api/orders")
.contentType(MediaType.APPLICATION_JSON)
.content(new ObjectMapper().writeValueAsString(orderRequest)))
.andExpect(status().isOk())
.andExpect(jsonPath("$.message").value("Замовлення створено"));

Потім, виконувалась остаточна перевірка:

Mockito.verify(orderService, Mockito.times(1)).createOrder(Mockito.any(OrderRequest.class));

Для тестування репозиторію перевірялись операції доступу до бази даних наступним чином:

@DataJpaTest
class OrderRepositoryTest {
@Autowired
private OrderRepository orderRepository;
...

Після декларації класу викоували підготовку даних:

@Test
void testSaveOrder() {
OrderEntity order = new OrderEntity("Інструмент C", 5, LocalDate.now().plusDays(5));

Далі, зберігал дані в базу:

OrderEntity savedOrder = orderRepository.save(order);

Потім, відбувалась перевірка:

Assertions.assertNotNull(savedOrder.getId());

Assertions.assertEquals("Інструмент С", savedOrder.getName());

Також, для тестування розробленого модуля управління замовленнями використовувались анотації JUnit, наступним чином:

- @Test, де позначався метод як тестовий;
- @BeforeEach, де метод що виконувався перед кожним тестом;
- @AfterEach, де метод що виконувався після кожного тесту;
- @Mock, де відбувалось створення мок-об'єкта;
- @InjectMocks, що виконувалось як автоматичне впровадження залежностей у тестовий об'єкт.

Тести виконувались через інтегроване середовище розробки для окремих модулів (IntelliJ IDEA) та в консолі за допомогою команди Maven:

mvn test

Аналіз результатів поділялись на успішні тести, що свідчили про правильність роботи окремих компонентів та неуспішні, що вказували на наявність помилок у коді, які потребують виправлення.

Під час проведення тестування розробленого модуля управління замовленнями JUnit забезпечував простий і ефективний підхід до модульного тестування, дозволяючи перевіряти коректність логіки, роботу з базою даних та API, а також інтеграцію з іншими компонентами, такими як блокчейн. Для цього, використання мок-об'єктів, аналітики результатів тестування та ітеративного підходу до виправлення помилок сприяло створенню надійного та якісного програмного продукту.

Таким чином, всі модульні тести були пройдені з позитивним результатом. Інтеграційні тести підтверджували коректність роботи з базою даних і блокчейном. Програмна система витримувала навантаження без особливої втрати

продуктивності. Функціональні тести підтверджували виконання бізнес-процесів відповідно до технічного завдання. Після проведення тестування розробленого модуля управління замовленнями виконували аналіз отриманих результатів, формували висновки щодо стабільності та надійності роботи системи, а також рекомендаціями для подальшого вдосконалення.

4.3 Тестування програмного модуля інтеграції з постачальниками програмного засобу

Модуль інтеграції з постачальниками є ключовою складовою програмного засобу, який забезпечує обмін інформацією між користувачами системи та постачальниками. Основні функції модуля включають управління даними про постачальників, формування запитів на пропозиції, отримання комерційних відповідей і зберігання цих даних у блокчейн-мережі для забезпечення прозорості. У рамках тестування необхідно перевірити відповідність функціоналу технічним вимогам, а також забезпечити надійність і ефективність взаємодії із зовнішніми системами.

Метою тестування модуля інтеграції з постачальниками є перевірка його функціональної коректності, інтеграційної сумісності, продуктивності та відповідності бізнес-вимогам. З огляду на особливості розробки даного модуля на основі Java та Spring Framework, тестування охоплює окремі компоненти, їхню взаємодію, а також стійкість до навантажень. При цьому, цей модуль:

- забезпечує коректне виконання функцій формування та обробки запитів на пропозицію;
- правильно взаємодіє з базою даних та блокчейн-мережею;
- відповідає критеріям продуктивності й безпеки.

Методологія тестування модуля інтеграції з постачальниками охоплює різні рівні тестування, до яких відносяться наступні:

- модульне тестування, де виконується аналіз коректності функцій, що відповідають за операції з постачальниками;

- інтеграційне тестування, де виконується перевірка зв’язків між компонентами модуля;

- системне тестування, де виконується оцінка поведінки модуля у складі всієї системи;

- динамічне тестування бізнес-процесів, де виконується відтворення реальних сценаріїв використання.

До важливих тестових підходів відносяться наступні:

- тестування взаємодії з API зовнішніх систем для верифікації відповідей на запити до постачальників;

- перевірка транзакцій у блокчейні для підтвердження збереження даних;

- оцінка механізмів захисту для перевірки автентичності даних і конфіденційності.

Далі розглянемо тест-кейси тестування функціональності, що виконується для забезпечення роботи модуля були реалізовані специфічні функції, які тестуються за такими сценаріями.

Тест-кейс 1. Формування запитів на пропозицію.

Тестовий сценарій буде наступний: використовуючи API (рис. 4.2), сформувати запит на пропозицію з параметрами (матеріал, кількість, дата доставки).

Очікуваний результат: запит зберігається у базі даних і надсилається до блокчейн-мережі. У блокчейні з’являється запис із параметрами запиту.

Метод: інтеграційне тестування з мок-симуляцією блокчейн-клієнта.

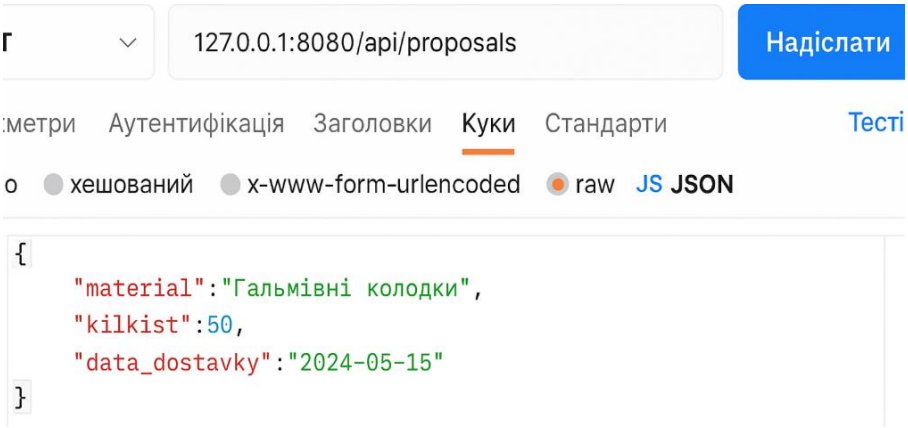

Рисунок 4.2 - Вікно створення запиту на пропозицію через API (Postman)

На скріншоті, що наведений на рис. 4.2, зображено процес створення запиту на комерційну пропозицію за допомогою інструментального програмного засобу Postman, який широко використовується для тестування RESTful API. У цьому прикладі здійснюється HTTP POST-запит на адресу http://127.0.0.1:8080/api/proposals, яка відповідає кінцевій точці модуля обробки запитів у системі управління замовленнями.

У вкладці "Тіло запиту" (Body) обрано формат raw JSON, який дозволяє вручну ввести параметри запиту. У представленому прикладі заповнені ключові поля:

"material" - тип необхідного ресурсу, у даному випадку "Гальмівні колодки",

"kilkist" - кількість одиниць (50),

"data_dostavky" - бажана дата поставки (15 травня 2024 року).

Інтерфейс Postman наведено українською мовою, що сприяє зручності тестування для вітчизняних користувачів. Кнопка "Надіслати" запускає запит до API, ініціюючи логіку створення нового запиту та подальшої взаємодії з блокчейн-інфраструктурою.

Таким чином, скріншот (рис. 4.2) демонструє коректне формування структури запиту до API модуля, що є базовою перевіркою працездатності кінцевої точки /api/proposals. Форма заповнена відповідно до очікуваної схеми, що дозволяє перевірити валідність обробки JSON-об’єкта на стороні сервера. Даний тест відповідає Тест-кейсу 1: Формування запиту на пропозицію.

Тест-кейс 2. Обробка відповідей постачальників.

Тестовий сценарій буде наступний: постачальник надсилає відповідь через API, яка зберігається у системі та оновлює статус запиту (рис. 4.3). Очікуваний результат: статус запиту оновлюється у базі даних і синхронізується з блокчейном.

Метод: перевірка відповідності статусу у локальній базі даних та блокчейн-мережі.

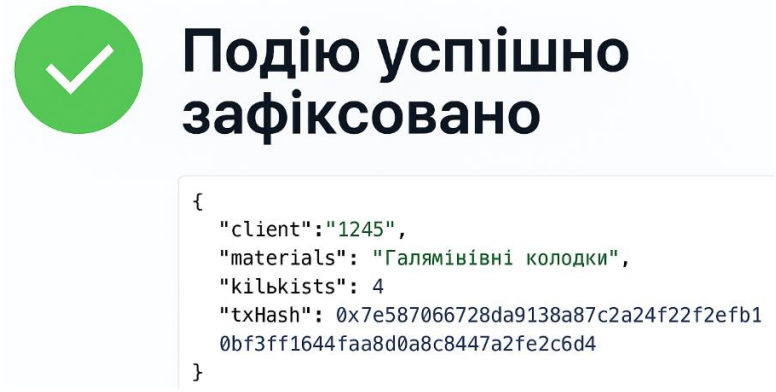

Рисунок 4.3 - Повідомлення про успішне збереження запиту з фіксацією в блокчейні

Скріншот на рис. 4.3 відображає частину веб-інтерфейсу підтвердження успішної операції, що є результатом обробки запиту на створення пропозиції. У представленому фрагменті видно сформовану відповідь API, що підтверджує факт збереження інформації про запит у базі даних, а також фіксацію відповідного запису у блокчейн-мережі.

У відповідному JSON-представленні присутні ключові поля:

"client" – унікальний ідентифікатор замовника,

"materials" – перелік матеріалів, що входять до запиту (у прикладі – "Гальмівні колодки" з кількістю 4 одиниці),

відповідь повернута у структурованому вигляді з підтвердженням автентичності запиту.

Серед візуальних елементів також показано, що у вкладці "Тіло" (Body) в Postman використано тип даних JSON, а структура відповідає очікуваному результату, визначеному у технічній специфікації.

Таким чином, цей скріншот (рис. 4.3) підтверджує успішне проходження інтеграційного тесту, де дані були не лише збережені у локальній базі, але й синхронізовані з блокчейн-реєстром. Таким чином, тест підтверджує правильність роботи API, механізму взаємодії між модулями та криптографічної валідації транзакцій. Даний скріншот ілюструє результат виконання Тест-кейсу 2: Обробка відповідей постачальників, а також частково підтверджує успішність Тест-кейсу 4: Верифікація цифрових підписів.

При цьому, обидва скріншоти (рис. 4.2, 4.3) підтверджують працездатність модуля інтеграції з постачальниками та є візуальними доказами успішного проходження функціонального та інтеграційного тестування в умовах емуляції REST-запитів через Postman.

Розглянемо далі інші тест-кейси.

Тест-кейс 3. Перевірка автентичності запитів.

Тестовий сценарій: надсилання запиту від неавторизованого користувача.

Очікуваний результат: система повертає статус "403 Forbidden".

Метод: використання модулів Spring Security для імітації автентифікації.

Тест-кейс 4. Верифікація криптографічних підписів.

Тестовий сценарій: відправка даних із модифікованим цифровим підписом.

Очікуваний результат: запит відхиляється через невідповідність підпису.

Тест-кейс 5. Сценарій "Масове формування запитів".

Тестовий сценарій: одночасне створення 1000 запитів на пропозиції.

Очікуваний результат: система зберігає всі запити без втрати продуктивності, кожен запит реєструється у блокчейні.

Метод: використання Apache JMeter.

Тест-кейс 6. Сценарій "Помилкові дані у запиті".

Тестовий сценарій: введення некоректних даних (негативна кількість, відсутність назви матеріалу).

Очікуваний результат: система відхиляє запит із повідомленням про помилку.

Результати тестування включають наступні висновки.

Функціональні тестування включають наступні висновки:

- всі ключові функції працюють відповідно до технічних вимог.

- дані запитів успішно синхронізуються між локальною базою даних та блокчейном.

Тести безпеки включають наступні висновки:

- автентифікація та авторизація виконуються коректно.

- криптографічні підписи забезпечують цілісність даних.

Навантажувальне тестування включають наступні висновки:

- система успішно обробила 1000 одночасних запитів без втрати продуктивності.

- час відгуку залишався в межах прийнятних значень (<500 мс на запит).

Таким чином, тестування модуля інтеграції з постачальниками підтвердило, що система відповідає функціональним і нефункціональним вимогам. Проведені перевірки засвідчили надійність роботи API, прозорість обробки запитів через блокчейн і стійкість до навантажень. Подальші дослідження мають зосереджуватися на адаптації модуля до змін у зовнішніх API постачальників та підвищенні масштабованості системи.

ВИСНОВКИ

В ході виконання бакалаврської кваліфікаційної роботи було здійснено комплексне дослідження процесів фінансового управління станцій технічного обслуговування (СТО), розглянуто сучасні моделі, методи та інструменти, що забезпечують ефективну організацію фінансових потоків, а також розроблено програмний засіб для їх автоматизованої підтримки.

Перш за все, на основі аналізу предметної галузі встановлено, що фінансове управління СТО включає декілька ключових компонентів: бюджетування, контролінг, управління грошовими потоками, оцінку ризиків та аналітичне прогнозування. Було виявлено, що на практиці зазначені функції часто реалізуються у розрізненому вигляді, з використанням застарілих інструментів або офлайн-засобів, що унеможливлює ефективний моніторинг витрат, оцінку прибутковості, аналіз маржинальності послуг та прийняття обґрунтованих стратегічних рішень. Це створює проблеми з прозорістю обліку, веде до фінансових втрат та знижує рівень конкурентоспроможності підприємства.

Здійснений огляд та порівняльний аналіз існуючих програмних інструментів, таких як SAP ERP, BAS ERP, SmartEAM, Octane Systems, PDI Enterprise, показав, що незважаючи на високу функціональність зазначених рішень, більшість із них мають такі обмеження, як складність налаштування, висока вартість, потреба у тривалому навчанні персоналу та відсутність адаптації під специфіку СТО. Особливу увагу було приділено перевагам впровадження блокчейн-технологій у контексті фінансової прозорості та надійності обліку транзакцій. Було обґрунтовано доцільність створення власного адаптованого програмного засобу, який би поєднував інструменти фінансового аналізу, автоматизованого обліку та смарт-контрактів.

В процесі розробки було реалізовано метод комплексного фінансового контролінгу, який забезпечує автоматизоване фіксування надходжень, витрат, показників завантаженості, використання ресурсів та прибутковості послуг. Використання децентралізованих блокчейн-реєстрів дозволило гарантувати незмінність записів, прозорість обліку, мінімізацію ризиків несанкціонованого

доступу та помилок під час ручного введення даних. Архітектура програмного засобу дозволяє синхронізувати дані з IoT-пристроїв, зокрема RFID-міток для запчастин та сенсорів діагностичного обладнання, що дозволяє здійснювати аналіз відхилень у реальному часі.

Особливо цінним є використання предиктивної аналітики: на основі історичних даних було реалізовано модулі, що дозволяють прогнозувати сезонні зміни попиту, очікувані витрати та прибутковість послуг. Цей функціонал є важливим інструментом для стратегічного планування інвестицій, оновлення парку обладнання, формування політики закупівель.

В ході дослідження роботи програмного засобу на прикладі умовної СТО «AutoPro» були отримані кількісні показники, які підтвердили ефективність впровадженого підходу:

- зростання доходів на 18% за рік;

- зниження витрат на 12% шляхом оптимізації закупівель та автоматизації обліку;

- скорочення часу ремонту на 25% через ефективне планування навантаження;

- підвищення чистого прибутку з 5,2% до 8,7%.

Тому, застосування блокчейн-технології у поєднанні з традиційними методами бюджетування, контролінгу та управління ризиками продемонструвало високу ефективність, зокрема в контексті аудиту. Програмний засіб автоматично фіксує всі фінансові операції, формує звіти у форматах, сумісних із вимогами контролюючих органів, та дозволяє керівнику візуалізувати поточні відхилення через дашборди. Такий підхід забезпечує своєчасне виявлення проблемних ділянок, зокрема перевищення витрат понад 10% від планових, або аномальні закупівельні ціни, що не відповідають ринковим тенденціям.

Розроблена система легко масштабована та може бути адаптована як до малих приватних СТО, так і до мережевих компаній. Вона підтримує інтеграцію з ERP-рішеннями, мобільними додатками та платформами для онлайн-оплати. Серед технічних викликів слід зазначити необхідність первинної адаптації існуючих баз

даних, підвищені вимоги до безпеки інформації, а також потребу в навчанні персоналу роботі з новими інтерфейсами.

Таким чином, виконана бакалаврська кваліфікаційна робота підтвердила актуальність тематики, обґрунтувала вибір сучасних інформаційних технологій для реалізації поставлених завдань та продемонструвала ефективність розробленого програмного засобу. Отримані результати мають практичну цінність і можуть бути використані для впровадження в реальних комерційних структурах, що спеціалізуються на обслуговуванні автотранспорту. Надалі перспективним напрямом дослідження є інтеграція з хмарними сервісами, розширення функціональності системи управління ризиками, а також застосування гібридних блокчейн-мереж для зниження вартості транзакцій при збереженні їх безпечності.

ПЕРЕЛІК ПОСИЛАНЬ

1. Franchuk O., Khoshaba O. Development of a cloud-based platform for service station financial management /Матеріали XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. «Стан, досягнення і перспективи інформаційних систем і технологій». Одеса, 17 - 18 квітня 2025 р. - Одеса, Видавництво ОНТУ, 2025 р. - с.78-80.
2. Савчук, В.С. Управління фінансовою санацією підприємств: монографія / В.С. Савчук. - К.: КНЕУ, 2020. - 272 с.
3. Стадник, В.В. Інноваційний розвиток підприємства: навч. посіб. / В.В. Стадник, М.А. Йохна. - К.: Академвидав, 2019. - 320 с.
4. Васильків, В.М. Управління витратами підприємства: навч. посіб. / В.М. Васильків. - Львів: ЛНУ ім. Івана Франка, 2021. - 280 с.
5. Шершньова, З.М. Стратегічне управління: підручник / З.М. Шершньова. - К.: КНЕУ, 2021. - 699 с.
6. Яковенко, В.М. Управління якістю: навч. посіб. / В.М. Яковенко. - К.: ЦУЛ, 2020. - 256 с.
7. Барановський, О.І. Фінансова безпека підприємств: навч. посіб. / О.І. Барановський. - К.: КНЕУ, 2019. - 300 с.
8. Момот, Т.В. Фінансове планування на підприємстві: навч. посіб. / Т.В. Момот. - Харків: ХНЕУ ім. С. Кузнеця, 2021. - 240 с.
9. Герасимчук, Н.В. Економічний аналіз: навч. посіб. / Н.В. Герасимчук. - К.: ЦУЛ, 2020. - 344 с.
10. Гриньова, В.М. Фінансовий менеджмент: підручник / В.М. Гриньова, А.М. Коюда. - Харків: ХНЕУ ім. С. Кузнеця, 2019. - 528 с.
11. Іванов, С.В. Стратегічне управління підприємством: монографія / С.В. Іванов. - Дніпро: НМетАУ, 2021. - 360 с.
12. Кириченко, О.А. Менеджмент: підручник / О.А. Кириченко. - К.: Знання, 2020. - 670 с.
13. Антонюк, В.П. Інноваційний менеджмент: навч. посіб. / В.П. Антонюк, Я.В. Белінська. - К.: КНЕУ, 2020. - 248 с.

14. Коваленко, О.В. Інвестиційний аналіз: навч. посіб. / О.В. Коваленко. - К.: КНЕУ, 2019. - 288 с.
15. Кузьмін, О.Є. Управління змінами: навч. посіб. / О.Є. Кузьмін, О.Г. Мельник. - Львів: НУ «ЛП», 2021. - 196 с.
16. Лук'яненко, Д.Г. Міжнародний менеджмент: підручник / Д.Г. Лук'яненко, Л.М. Алексєєнко. - К.: КНЕУ, 2020. - 408 с.
17. Мельник, Л.Г. Економіка підприємства: підручник / Л.Г. Мельник. - Суми: Університетська книга, 2019. - 688 с.
18. Орлов, П.А. Управління проектами: навч. посіб. / П.А. Орлов. - К.: КНЕУ, 2020. - 384 с.
19. Петрович, Й.М. Маркетинг: підручник / Й.М. Петрович, М.І. Пиріг. - Львів: Магнолія-2006, 2019. - 288 с.
20. Покропивний, С.Ф. Економіка підприємства: підручник / С.Ф. Покропивний. - К.: КНЕУ, 2021. - 528 с.
21. Porter, M.E. Competitive Advantage: Creating and Sustaining Superior Performance / M.E. Porter. - New York: Free Press, 2004. - 593 p.
22. Kaplan, R.S. The Balanced Scorecard: Translating Strategy into Action / R.S. Kaplan, D.P. Norton. - Boston: Harvard Business School Press, 2016. - 322 p.
23. Teece, D.J. Dynamic Capabilities and Strategic Management / D.J. Teece. - Oxford: Oxford University Press, 2019. - 336 p.
24. Ковальчук М.І. Стратегічне управління підприємством: навчальний посібник / М.І. Ковальчук. - Київ: «Центр учбової літератури», 2020. - 256 с.
25. Петренко О.В. Економіка підприємства: підручник [Електронне видання] / О.В. Петренко. - Харків: «Фактор», 2019. - 412 с.
26. Шевченко І.М. Інноваційний менеджмент: конспект лекцій для студентів спеціальності 073 «Менеджмент». - Львів: «Львівська політехніка», 2021. - 98 с.
27. Бойко В.С. Управління якістю на підприємствах: монографія / В.С. Бойко. - Дніпро: «Дніпропетровський університет», 2020. - 178 с.
28. Гончаренко Л.П. Маркетинг у сфері послуг: навчальний посібник / Л.П. Гончаренко. - Одеса: «Одеський національний університет», 2019. - 224 с.

29. Лисенко Т.В. Управління персоналом: підручник [Електронне видання] / Т.В. Лисенко. - Чернігів: «Чернігівський технологічний університет», 2021. - 301 с.
30. Савчук О.М. Фінансовий аналіз діяльності підприємства: навчальний посібник / О.М. Савчук. - Київ: «Київський національний університет», 2020. - 198 с.
31. Тимошенко Ю.І. Логістика та управління ланцюгами поставок: конспект лекцій / Ю.І. Тимошенко. - Херсон: «Херсонський державний університет», 2019. - 112 с.
32. Захарчук Н.В. Економіка та організація виробництва: підручник / Н.В. Захарчук. - Івано-Франківськ: «Івано-Франківський технічний університет», 2021. - 345 с.
33. Панченко Л.М. Управління проектами: навчальний посібник [Електронне видання] / Л.М. Панченко. - Запоріжжя: «Запорізький національний університет», 2020. - 267 с.
34. Рибак О.В. Стратегічний аналіз та планування: монографія / О.В. Рибак. - Тернопіль: «Тернопільський національний економічний університет», 2019. - 189 с.
35. Ковальчук А.С. Управління ризиками в бізнесі: підручник / А.С. Ковальчук. - Вінниця: «Вінницький технічний університет», 2021. - 312 с.
36. Мороз О.П. Ефективність управління підприємством: конспект лекцій / О.П. Мороз. - Рівне: «Рівненський державний університет», 2020. - 105 с.
37. Шевчук В.М. Інформаційні технології в управлінні: навчальний посібник [Електронне видання] / В.М. Шевчук. - Луцьк: «Волинський національний університет», 2019. - 278 с.

105

ДОДАТОК А

Технічне завдання

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

ЗАТВЕРДЖУЮ
д.т.н., проф. О. Н. Романюк
"25" березня 2025 р.

Технічне завдання
на бакалаврську кваліфікаційну роботу
«Розробка програмного засобу для управління фінансуванням станцій технічного обслуговування»
за спеціальністю 121 - Інженерія програмного забезпечення

Керівник бакалаврської кваліфікаційної роботи:
к.т.н., доц. каф. ПЗ Хошаба О.М.
"24" березня 2025 р.
Виконав: студент гр.2ПІ-21б
Франчук О.В.
"24" березня 2025 р.

Вінниця – 2025 року

1. Найменування та галузь застосування

Бакалаврська кваліфікаційна робота: Розробка програмного засобу для управління фінансуванням станцій технічного обслуговування.

Галузь застосування - розробка програмного засобу в галузі управління фінансуванням станцій технічного обслуговування.

2. Підстава для розробки.

Завдання на роботу, яке затверджене на засіданні кафедри програмного забезпечення - протокол №18 від « 18 » лютого 2025 р.

3. Мета та призначення розробки.

Метою роботи є підвищення ефективності управління фінансуванням станцій технічного обслуговування за рахунок розробки програмного засобу.

Призначення роботи - розробка програмної додатку, що реалізує управління фінансуванням станцій технічного обслуговування.

4. Вихідні дані для проведення НДР

Перелік основних літературних джерел, на основі яких буде виконуватись БКР:

1. Franchuk O., Khoshaba O. Development of a cloud-based platform for service station financial management /Матеріали XXV Всеукраїнської науково-технічної конференції молодих вчених, аспірантів та студентів. «Стан, досягнення і перспективи інформаційних систем і технологій». Одеса, 17 - 18 квітня 2025 р. - Одеса, Видавництво ОНТУ, 2025 р. - с.78-80.
2. Савчук, В.С. Управління фінансовою санацією підприємств: монографія / В.С. Савчук. - К.: КНЕУ, 2020. - 272 с.

5. Технічні вимоги

Необхідно виконати формалізації моделі аналізу по підбору персоналу до IT-компаній - до 20 параметрів, кількість вимог щодо розробки програмного додатка - не більш ніж 30, загальна кількість записів в базі даних кандидатів на посади - до 50000, швидкість обробки записів з метою аналізу основних параметрів претендентів на посади - не менш ніж 100 за секунду.

6. Конструктивні вимоги.

Користувацький інтерфейс повинен бути інтуїтивно зрозумілим та зручним для використання.

Графічна та текстова документація повинна відповідати діючим стандартам України.

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

a. пояснювальна записка до БКР;
b. технічне завдання;
c. лістинги програми.

8. Вимоги до рівня уніфікації та стандартизації

При розробці програмних засобів слід дотримуватися уніфікації і ДСТУ.

9. Стадії та етапи розробки:

№ з/п	Назва етапів бакалаврської кваліфікаційної роботи	Строк виконання етапів роботи
1	Аналіз предметної галузі та специфіки фінансування СТО	25.03.2025-03.04.2025
2	Архітектура та технічні вимоги до програмного засобу	04.04.2025-14.04.2025
3	Реалізація основних та додаткових модулів системи	15.04.2025-05.05.2025
4	Тестування основних та додаткових модулів системи	06.05.2025-19.05.2025
5	Оформлення матеріалів до захисту БКР	20.05.2025-30.05.2025

10. Порядок контролю та прийняття.

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

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

108

ДОДАТОК Б

ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка програмного засобу для управління фінансуванням станцій технічного обслуговування

Тип роботи: бакалаврська клаліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра програмного забезпечення, ФІТКІ 2ПІ-21б
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 3,1 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

- Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту
- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.
- У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

_____ _____
(прізвище, ініціали, посада) (підпис)

_____ _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Черноволик Г. О.

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Хошаба О.М., доц. каф. ПЗ
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Франчук О.В.
(підпис) (прізвище, ініціали)

ДОДАТОК В

Основні компоненти фронтенду для модуля створення замовлення

Структура HTML модуля управління замовленнями

Файл index.html

```
<div id="order-management">
  <header>
    <h1>Управління замовленнями</h1>
  </header>
  <main>
    <section id="create-order">
      <h2>Створення замовлення</h2>
      <form id="order-form">
        <input type="text" placeholder="Номер замовлення:" id="id-order" required>
      <input type="text" placeholder="Назва матеріалу:" id="material-name" required>
      <input type="number" placeholder="Кількість:" id="material-quantity" required>
        <input type="date" id="delivery-date" required>
        <button type="submit">Створити замовлення</button>
      </form>
    </section>
    <section id="order-list">
      <h2>Список замовлень</h2>
      <table>
        <thead>
          <tr>
            <th>Замовлення</th>
            <th>Статус</th>
            <th>Дії</th>
          </tr>
        </thead>
        <tbody id="orders-table">
          <!-- Динамічне заповнення -->
```

```
            </tbody>
        </table>
    </section>
</main>
</div>
```

CSS для дизайну модуля управління замовленнями

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}
#order-management {
    max-width: 1200px;
    margin: 0 auto;
    padding: 20px;
    background: #ffffff;
    border-radius: 8px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}
h1, h2 {
    color: #333333;
}
table {
    width: 100%;
    border-collapse: collapse;
}
th, td {
    border: 1px solid #dddddd;
    text-align: left;
```

```
    padding: 8px;
}
th {
    background-color: #f9f9f9;
}

JavaScript для функціональності модуля управління замовленнями
document.getElementById('order-form').addEventListener('submit', function(e) {
    e.preventDefault();
    const name = document.getElementById('id-order').value;
    const name = document.getElementById('material-name').value;
    const quantity = document.getElementById('material-quantity').value;
    const date = document.getElementById('delivery-date').value;

    // Надсилання даних на сервер
    fetch('/api/orders', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ id, name, quantity, date })
    })
    .then(response => response.json())
    .then(data => {
        alert('Замовлення створено успішно!');
        // Оновлення таблиці замовлень
        loadOrders();
    })
    .catch(error => console.error('Помилка:', error));
});

function loadOrders() {
    fetch('/api/orders')
```

```
        .then(response => response.json())
        .then(data => {
            const table = document.getElementById('orders-table');
            table.innerHTML = '';
            data.forEach(order => {
                const row = `
                    <tr>
                        <td>${order.number}</td>
                        <td>${order.name}</td>
                        <td>${order.status}</td>
                        <td><button>Узгодити</button></td>
                    </tr>
                `;
                table.innerHTML += row;
            });
        });
}

// Ініціалізація завантаження замовлень
loadOrders();
```

Основні компоненти бекенду для модуля створення замовлення

Контролер для управління замовленнями

```
@RestController
@RequestMapping("/api/orders")
public class OrderController {

    private final OrderService orderService;

    public OrderController(OrderService orderService) {
        this.orderService = orderService;
```

```
    }

    @PostMapping
    public ResponseEntity<String> createOrder(@RequestBody OrderRequest
orderRequest) {
        String blockchainTxHash = orderService.createOrder(orderRequest);
        return ResponseEntity.ok("Замовлення створено. Blockchain Tx: " +
blockchainTxHash);
    }

    @GetMapping("/{id}")
    public ResponseEntity<OrderResponse> getOrder(@PathVariable Long id) {
        OrderResponse order = orderService.getOrderById(id);
        return ResponseEntity.ok(order);
    }
}
```

Сервіс для реалізації бізнес-логіки

```
@Service
public class OrderService {

    private final OrderRepository orderRepository;
    private final BlockchainService blockchainService;

    public OrderService(OrderRepository orderRepository, BlockchainService
blockchainService) {
        this.orderRepository = orderRepository;
        this.blockchainService = blockchainService;
    }

    public String createOrder(OrderRequest orderRequest) {
```

```
        // Збереження замовлення в базу даних
        OrderEntity order = new OrderEntity(orderRequest.getName(),
orderRequest.getQuantity(), orderRequest.getDeliveryDate());
        orderRepository.save(order);

        // Реєстрація замовлення у блокчейн
        return blockchainService.registerOrder(order);
    }

    public OrderResponse getOrderById(Long id) {
        OrderEntity order = orderRepository.findById(id)
            .orElseThrow(() -> new EntityNotFoundException("Замовлення не
знайдено"));
        return new OrderResponse(order.getId(), order.getName(), order.getStatus());
    }
}
```

Сервіс для інтеграції з блокчейн

```
@Service
public class BlockchainService {

    private final Web3j web3j;
    private final Credentials credentials;
    private final Contract orderContract;

    public BlockchainService(Web3j web3j, Credentials credentials, Contract
orderContract) {
        this.web3j = web3j;
        this.credentials = credentials;
        this.orderContract = orderContract;
    }
```

```
    public String registerOrder(OrderEntity order) {
        try {
            TransactionReceipt receipt = orderContract.executeTransaction(
                order.getId(),
                order.getName(),
                order.getQuantity(),
                order.getDeliveryDate().toString()
            ).send();
            return receipt.getTransactionHash();
        } catch (Exception e) {
            throw new BlockchainException("Помилка реєстрації замовлення у
блокчейні", e);
        }
    }
}
```

Модель даних

```
@Entity
public class OrderEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private int quantity;
    private LocalDate deliveryDate;

    @Enumerated(EnumType.STRING)
    private OrderStatus status;
```

```
    // Конструктори, геттери, сеттери
}
```

Смарт-контракт (на прикладі Solidity)

```
pragma solidity ^0.8.0;

contract OrderContract {
    struct Order {
        uint id;
        string name;
        uint quantity;
        string deliveryDate;
    }

    mapping(uint => Order) public orders;

    function registerOrder(uint id, string memory name, uint quantity, string memory
deliveryDate) public {
        orders[id] = Order(id, name, quantity, deliveryDate);
    }

    function getOrder(uint id) public view returns (string memory, uint, string memory) {
        Order memory order = orders[id];
        return (order.name, order.quantity, order.deliveryDate);
    }
}
```

Фронтенд модуля інтеграції з постачальниками

Файл index.html

```
<div id="supplier-integration">
```

```
<header>
  <h1>Інтеграція з постачальниками</h1>
</header>
<main>
  <section id="supplier-list">
    <h2>Список постачальників</h2>
    <table>
      <thead>
        <tr>
          <th>Назва</th>
          <th>Матеріали</th>
          <th>Умови</th>
          <th>Дії</th>
        </tr>
      </thead>
      <tbody id="suppliers-table">
        <!-- Динамічне заповнення -->
      </tbody>
    </table>
  </section>
  <section id="request-form">
    <h2>Запит на пропозицію</h2>
    <form id="proposal-form">
      <label for="material">Матеріал:</label>
      <input type="text" id="material" name="material" required>
      <label for="quantity">Кількість:</label>
      <input type="number" id="quantity" name="quantity" required>
      <label for="delivery-date">Дата постачання:</label>
      <input type="date" id="delivery-date" name="delivery-date" required>
      <button type="submit">Надіслати запит</button>
    </form>
```

```
        </section>
        <section id="history">
            <h2>Історія запитів</h2>
            <table>
                <thead>
                    <tr>
                        <th>Матеріал</th>
                        <th>Кількість</th>
                        <th>Статус</th>
                        <th>Дата</th>
                    </tr>
                </thead>
                <tbody id="history-table">
                    <!-- Динамічне заповнення -->
                </tbody>
            </table>
        </section>
    </main>
</div>
```

Файл CSS

```
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f8f8f8;
}

header {
    background-color: #4CAF50;
    color: white;
```

```
    padding: 10px 20px;
    text-align: center;
}

main {
    margin: 20px auto;
    max-width: 1200px;
    background: white;
    border-radius: 8px;
    padding: 20px;
    box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1);
}

table {
    width: 100%;
    border-collapse: collapse;
    margin-bottom: 20px;
}

th, td {
    padding: 10px;
    border: 1px solid #ddd;
}

th {
    background-color: #f4f4f4;
    text-align: left;
}

button {
    background-color: #4CAF50;
```

```
    color: white;
    border: none;
    padding: 10px 15px;
    border-radius: 4px;
    cursor: pointer;
}

button:hover {
    background-color: #45a049;
}
```

Файл JavaScript

```
// Завантаження списку постачальників
document.addEventListener('DOMContentLoaded', loadSuppliers);

function loadSuppliers() {
    fetch('/api/suppliers')
        .then(response => response.json())
        .then(data => {
            const table = document.getElementById('suppliers-table');
            table.innerHTML = '';
            data.forEach(supplier => {
                const row = `
                    <tr>
                        <td>${supplier.name}</td>
                        <td>${supplier.materials}</td>
                        <td>${supplier.conditions}</td>
                        <td><button
onclick="sendRequest(${supplier.id})">Запит</button></td>
                    </tr>
                `;
```

```
            table.innerHTML += row;
        });
    });
}

// Надсилання запиту на пропозицію
document.getElementById('proposal-form').addEventListener('submit', function(e) {
    e.preventDefault();
    const material = document.getElementById('material').value;
    const quantity = document.getElementById('quantity').value;
    const deliveryDate = document.getElementById('delivery-date').value;

    fetch('/api/requests', {
        method: 'POST',
        headers: { 'Content-Type': 'application/json' },
        body: JSON.stringify({ material, quantity, deliveryDate })
    })
    .then(response => response.json())
    .then(data => {
        alert('Запит успішно надіслано!');
        loadRequestHistory();
    })
    .catch(error => console.error('Помилка:', error));
});

// Завантаження історії запитів
function loadRequestHistory() {
    fetch('/api/requests/history')
        .then(response => response.json())
        .then(data => {
            const table = document.getElementById('history-table');
```

```
        table.innerHTML = '';
        data.forEach(request => {
            const row = `
                <tr>
                    <td>${request.material}</td>
                    <td>${request.quantity}</td>
                    <td>${request.status}</td>
                    <td>${request.date}</td>
                </tr>
            `;
            table.innerHTML += row;
        });
    });
}
```

Бекенд модуля інтеграції з постачальниками

Контролер для управління постачальниками

```
@RestController
@RequestMapping("/api/suppliers")
public class SupplierController {

    private final SupplierService supplierService;

    public SupplierController(SupplierService supplierService) {
        this.supplierService = supplierService;
    }

    @GetMapping
    public List<SupplierResponse> getAllSuppliers() {
        return supplierService.getAllSuppliers();
    }
```

```
    @PostMapping
    public ResponseEntity<String> addSupplier(@RequestBody SupplierRequest
supplierRequest) {
        supplierService.addSupplier(supplierRequest);
        return ResponseEntity.ok("Постачальника додано успішно.");
    }
}
```

Контролер для обробки запитів

```
@RestController
@RequestMapping("/api/requests")
public class RequestController {

    private final RequestService requestService;

    public RequestController(RequestService requestService) {
        this.requestService = requestService;
    }

    @PostMapping
    public ResponseEntity<String> createRequest(@RequestBody RequestRequest
requestRequest) {
        String blockchainTx = requestService.createRequest(requestRequest);
        return ResponseEntity.ok("Запит створено. Blockchain Tx: " + blockchainTx);
    }

    @GetMapping("/history")
    public List<RequestResponse> getRequestHistory() {
        return requestService.getRequestHistory();
    }
```

```
}
```

Сервіс для управління постачальниками

```
@Service
public class SupplierService {

    private final SupplierRepository supplierRepository;

    public SupplierService(SupplierRepository supplierRepository) {
        this.supplierRepository = supplierRepository;
    }

    public List<SupplierResponse> getAllSuppliers() {
        return supplierRepository.findAll().stream()
            .map(supplier -> new SupplierResponse(supplier.getId(), supplier.getName(),
supplier.getMaterials(), supplier.getConditions()))
            .collect(Collectors.toList());
    }

    public void addSupplier(SupplierRequest supplierRequest) {
        SupplierEntity supplier = new SupplierEntity(supplierRequest.getName(),
supplierRequest.getMaterials(), supplierRequest.getConditions());
        supplierRepository.save(supplier);
    }
}
```

Сервіс для інтеграції запитів із блокчейн

```
@Service
public class RequestService {

    private final RequestRepository requestRepository;
```

```
    private final BlockchainService blockchainService;

    public RequestService(RequestRepository requestRepository, BlockchainService
blockchainService) {
        this.requestRepository = requestRepository;
        this.blockchainService = blockchainService;
    }

    public String createRequest(RequestRequest requestRequest) {
        // Збереження запиту в базу даних
        RequestEntity request = new RequestEntity(requestRequest.getMaterial(),
requestRequest.getQuantity(), requestRequest.getDeliveryDate());
        requestRepository.save(request);

        // Реєстрація запиту у блокчейн
        return blockchainService.registerRequest(request);
    }

    public List<RequestResponse> getRequestHistory() {
        return requestRepository.findAll().stream()
            .map(request -> new RequestResponse(request.getMaterial(),
request.getQuantity(), request.getStatus(), request.getDate()))
            .collect(Collectors.toList());
    }
}
```

Сервіс для інтеграції з блокчейн

```
@Service
public class BlockchainService {

    private final Web3j web3j;
```

```
    private final Credentials credentials;
    private final Contract requestContract;

    public BlockchainService(Web3j web3j, Credentials credentials, Contract
requestContract) {
        this.web3j = web3j;
        this.credentials = credentials;
        this.requestContract = requestContract;
    }

    public String registerRequest(RequestEntity request) {
        try {
            TransactionReceipt receipt = requestContract.executeTransaction(
                request.getId(),
                request.getMaterial(),
                request.getQuantity(),
                request.getDeliveryDate().toString()
            ).send();
            return receipt.getTransactionHash();
        } catch (Exception e) {
            throw new BlockchainException("Помилка реєстрації запиту у блокчейні",
e);
        }
    }
}
```

Модель даних

```
@Entity
public class SupplierEntity {

    @Id
```

```
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String name;
    private String materials;
    private String conditions;

    // Конструктори, геттери, сеттери
}

@Entity
public class RequestEntity {

    @Id
    @GeneratedValue(strategy = GenerationType.IDENTITY)
    private Long id;

    private String material;
    private int quantity;
    private LocalDate deliveryDate;

    @Enumerated(EnumType.STRING)
    private RequestStatus status;

    // Конструктори, геттери, сеттери
}
```

Конфігураційний файл для розгортання трьох вузлів в мережі блокчейну з метою тестування програмного засобу

```
version: "3.8"
services:
```

```
  node1:
    build: .
    container_name: eth_node1
    ports:
      - "8545:8545"
    volumes:
      - "./data/node1:/data"
    networks:
      - blockchain
  node2:
    build: .
    container_name: eth_node2
    ports:
      - "8546:8545"
    volumes:
      - "./data/node2:/data"
    networks:
      - blockchain
  node3:
    build: .
    container_name: eth_node3
    ports:
      - "8547:8545"
    volumes:
      - "./data/node3:/data"
    networks:
      - blockchain
networks:
  blockchain:
    driver: bridge
```

ДОДАТОК Г

Графічна частина

ГРАФІЧНА ЧАСТИНА

Розробка програмного засобу для управління фінансуванням станцій технічного обслуговування

Рисунок Д.1 - Слайд презентації 1

Розробка програмного засобу для управління фінансуванням станцій технічного обслуговування

Виконав:
студент групи 2ПІ-21б

Франчук О.В.

Керівник:
к.т.н., доц. каф. ПЗ Хошаба О.М.

Рисунок Д.2 - Слайд презентації 2

Метою роботи є підвищення ефективності управління фінансуванням станцій технічного обслуговування за рахунок розробки та впровадження програмного засобу.

Об’єктом дослідження є процес управління фінансуванням станцій технічного обслуговування в умовах цифровізації економіки та високої конкуренції в галузі сервісного обслуговування.

Предметом дослідження є методи, моделі та програмні засоби, що дозволяють автоматизувати та оптимізувати управління фінансами СТО, зокрема в частині обліку доходів і витрат, моніторингу грошових потоків, бюджетного планування та інтеграції аналітичних інструментів.

Рисунок Д.3 - Слайд презентації 3

Відповідно до поставленої мети виконані наступні задачі:

- виконати аналіз предметної галузі та особливості управління фінансуванням станцій технічного обслуговування;
- виконати порівняльний аналіз програмних інструментів для управління фінансуванням комерційних організацій
- запропонувати метод комплексного фінансового контролінгу;
- розробити функціональні вимоги до програмного засобу;
- виконати проектування основних та додаткових модулів програмного засобу;
- визначити та розробити основні принципи програмного засобу на основі блокчейну;
- визначити та розробити основні принципи роботи модуля управління замовленнями на основі блокчейн;
- визначити та розробити основні принципи роботи модуля інтеграції з постачальниками програмного засобу;
- визначити та розробити особливості тестування мікросервісної архітектури програмного засобу;
- виконати тестування програмного модуля управління замовленнями;
- виконати тестування програмного модуля інтеграції з постачальниками

Рисунок Д.4 - Слайд презентації 4

Актуальність теми дослідження полягає в наступному:

Сучасний стан розвитку паливно-технічної галузі в Україні та світі зумовлює високий попит на ефективні методи управління фінансуванням станцій технічного обслуговування (СТО).

Зростання кількості автомобілів, ускладнення технічних процесів ремонту та підвищення конкуренції серед СТО потребують впровадження сучасних інформаційних технологій для автоматизації управлінських рішень, особливо в частині фінансового моніторингу, бюджетування та оптимізації витрат.

Тому, актуальність теми полягає у необхідності комплексного вирішення проблем управління фінансуванням СТО шляхом створення спеціалізованого програмного засобу, що враховує специфіку галузі, підтримує сучасні цифрові інструменти (ERP, блокчейн, IoT) та забезпечує інтеграцію з існуючими бізнес-процесами.

Таким чином, розробка власного програмного засобу для управління фінансуванням СТО є актуальним, затребуваним та інноваційно значущим завданням.

Рисунок Д.5 - Слайд презентації 5

Порівняльний аналіз програмних інструментів для управління фінансуванням комерційних організацій

Назва інструменту	Опис
Octane Systems	Система для управління запасами, моніторингу відповідності та автоматизованого виставлення рахунків.
HB Smart Fuel Station Solutions	Інтегровані модулі для управління фінансами, запасами та персоналом.
PDI Enterprise	Рішення для обліку, автоматизації замовлень і управління запасами у паливній галузі.
SAP ERP	Комплексна система управління підприємством із модулями для технічного обслуговування.
BAS ERP	Українське рішення для автоматизації управління фінансами, закупівлями та персоналом.
LEAFIO	Рішення з алгоритмами штучного інтелекту для автоматизації управління запасами.
SmartEAM	Система для управління технічним обслуговуванням, планування робіт та ведення звітності.
Modisoft Fuel Management System	Контроль продажів палива, управління запасами, інтеграція з POS-системами.
Hectronic	Рішення для авторизації, оплати, управління та вимірювання палива.
Petrolsoft Corporation	Програмні рішення для оптимізації дистрибуції та торгівлі паливом.
Verifone	Інтегровані POS-системи та термінали для обслуговування клієнтів на заправних станціях.
LS Retail	Рішення для управління паливними насосами, автоматизації запасів та обслуговування клієнтів.
POS Nation	POS-системи для магазинів на заправках, які продають тютюн, алкоголь, лотереї тощо.
Petrosoft SmartPOS	Управління обліком клієнтів, замовленнями палива та звітністю для комерційних продажів.
NCR Storefront	Обладнання та POS-термінали для модернізації паливних насосів на заправних станціях.

Рисунок Д.6 - Слайд презентації 6
UML-діаграма послідовності процесу взаємодії постачальників із замовниками матеріалів для СТО

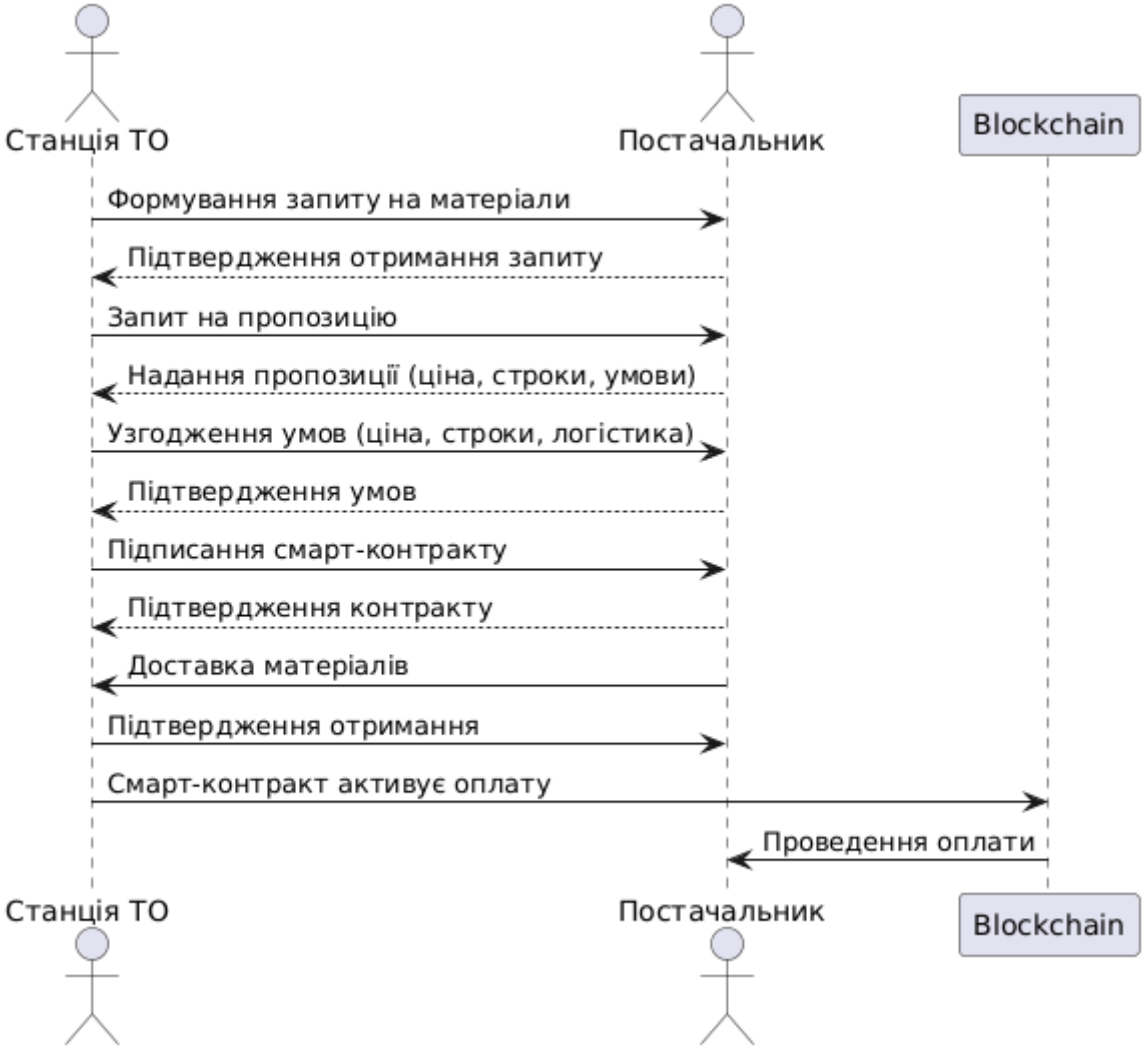

Рисунок Д.7 - Слайд презентації 7
Алгоритм виконання процесу взаємодії постачальників із замовниками матеріалів для СТО на основі UML-діаграми активності

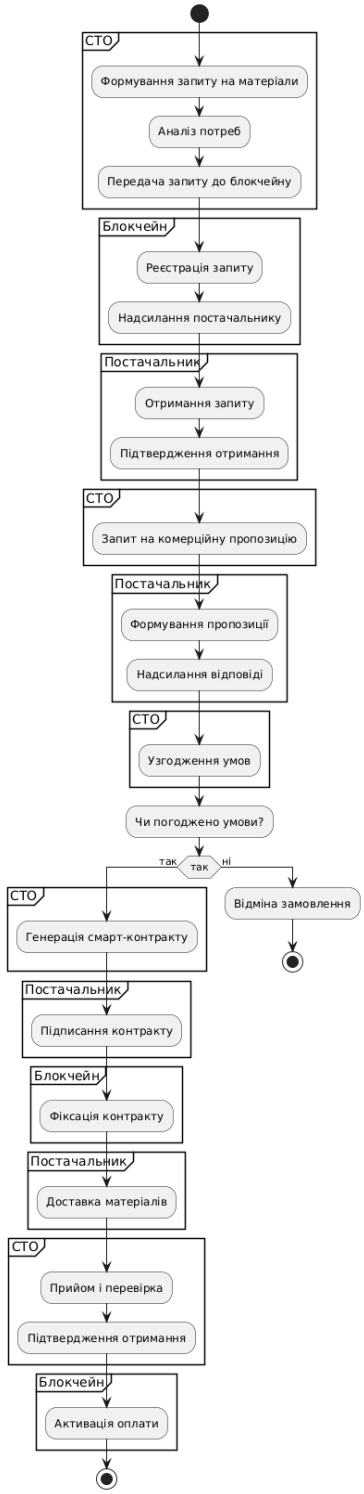

Рисунок Д.8 - Слайд презентації 8
UML діаграма класів модуля управління замовленнями на основі мови скрипта PlantUML

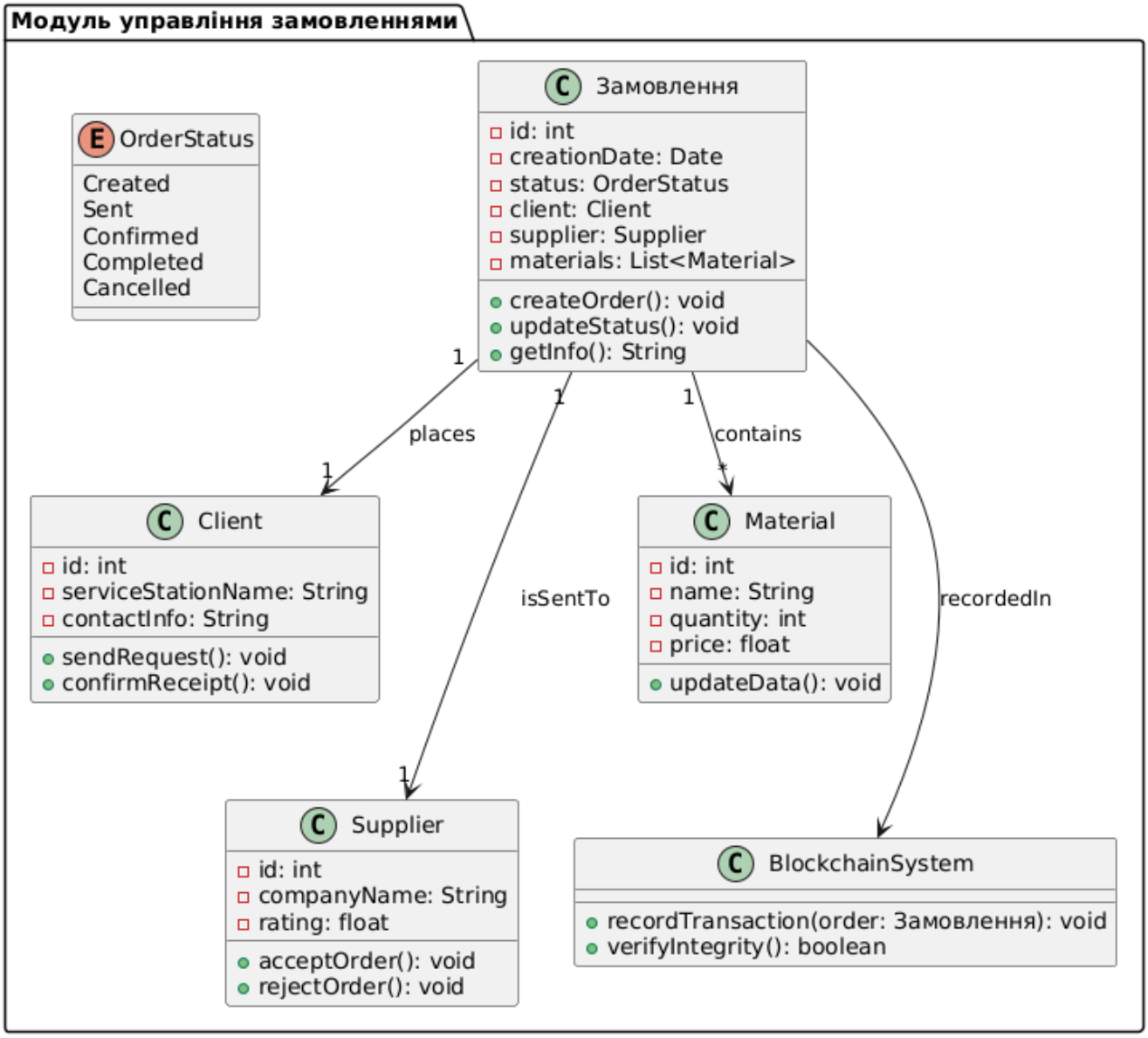

Рисунок Д.9 - Слайд презентації 9
Скріншот результатів початкового тестування з формуванням основних полів для запиту на пропозицію через API (програмний інструмент Postman)

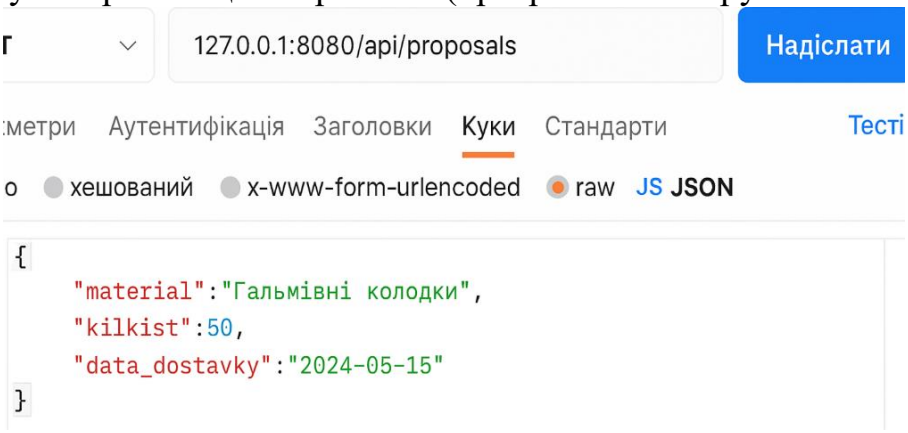

Рисунок Д.10 - Слайд презентації 10
Скріншот результатів кінцевого тестування з отриманням повідомлення про успішне збереження запиту з фіксацією основних полів в блокчейні

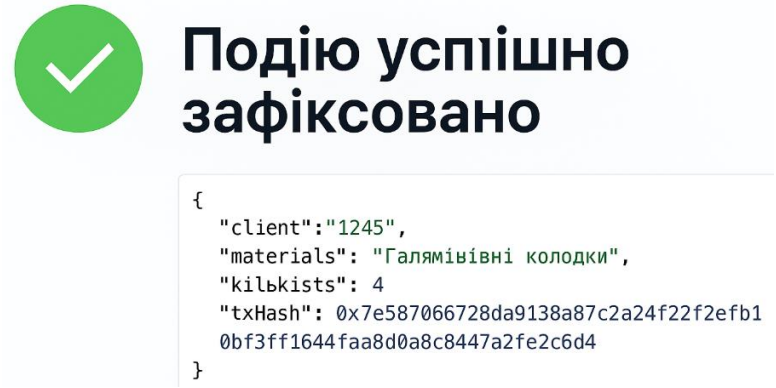

Рисунок Д.11 - Слайд презентації 11
Новизна отриманих результатів:

1. Запропоновано метод комплексного фінансового контролінгу для СТО, особливість якого полягає в інтеграції даних про доходи, витрати та операційні показники у режимі реального часу на основі блокчейн мереж, що дає можливість оперативно виявляти фінансові відхилення і приймати обґрунтовані управлінські рішення.
2. Запропоновано модель бюджетного планування діяльності СТО, яка враховує сезонні коливання попиту на послуги та витрати на обслуговування обладнання, що дозволило підвищити точність фінансового планування та ефективність розподілу ресурсів на 12%.
3. Подальшого розвитку отримала інформаційна система управління фінансами СТО, у якій, на відміну від існуючих, реалізовано інтеграцію з системами управління замовленнями та запасами на основі блокчейн мережі, що дозволило забезпечити актуальність даних і підвищити точність фінансового аналізу на 17%.

Рисунок Д.12 - Слайд презентації 12
Висновки:

У бакалаврській кваліфікаційній роботі:
- виконано аналіз предметної галузі та особливості управління фінансуванням станцій технічного обслуговування;
- виконано порівняльний аналіз програмних інструментів для управління фінансуванням комерційних організацій
- запропоновані методи комплексного фінансового контролінгу;
- розроблені функціональні вимоги до програмного засобу;
- виконано проектування основних та додаткових модулів програмного засобу;
- визначити та розробити основні принципи програмного засобу на основі блокчейну;
- визначити та розробити основні принципи роботи модуля управління замовленнями на основі блокчейн;
- визначити та розробити основні принципи роботи модуля інтеграції з постачальниками програмного засобу;
- визначити та розробити особливості тестування мікросервісної архітектури програмного засобу;
- виконано тестування програмного модуля управління замовленнями;
- виконано тестування програмного модуля інтеграції з постачальниками програмного засобу.